

Documentatie

prerequisites & run

1. Se instaleaza VsCode de pe siteul oficial
 - i. ([Download Visual Studio Code - Mac, Linux, Windows](#))
2. Dupa instalare se deschide VsCode
3. Descarcam Python (de preferat o versiune stabila)

Python 3.13.1 64bit
4. In VsCode se alege ca interpretor versiunea de Python instalata (de aceea trebuia bifata optiunea Path de la punctul 3.)
5. (O metoda foarte simpla este:)Deschid programul cu VsCode->ctrl+shift+p-> Python: Select Interpreter->Selectez versiunea de Python dorita
6. Se deschide fisierul rogue.py cu VsCode

Explicații

Acest script implementează un joc simplu într-un spațiu bidimensional de 10x10, în care un jucător interacționează cu monștri. Vom detalia funcțiile și componentele programului:

1. Definiții și inițializări:

- **Variabile globale:**
 - WIDTH și HEIGHT: Dimensiunile hărții (10x10).
 - MAP: Reprezintă harta, inițializată cu puncte (.) pentru a simboliza spații libere.
 - player_position: Poziția inițială a jucătorului pe hartă (5, 5).

- `player_health`: Sănătatea inițială a jucătorului (100).
 - `player_symbol`: Simbolul care reprezintă jucătorul pe hartă (@).
 - `monster_health`: Sănătatea inițială a unui monstru (20).
 - `monster_symbol`: Simbolul care reprezintă un monstru (M).
 - `monsters`: Listă care va conține monștrii, fiecare reprezentat ca un dicționar cu poziția și sănătatea.
-

2. Funcții principale:

a. `spawn_monsters(num_monsters)`

Această funcție generează un număr specificat de monștri pe harta 10x10:

- Utilizează `random.randint` pentru a genera poziții aleatorii.
- Fiecare monstru este reprezentat printr-un dicționar cu două chei:
 - `position`: Poziția pe hartă (ex.: `[y, x]`).
 - `health`: Sănătatea inițială (20).

b. `draw_map()`

Desenează harta în consolă:

- Iterează prin toate pozițiile (linii și coloane) din hartă.
- Verifică:
 - Dacă poziția curentă este a jucătorului (`player_position`).
 - Dacă există un monstru pe poziția curentă.
 - Dacă nu e nimic, afișează un punct (.

c. `move_player(direction)`

Permite jucătorului să se deplaseze în cele patru direcții:

- w: Sus.
- s: Jos.
- a: Stânga.
- d: Dreapta.
- Verifică limitele hărții, pentru a nu ieși din aceasta.

d. `fight(monster)`

Inițiază o luptă între jucător și un monstru aflat pe aceeași poziție:

- Jucătorul atacă:
 - Daunele produse sunt aleatorii între 5 și 10.
 - Dacă sănătatea monstrului scade sub 0, lupta se încheie.
- Monstrul atacă:
 - Daunele produse sunt aleatorii între 1 și 5.
 - Dacă sănătatea jucătorului scade sub 0, jocul se termină.

- Afișează în mod constant sănătatea ambelor părți.

e. `check_win()`

Verifică dacă toți monștrii au fost eliminați:

- Dacă `monsters` este gol (toți monștrii au fost învinși), afișează un mesaj de câștig.

f. `game_loop()`

Conține bucla principală a jocului:

- Generează 3 monștri folosind `spawn_monsters`.
 - Afișează harta și sănătatea jucătorului.
 - Permite:
 - Mișcarea jucătorului (w, a, s, d).
 - Inițierea unei lupte (p), doar dacă un monstru este pe aceeași poziție.
 - Actualizează lista de monștri după lupte (elimină monștrii cu sănătate ≤ 0).
 - Verifică dacă jucătorul câștigă sau moare, oprind bucla corespunzător.
-

3. Fluxul jocului:

1. Harta este afișată.
 2. Jucătorul introduce o comandă:
 - Se poate deplasa.
 - Poate lupta dacă există un monstru pe poziția sa.
 3. Se actualizează starea jocului:
 - Poziția jucătorului.
 - Sănătatea jucătorului și a monștrilor.
 4. Jocul se termină dacă:
 - Toți monștrii au fost eliminați.
 - Jucătorul moare.
-

Îmbunătățiri posibile:

1. **Harta interactivă:**
 - Adăugarea de obstacole sau alte elemente pentru a diversifica jocul.
2. **Inteligența artificială pentru monștri:**
 - Monștrii să se deplaseze spre jucător.
3. **Diversitate în arme și atacuri:**
 - Introducerea mai multor opțiuni de atac pentru jucător.
4. **Salvarea progresului:**
 - Permite jucătorului să continue jocul ulterior.

