Documentatie proiect

Descriere funcțională

Această aplicație este un joc de tip **"Fighting Game"**, inspirat de stilul clasic al seriei Mortal Kombat. Este un joc pentru doi jucători locali, în care fiecare controlează un personaj. Obiectivul este să reduci sănătatea adversarului la zero prin atacuri.

Caracteristici principale:

1. **Mișcare laterală și sărituri**:
   - Jucătorii se pot deplasa stânga-dreapta și pot sări.
2. **Sistem de sănătate**:
   - Fiecare jucător începe cu 100 de puncte de sănătate.
   - Atacurile reduc sănătatea adversarului.
3. **Tipuri de atacuri**:
   - Atac ușor (reduce puțină sănătate).
   - Atac greu (reduce mai multă sănătate).
4. **Finalizarea jocului**:
   - Când sănătatea unui jucător ajunge la zero, jocul afișează numele câștigătorului în consolă și se oprește.

Descriere backend și fluxul aplicației

1. **Pornirea aplicației**:
   - Când aplicația este lansată, controlul merge la funcția `main()` definită în fișierul principal.
   - Aceasta inițializează jocul, pornește bucla principală și afișează fereastra jocului.
2. **Inițializare**:
   - Funcția `main()` apelează diferite module `pygame` pentru a seta ferestrele, FPS-ul și variabilele globale, precum sănătatea și pozițiile jucătorilor.
3. **Bucla principală**:
   - Bucla principală din `main()`:
     - Gestionează evenimentele (cum ar fi apăsările de taste sau ieșirea din aplicație).
     - Apelează funcțiile de manipulare a mișcării (`handle_movement()`), coliziunilor (`handle_collision()`) și desenare (`draw_window()`).
4. **Mișcările jucătorilor**:
   - În funcția `handle_movement()`:
     - Tastele `A` și `D` controlează deplasarea jucătorului 1, iar tastele săgeți stânga și dreapta controlează deplasarea jucătorului 2.

        - Atât jucătorul 1, cât și jucătorul 2 pot sări folosind `W` și săgeata sus, respectiv.
5. **Coliziuni și atacuri**:
    - Funcția `handle_collision()`:
        - Verifică dacă dreptunghiurile celor doi jucători se intersectează.
        - Dacă există o coliziune, se verifică dacă jucătorii au apăsat tastele corespunzătoare pentru atacuri.
        - Atacurile scad sănătatea adversarului, fie cu 1 punct (atac ușor), fie cu 3 puncte (atac greu).
6. **Desenarea ferestrei**:
    - Funcția `draw_window()`:
        - Umple ecranul cu negru.
        - Desenează barele de sănătate pentru ambii jucători.
        - Desenează jucătorii ca dreptunghiuri colorate (roșu pentru jucătorul 1 și albastru pentru jucătorul 2).
        - Actualizează fereastra pentru a reflecta modificările.
7. **Finalizarea jocului**:
    - În bucla principală, se verifică periodic dacă sănătatea unuia dintre jucători a ajuns la zero.
    - Dacă acest lucru se întâmplă, numele câștigătorului este afișat în consolă, iar jocul se oprește prin `pygame.quit()` și `sys.exit()`.

Exemplu concret al fluxului

- **Deschid aplicația**:
    - Se apelează funcția `main()`.
    - Jocul afișează fereastra de joc și intră în bucla principală.
- **Apăs tastele `A` sau `D`**:
    - Funcția `handle_movement()` detectează apăsarea și mută jucătorul 1 la stânga sau dreapta.
- **Apăs `Q` pentru atac ușor**:
    - Funcția `handle_collision()` detectează coliziunea între jucători și aplică reducerea de sănătate pentru jucătorul 2.
- **Un jucător rămâne fără sănătate**:
    - Funcția `main()` detectează condiția de victorie, afișează mesajul câștigătorului și închide jocul.

Explicatii cod:

1. **`import pygame`** - Importă biblioteca Pygame, utilizată pentru crearea jocurilor.
2. **`import sys`** - Importă biblioteca Sys pentru a permite ieșirea din program.

