

Minesweeper

Introducere:

Minesweeper, cunoscut și sub denumirea de Mina, este un joc clasic de strategie pentru un singur jucător. El se bazează pe logica deductivă și pe ghicitul inteligent, permițând jucătorilor să-și exerseze abilitățile de raționament și să-și testeze strategiile.

Minesweeper este captivant și accesibil, potrivit pentru jucători de toate vârstele. Cu reguli simple, dar provocări complexe de rezolvat, jocul oferă o experiență distractivă și educativă pentru toți cei care doresc să își testeze gândirea strategică.

Mod de joc:

Scopul jocului constă în dezvăluirea tuturor câmpurilor fără mine dintr-o tablă, acest lucru ducând la victorie. Se evită, în același timp descoperirea oricărui câmp care ascunde o mină.

Jucătorii scriu coordonatele unei căsuțe (de la 0 la 4), pentru a dezvălui câmpul. Dacă în acel loc se află o mină, jocul se termină. În caz contrar, numărul afișat pe câmpurile dezvăluite indică câte mine se află în câmpurile adiacente.

Cum funcționează:

Jocul rulează pe două planuri, cel al clientului și cel al serverului. Componentele sunt conectate TCP. Serverul este deschis primul, fiind configurat să asculte pe un anumit port și generând un tabel cu bombe așezate pe poziții aleatorii. Clientul va trimite date pe acest port. Datele trimise (coordonatele unei căsuțe) vor fi procesate de server. Serverul verifică dacă coordonatele acelea corespund unei bombe în tabela generată. Dacă răspunsul este negativ, jocul continuă, iar serverul trimite clientului o altă tabelă cu căsuța respectivă dezvăluită. Serverul numără câte căsuțe au fost descoperite pentru a putea încheia jocul. O altă modalitate de a termina jocul este prin a pierde. Asta se poate întâmpla prin găsirea unei bombe: GAME OVER! La finalul jocului, și dacă ai pierdut și dacă ai câștigat, vei primi un mesaj de "la revedere": BYE!

Serverul se conectează la client:

```
anna@DESKTOP-ER7G26T:~/retele$ python3 mines_server.py
waiting connection...
Connected by ('127.0.0.1', 59240)
anna@DESKTOP-ER7G26T:~/retele$
```

Funcția care ajută la conectare este cea de start_server:

- Se creează socketurile și conexiunea TCP
- Se apelează funcția de începere a jocului

```
def start_server(host, port):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server.bind((host, port))
    server.listen(1)
    print("waiting connection...")
    conn, addr = server.accept()
    print('Connected by', addr)
    main_game(conn, server)

start_server("localhost", 9999)
```

Jocul începe:

- Se generează tabela cu bombele (nu este vizibilă pentru jucător)
- Tabela văzută de client este formată inițial doar din '_'
- Există o variabilă care va număra câte câmpuri au fost descoperite
- Jocul ia sfârșit când au fost descoperite toate câmpurile fără bombe

```
def main_game(conn, server):
    size = 5
    bomb_count = 8
    board = generate_board(size, bomb_count)
    client_board = [['_' for _ in range(size)] for _ in range(size)]
    checked = 0
    conn.send(str.encode(str(size)))
    while checked < size*size-bomb_count:
        x,y = read_client(conn)
        if board[x][y] != 'X':
            client_board[x][y] = board[x][y]
            write_msg(conn, client_board)
            checked += 1
        elif board[x][y]=='X':
            conn.send(str.encode("GAME OVER!"))
            break
    conn.send(str.encode("BYE!"))
    conn.close()
    server.close()
    exit(0)
```

Generarea tablei:

- Se generează aleatoriu coordonate pentru bombe
- Bombele vor fi notate cu 'X'
- După generarea bombelor urmează vecinii lor
- Fiecare câmp care nu conține o bombă va afișa numărul de bombe din jurul lui

```
def generate_board(size, num_mines):
    board = [['0' for _ in range(size)] for _ in range(size)]
    for _ in range(num_mines):
        x, y = random.randint(0, size - 1), random.randint(0, size - 1)
        board[x][y] = 'X'
    for i in range(size):
        for j in range(size):
            if board[i][j] != 'X':
                board[i][j] = count_neighbor(board, i, j)
    return board

def count_neighbor(board, x, y):
    directions = [(dx, dy) for dx in [-1, 0, 1]
                  for dy in [-1, 0, 1] if not dx == dy == 0]
    count = 0
    size = len(board)
    for dx, dy in directions:
        new_x, new_y = x + dx, y + dy
        if 0 <= new_x < size and 0 <= new_y < size and board[new_x][new_y] == 'X':
            count += 1
    return count
```

Funcția read_client:

- Procesează datele trimise de client
- Datele sunt reprezentate de 2 cifre separate de un spațiu

```
def read_client(conn):
    data = conn.recv(3)
    # print("data:", data)
    data = data.decode("utf-8")
    return int(data[0]), int(data[2])
```

Funcția write_msg:

- Trimite spre client tabela cu câmpurile descoperite de el
- Tabela e transformată în string și este codificată

```
def write_msg(conn, client_board):
    message = ''
    for row in client_board:
        row = map(lambda a: str(a), row)
        message += ''.join(row) + '\n'
    conn.send(str.encode(message))
```

Clientul:

- Trebuie apelat după server
- Se conectează la server
- Trimite serverului cele 2 coordonate care vor fi descoperite
- Primește de la server matricea cu pozițiile descoperite

```
<?php
$s=socket_create(AF_INET,SOCK_STREAM,0);
socket_connect($s,"127.0.0.1",9999);
socket_recv($s,$buf,20,0);
$size=intval($buf);
while($buf!="GAME OVER!BYE!")
{
    $send_me= readline();
    socket_send($s,$send_me,4,0);
    for($i=0;$i<$size+1;$i++)
    {
        socket_recv($s,$buf,$size,0);
        echo $buf;
    }
}
?>
```

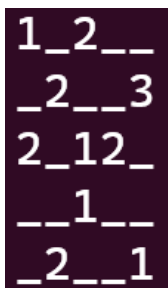
La rulare:

- Clientul trimite primele coordonate (de ex. 0 0) -fig. 1
- După 10 alte inputuri, tabela clientului va arăta astfel -fig. 2
(de ex. [4 4], [3 2], [4 1], [1 4], [2 3], [3 2], [0 2], [2 0], [1 1], [2 2])
- În caz că sunt introduse coordonatele unei bombe, jocul se sfârșește – fig. 3



```
0 0
1_ _ _ _
_ _ _ _ _
_ _ _ _ _
_ _ _ _ _
_ _ _ _ _
_ _ _ _ _
```

fig. 1



```
1_2_ _
_2_ _3
2_12_
_ _1_ _
_2_ _1
```

fig. 2



```
0 3
GAME OVER!BYE!
```

fig. 3