Acest cod simplu poate fi extins în multe moduri pentru a crea un joc mai complex și captivant.

Explicația a fost realizată cu Chatgpt

```
explica indetaliat
import random

WIDTH = 10
HEIGHT = 10

MAP = [['.' for _ in range(WIDTH)] for _ in range(HEIGHT)]
player_position = [5, 5]
player_health = 100
player_symbol = '@'
monster_health = 20
monster_symbol = 'M'
monsters = []

def spawn_monsters(num_monsters):
    for _ in range(num_monsters):
        monster_pos = [random.randint(0, HEIGHT-1), random.randint(0,
WIDTH-1)]
        monsters.append({'position': monster_pos, 'health':
```

prerequisites & run

1. Se instaleaza VsCode de pe siteul oficial

i.

([Do](#)
[wnl](#)
[oad](#)
[Vis](#)
[ual](#)
[Stu](#)

2. După instalare se deschide VSCode
3. Descarcăm Python (de preferat o versiune stabilă)
4. În VSCode se alege ca interpretor versiunea de Python instalată (de aceea trebuia bifată opțiunea Path de la punctul 3.)
5. (O metodă foarte simplă este:) Deschid programul cu VSCode->ctrl+shift+p-> Python: Select Interpreter->Selectez versiunea de Python dorită
6. Se deschide cu VSCode
7. Se scrie programul
8. Se salvează programul (ca fișiere .py)
9. Se rulează Codurile

Explicații

Acest script implementează un joc simplu într-un spațiu bidimensional de 10x10, în care un jucător interacționează cu monștri. Vom detalia funcțiile și componentele programului:

1. Definiții și inițializări:

- **Variabile globale:**
 - WIDTH și HEIGHT: Dimensiunile hărții (10x10).
 - MAP: Reprezintă harta, inițializată cu puncte (.) pentru a simboliza spații libere.
 - player_position: Poziția inițială a jucătorului pe hartă (5, 5).
 - player_health: Sănătatea inițială a jucătorului (100).

- `player_symbol`: Simbolul care reprezintă jucătorul pe hartă (@).
 - `monster_health`: Sănătatea inițială a unui monstru (20).
 - `monster_symbol`: Simbolul care reprezintă un monstru (M).
 - `monsters`: Listă care va conține monștrii, fiecare reprezentat ca un dicționar cu poziția și sănătatea.
-

2. Funcții principale:

a. `spawn_monsters(num_monsters)`

Această funcție generează un număr specificat de monștri pe harta 10x10:

- Utilizează `random.randint` pentru a genera poziții aleatorii.
- Fiecare monstru este reprezentat printr-un dicționar cu două chei:
 - `position`: Poziția pe hartă (ex.: [y, x]).
 - `health`: Sănătatea inițială (20).

b. `draw_map()`

Desenează harta în consolă:

- Iterează prin toate pozițiile (linii și coloane) din hartă.
- Verifică:
 - Dacă poziția curentă este a jucătorului (`player_position`).
 - Dacă există un monstru pe poziția curentă.
 - Dacă nu e nimic, afișează un punct (.

c. `move_player(direction)`

Permite jucătorului să se deplaseze în cele patru direcții:

- w: Sus.
- s: Jos.
- a: Stânga.
- d: Dreapta.
- Verifică limitele hărții, pentru a nu ieși din aceasta.

d. `fight(monster)`

Inițiază o luptă între jucător și un monstru aflat pe aceeași poziție:

- Jucătorul atacă:
 - Daunele produse sunt aleatorii între 5 și 10.
 - Dacă sănătatea monstrului scade sub 0, lupta se încheie.
- Monstrul atacă:
 - Daunele produse sunt aleatorii între 1 și 5.
 - Dacă sănătatea jucătorului scade sub 0, jocul se termină.
- Afișează în mod constant sănătatea ambelor părți.

e. `check_win()`

Verifică dacă toți monștrii au fost eliminați:

- Dacă `monsters` este gol (toți monștrii au fost învinși), afișează un mesaj de câștig.

f. `game_loop()`

Conține bucla principală a jocului:

- Generează 3 monștri folosind `spawn_monsters`.
 - Afișează harta și sănătatea jucătorului.
 - Permite:
 - Mișcarea jucătorului (w, a, s, d).
 - Inițierea unei lupte (p), doar dacă un monstru este pe aceeași poziție.
 - Actualizează lista de monștri după lupte (elimină monștrii cu sănătate ≤ 0).
 - Verifică dacă jucătorul câștigă sau moare, oprind bucla corespunzător.
-

3. Fluxul jocului:

5. Harta este afișată.
 6. Jucătorul introduce o comandă:
 - Se poate deplasa.
 - Poate lupta dacă există un monstru pe poziția sa.
 7. Se actualizează starea jocului:
 - Poziția jucătorului.
 - Sănătatea jucătorului și a monștrilor.
 8. Jocul se termină dacă:
 - Toți monștrii au fost eliminați.
 - Jucătorul moare.
-

Îmbunătățiri posibile:

5. **Harta interactivă:**
 - Adăugarea de obstacole sau alte elemente pentru a diversifica jocul.
6. **Inteligența artificială pentru monștri:**
 - Monștrii să se deplaseze spre jucător.
7. **Diversitate în arme și atacuri:**
 - Introducerea mai multor opțiuni de atac pentru jucător.
8. **Salvarea progresului:**
 - Permite jucătorului să continue jocul ulterior.

Acest cod simplu poate fi extins în multe moduri pentru a crea un joc mai complex și captivant.

Explicația a fost realizată cu Chatgpt

```

explica indetaliat
import random

WIDTH = 10
HEIGHT = 10

MAP = [['.' for _ in range(WIDTH)] for _ in range(HEIGHT)]
player_position = [5, 5]
player_health = 100
player_symbol = '@'
monster_health = 20
monster_symbol = 'M'
monsters = []

def spawn_monsters(num_monsters):
    for _ in range(num_monsters):
        monster_pos = [random.randint(0, HEIGHT-1), random.randint(0,
WIDTH-1)]
        monsters.append({'position': monster_pos, 'health':

```

prerequisites & run

1. Se instaleaza VsCode de pe siteul oficial

i.

([Do](#)
[wnl](#)
[oad](#)
[Vis](#)
[ual](#)
[Stu](#)
[dio](#)
[Co](#)
[de -](#)
[Ma](#)
[c.](#)

2. După instalare se deschide VsCode
3. Descarcăm Python (de preferat o versiune stabilă)
4. În VsCode se alege ca interpretor versiunea de Python instalată (de aceea trebuia bifată opțiunea Path de la punctul 3.)
5. (O metodă foarte simplă este:) Deschid programul cu VsCode->ctrl+shift+p-> Python: Select Interpreter->Selectez versiunea de Python dorită
6. Se deschide cu VsCode
7. Se scrie programul
8. Se salvează programul (ca fișiere .py)
9. Se rulează Codurile

Explicații

Acest script implementează un joc simplu într-un spațiu bidimensional de 10x10, în care un jucător interacționează cu monștri. Vom detalia funcțiile și componentele programului:

1. Definiții și inițializări:

- **Variabile globale:**
 - WIDTH și HEIGHT: Dimensiunile hărții (10x10).
 - MAP: Reprezintă harta, inițializată cu puncte (.) pentru a simboliza spații libere.
 - player_position: Poziția inițială a jucătorului pe hartă (5, 5).
 - player_health: Sănătatea inițială a jucătorului (100).
 - player_symbol: Simbolul care reprezintă jucătorul pe hartă (@).
 - monster_health: Sănătatea inițială a unui monstru (20).
 - monster_symbol: Simbolul care reprezintă un monstru (M).
 - monsters: Listă care va conține monștrii, fiecare reprezentat ca un dicționar cu poziția și sănătatea.