3. `pygame.init()` - Inițializează toate modulele Pygame.
4. `WIDTH, HEIGHT = 800, 400` - Definește dimensiunile ferestrei jocului (lățime și înălțime).
5. `SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))` - Creează fereastra de joc.
6. `pygame.display.set_caption("Fighting Game - Mortal Kombat Style")` - Setează titlul ferestrei.
7. `WHITE = (255, 255, 255)` - Definește culoarea albă în format RGB.
8. `BLACK = (0, 0, 0)` - Definește culoarea neagră în format RGB.
9. `RED = (255, 0, 0)` - Definește culoarea roșie.
10. `BLUE = (0, 0, 255)` - Definește culoarea albastră.
11. `FPS = 60` - Setează numărul maxim de cadre pe secundă.
12. `clock = pygame.time.Clock()` - Creează un obiect de tip ceas pentru sincronizarea cadrelor.
13. `PLAYER_WIDTH, PLAYER_HEIGHT = 50, 100` - Dimensiunile jucătorilor (lățime și înălțime).
14. `PLAYER_SPEED = 5` - Viteza de mișcare a jucătorilor.
15. `PLAYER1_START = (100, HEIGHT - PLAYER_HEIGHT - 10)` - Poziția de start a primului jucător.
16. `PLAYER2_START = (WIDTH - 150, HEIGHT - PLAYER_HEIGHT - 10)` - Poziția de start a celui de-al doilea jucător.
17. `PLAYER1_HEALTH = 100` - Sănătatea inițială a primului jucător.
18. `PLAYER2_HEALTH = 100` - Sănătatea inițială a celui de-al doilea jucător.
19. `player1 = pygame.Rect(*PLAYER1_START, PLAYER_WIDTH, PLAYER_HEIGHT)` - Creează un dreptunghi pentru primul jucător.
20. `player2 = pygame.Rect(*PLAYER2_START, PLAYER_WIDTH, PLAYER_HEIGHT)` - Creează un dreptunghi pentru al doilea jucător.
21. `def draw_window():` - Funcția care desenează elementele jocului.
22. `SCREEN.fill(BLACK)` - Umple ecranul cu negru.
23. `pygame.draw.rect(SCREEN, RED, (20, 20, PLAYER1_HEALTH * 2, 20))` - Desenează bara de sănătate a primului jucător.
24. `pygame.draw.rect(SCREEN, BLUE, (WIDTH - 220, 20, PLAYER2_HEALTH * 2, 20))` - Desenează bara de sănătate a celui de-al doilea jucător.
25. `pygame.draw.rect(SCREEN, RED, player1)` - Desenează dreptunghiul pentru primul jucător.
26. `pygame.draw.rect(SCREEN, BLUE, player2)` - Desenează dreptunghiul pentru al doilea jucător.
27. `pygame.display.flip()` - Actualizează afișajul.
28. `def handle_collision():` - Funcția care gestionează coliziunile și atacurile.
29. `if player1.colliderect(player2):` - Verifică dacă jucătorii se ating.
30. `if keys[pygame.K_q]:` - Atac ușor al primului jucător.
31. `PLAYER2_HEALTH -= 1` - Reduce sănătatea celui de-al doilea jucător.
32. `if keys[pygame.K_w]:` - Atac greu al primului jucător.
33. `PLAYER2_HEALTH -= 3` - Reduce mai mult sănătatea celui de-al doilea jucător.
34. `if keys[pygame.K_KP1]:` - Atac ușor al celui de-al doilea jucător.
35. `PLAYER1_HEALTH -= 1` - Reduce sănătatea primului jucător.

36. `if keys[pygame.K_KP2]:` - Atac greu al celui de-al doilea jucător.
37. `PLAYER1_HEALTH -= 3` - Reduce mai mult sănătatea primului jucător.
38. `def handle_movement(keys):` - Funcția care gestionează mișcarea jucătorilor.
39. `if keys[pygame.K_a] and player1.x - PLAYER_SPEED > 0:` - Mișcarea la stânga a primului jucător.
40. `if keys[pygame.K_d] and player1.x + PLAYER_SPEED + PLAYER_WIDTH < WIDTH:` - Mișcarea la dreapta a primului jucător.
41. `if keys[pygame.K_w]:` - Săritura primului jucător.
42. `if player1.y < HEIGHT - PLAYER_HEIGHT - 10:` - Revenirea pe sol a primului jucător.
43. **(similar pentru al doilea jucător)**
44. `def main():` - Funcția principală.
45. `while running:` - Bucla principală a jocului.
46. `clock.tick(FPS)` - Asigură sincronizarea cadrelor.
47. `if event.type == pygame.QUIT:` - Permite închiderea jocului.
48. `handle_movement(keys)` - Gestionează mișcarea.
49. `handle_collision()` - Gestionează coliziunile.
50. `draw_window()` - Actualizează ecranul.
51. `if PLAYER1_HEALTH <= 0 or PLAYER2_HEALTH <= 0:` - Verifică dacă există un câștigător.
52. `pygame.quit()` - Închide Pygame.
53. `sys.exit()` - Iese din program.
54. `if __name__ == "__main__":` - Rulează funcția principală dacă fișierul este executat direct.

Pentru a rula jocul am folosit VisualStudio Code unde am scris codul iar pentru a putea juca am folosit comanda pip install pygame ulterior verficand cu python -m pygame –version, pe care le-am pus in terminal.

Referinte:

https://www.youtube.com/watch?v=y9VG3Pztok8
https://www.w3schools.com/python/default.asp