2. Funcții principale:

a. `spawn_monsters(num_monsters)`

Această funcție generează un număr specificat de monștri pe harta 10x10:

- Utilizează `random.randint` pentru a genera poziții aleatorii.
- Fiecare monstru este reprezentat printr-un dicționar cu două chei:
 - `position`: Poziția pe hartă (ex.: `[y, x]`).
 - `health`: Sănătatea inițială (20).

b. `draw_map()`

Desenează harta în consolă:

- Iterează prin toate pozițiile (linii și coloane) din hartă.
- Verifică:
 - Dacă poziția curentă este a jucătorului (`player_position`).
 - Dacă există un monstru pe poziția curentă.
 - Dacă nu e nimic, afișează un punct (`.`).

c. `move_player(direction)`

Permite jucătorului să se deplaseze în cele patru direcții:

- `w`: Sus.
- `s`: Jos.
- `a`: Stânga.
- `d`: Dreapta.
- Verifică limitele hărții, pentru a nu ieși din aceasta.

d. `fight(monster)`

Inițiază o luptă între jucător și un monstru aflat pe aceeași poziție:

- Jucătorul atacă:
 - Daunele produse sunt aleatorii între 5 și 10.
 - Dacă sănătatea monstrului scade sub 0, lupta se încheie.
- Monstrul atacă:
 - Daunele produse sunt aleatorii între 1 și 5.
 - Dacă sănătatea jucătorului scade sub 0, jocul se termină.
- Afișează în mod constant sănătatea ambelor părți.

e. `check_win()`

Verifică dacă toți monștrii au fost eliminați:

- Dacă `monsters` este gol (toți monștrii au fost învinși), afișează un mesaj de câștig.

f. game_loop()

Conține bucla principală a jocului:

- Generează 3 monștri folosind spawn_monsters.
 - Afișează harta și sănătatea jucătorului.
 - Permite:
 - Mișcarea jucătorului (w, a, s, d).
 - Inițierea unei lupte (p), doar dacă un monstru este pe aceeași poziție.
 - Actualizează lista de monștri după lupte (elimină monștrii cu sănătate ≤ 0).
 - Verifică dacă jucătorul câștigă sau moare, oprind bucla corespunzător.
-

3. Fluxul jocului:

9. Harta este afișată.
 10. Jucătorul introduce o comandă:
 - Se poate deplasa.
 - Poate lupta dacă există un monstru pe poziția sa.
 11. Se actualizează starea jocului:
 - Poziția jucătorului.
 - Sănătatea jucătorului și a monștrilor.
 12. Jocul se termină dacă:
 - Toți monștrii au fost eliminați.
 - Jucătorul moare.
-

Îmbunătățiri posibile:

9. **Harta interactivă:**
 - Adăugarea de obstacole sau alte elemente pentru a diversifica jocul.
10. **Inteligența artificială pentru monștri:**
 - Monștrii să se deplaseze spre jucător.
11. **Diversitate în arme și atacuri:**
 - Introducerea mai multor opțiuni de atac pentru jucător.
12. **Salvarea progresului:**
 - Permite jucătorului să continue jocul ulterior.

Acest cod simplu poate fi extins în multe moduri pentru a crea un joc mai complex și captivant.

Explicația a fost realizată cu Chatgpt

```

explica indetaliat
import random

WIDTH = 10
HEIGHT = 10

MAP = [['.' for _ in range(WIDTH)] for _ in range(HEIGHT)]
player_position = [5, 5]
player_health = 100
player_symbol = '@'
monster_health = 20
monster_symbol = 'M'
monsters = []

def spawn_monsters(num_monsters):
    for _ in range(num_monsters):
        monster_pos = [random.randint(0, HEIGHT-1), random.randint(0,
WIDTH-1)]
        monsters.append({'position': monster_pos, 'health':

```

prerequisites & run

1. Se instaleaza VsCode de pe siteul oficial

i.

([Do](#)
[wnl](#)
[oad](#)
[Vis](#)
[ual](#)
[Stu](#)
[dio](#)
[Co](#)
[de -](#)
[Ma](#)
[c.](#)

2. După instalare se deschide VsCode
3. Descarcăm Python (de preferat o versiune stabilă)
4. În VsCode se alege ca interpretor versiunea de Python instalată (de aceea trebuia bifată opțiunea Path de la punctul 3.)
5. (O metodă foarte simplă este:) Deschid programul cu VsCode->ctrl+shift+p-> Python: Select Interpreter->Selectez versiunea de Python dorită
6. Se deschide cu VsCode
7. Se scrie programul
8. Se salvează programul (ca fișiere .py)
9. Se rulează Codurile

Explicații

Acest script implementează un joc simplu într-un spațiu bidimensional de 10x10, în care un jucător interacționează cu monștri. Vom detalia funcțiile și componentele programului:

1. Definiții și inițializări:

- **Variabile globale:**
 - WIDTH și HEIGHT: Dimensiunile hărții (10x10).
 - MAP: Reprezintă harta, inițializată cu puncte (.) pentru a simboliza spații libere.
 - player_position: Poziția inițială a jucătorului pe hartă (5, 5).
 - player_health: Sănătatea inițială a jucătorului (100).
 - player_symbol: Simbolul care reprezintă jucătorul pe hartă (@).
 - monster_health: Sănătatea inițială a unui monstru (20).
 - monster_symbol: Simbolul care reprezintă un monstru (M).
 - monsters: Listă care va conține monștrii, fiecare reprezentat ca un dicționar cu poziția și sănătatea.

2. Funcții principale:

a. `spawn_monsters(num_monsters)`

Această funcție generează un număr specificat de monștri pe harta 10x10:

- Utilizează `random.randint` pentru a genera poziții aleatorii.
- Fiecare monstru este reprezentat printr-un dicționar cu două chei:
 - `position`: Poziția pe hartă (ex.: `[y, x]`).
 - `health`: Sănătatea inițială (20).

b. `draw_map()`

Desenează harta în consolă:

- Iterează prin toate pozițiile (linii și coloane) din hartă.
- Verifică:
 - Dacă poziția curentă este a jucătorului (`player_position`).
 - Dacă există un monstru pe poziția curentă.
 - Dacă nu e nimic, afișează un punct (`.`).

c. `move_player(direction)`

Permite jucătorului să se deplaseze în cele patru direcții:

- `w`: Sus.
- `s`: Jos.
- `a`: Stânga.
- `d`: Dreapta.
- Verifică limitele hărții, pentru a nu ieși din aceasta.

d. `fight(monster)`

Inițiază o luptă între jucător și un monstru aflat pe aceeași poziție:

- Jucătorul atacă:
 - Daunele produse sunt aleatorii între 5 și 10.
 - Dacă sănătatea monstrului scade sub 0, lupta se încheie.
- Monstrul atacă:
 - Daunele produse sunt aleatorii între 1 și 5.
 - Dacă sănătatea jucătorului scade sub 0, jocul se termină.
- Afișează în mod constant sănătatea ambelor părți.

e. `check_win()`

Verifică dacă toți monștrii au fost eliminați:

- Dacă `monsters` este gol (toți monștrii au fost învinși), afișează un mesaj de câștig.

f. game_loop()

Conține bucla principală a jocului:

- Generează 3 monștri folosind spawn_monsters.
 - Afișează harta și sănătatea jucătorului.
 - Permite:
 - Mișcarea jucătorului (w, a, s, d).
 - Inițierea unei lupte (p), doar dacă un monstru este pe aceeași poziție.
 - Actualizează lista de monștri după lupte (elimină monștrii cu sănătate ≤ 0).
 - Verifică dacă jucătorul câștigă sau moare, oprind bucla corespunzător.
-

3. Fluxul jocului:

13. Harta este afișată.
 14. Jucătorul introduce o comandă:
 - Se poate deplasa.
 - Poate lupta dacă există un monstru pe poziția sa.
 15. Se actualizează starea jocului:
 - Poziția jucătorului.
 - Sănătatea jucătorului și a monștrilor.
 16. Jocul se termină dacă:
 - Toți monștrii au fost eliminați.
 - Jucătorul moare.
-

Îmbunătățiri posibile:

13. **Harta interactivă:**
 - Adăugarea de obstacole sau alte elemente pentru a diversifica jocul.
14. **Inteligența artificială pentru monștri:**
 - Monștrii să se deplaseze spre jucător.
15. **Diversitate în arme și atacuri:**
 - Introducerea mai multor opțiuni de atac pentru jucător.
16. **Salvarea progresului:**
 - Permite jucătorului să continue jocul ulterior.

Acest cod simplu poate fi extins în multe moduri pentru a crea un joc mai complex și captivant.

Explicația a fost realizată cu Chatgpt

```
explica indetaliat
import random
```

```
WIDTH = 10
HEIGHT = 10
```

```
MAP = [['.' for _ in range(WIDTH)] for _ in range(HEIGHT)]
player_position = [5, 5]
player_health = 100
player_symbol = '@'
monster_health = 20
monster_symbol = 'M'
monsters = []

def spawn_monsters(num_monsters):
    for _ in range(num_monsters):
        monster_pos = [random.randint(0, HEIGHT-1), random.randint(0,
WIDTH-1)]
        monsters.append({'position': monster_pos, 'health':
```

BIBLIOGRAFIE

Este buna sintaxa in python MAP=[WIDTH][HEIGHT]?

ChatGPT said:

ChatGPT

Nu, sintaxa `MAP = [WIDTH][HEIGHT]` nu este corectă în Python. Dacă vrei să creezi o matrice 2D sau o listă de liste în Python, ar trebui să folosești o abordare diferită.

Pentru a crea o matrice de dimensiuni `WIDTH x HEIGHT`, poți folosi una dintre următoarele metode:

1. Folosind list comprehensions:

python

Copy code

```
MAP = [[0 for _ in range(HEIGHT)] for _ in range(WIDTH)]
```


Aceasta va crea o matrice 2D, unde fiecare element este inițializat cu valoarea 0.

2. Folosind **numpy** (pentru performanță și manipulare ușoară a matricelor):

python

Copy code

```
import numpy as np
MAP = np.zeros((WIDTH, HEIGHT))
```

Aceasta creează o matrice de dimensiuni **WIDTH** x **HEIGHT**, cu valori inițializate la 0.

Explicație:

- **[WIDTH][HEIGHT]** este sintaxă incorectă, deoarece în Python pentru a accesa un element într-o listă de liste trebuie să folosești două indicii: **MAP[width][height]**.
- **[[0] * HEIGHT] * WIDTH** ar putea să fie folosit pentru a crea o listă de liste, dar această abordare poate avea probleme de referință (toate sublistele vor fi aceleași obiecte în memorie), așa că este recomandat să folosești metoda cu list comprehensions pentru a evita această problemă.

În concluzie, pentru a crea o matrice 2D, îți recomand să folosești list comprehensions sau **numpy** pentru eficiență.

4o mini

ce am putea folosi pentru a pune un numar random in python

ChatGPT said:

ChatGPT

Pentru a genera un număr aleatoriu în Python, poți folosi modulul integrat **random**. Acesta oferă funcții pentru generarea de numere aleatorii, atât întregi, cât și în virgulă mobilă.

Exemple:

Generarea unui număr întreg aleatoriu într-un interval:

python

Copy code

```
import random

numar_aleatoriu = random.randint(1, 100) # Un număr între 1 și 100 inclusiv
print(numar_aleatoriu)
```

1.

Generarea unui număr aleatoriu în virgulă mobilă:

python

Copy code

```
numar_aleatoriu = random.uniform(1.0, 10.0) # Un număr între 1.0 și 10.0  
print(numar_aleatoriu)
```

2.

Generarea unui număr între 0 și 1:

python

Copy code

```
numar_aleatoriu = random.random() # Valoare între 0.0 și 1.0  
print(numar_aleatoriu)
```

3.

Selectarea aleatorie dintr-o listă:

python

Copy code

```
culori = ["roșu", "verde", "albastru", "galben"]  
culoare_aleasa = random.choice(culori)  
print(culoare_aleasa)
```

4.

Generarea unui număr aleatoriu dintr-un interval cu pas personalizat:

python

Copy code

```
numar_aleatoriu = random.randrange(0, 100, 5) # Multiplu de 5 între 0 și 100  
print(numar_aleatoriu)
```

5.

Amestecarea elementelor unei liste:

python

Copy code

```
lista = [1, 2, 3, 4, 5]  
random.shuffle(lista)  
print(lista)
```

6.

Observații:

- Pentru control mai mare asupra generării numerelor aleatorii, poți folosi modulul **numpy**, care oferă funcții avansate pentru numere aleatorii.

Dacă ai nevoie de reproducerea aceleiași secvențe aleatorii (pentru debugging, de exemplu), poți seta o **sămânță (seed)**:

python

Copy code

```
random.seed(42) # Asigură reproducerea aceleiași secvențe de numere
```

-

4o

w3schools python

chatgpt a fost realizat

Acest proiect a fost inspirat din jocul original Rogue + linkul
([Rogue - The Original Roguelike - YouTube](#))