



Alternative 7 Schiffe versenken

BS&RN SoSe2022

Habib Mustafa Koca, Divya Kharbanda, Eyüp Tolgahan
Yildirim, Akash Mehra, Felix-Markus Koschitzky

Inhaltsverzeichnis

➤ Start

- Namenseingabe
Input
- Board Erstellung
- Zuweisung *Append*
- Platzierung *def placeship*
 - Platzierung (Zeile; Spalte; Horiz./Vert.)
 - Speicherung
 - Kontrolle (Überlappung)

➤ Spiel

- Angriff *def attack*
 - Platzierung (Koordinaten)
 - Kontrolle
 - Speicherung
- Spielstand
- clearscreen

➤ Ende

- „Game Over“

➤ Timer

- (Threads)

➤ Interprozesskommunikation

- (Probleme)

➤ Quellen



Start

- Spiel wird im Terminal gespielt
- Multiplayerspiel (2 Spieler)
- Spielfeldgröße Erlaubt: 8-12
 - Auswahl der Spielfeldgröße
- Namenseingabe durch **Input**
 - „Wie heißt Spieler1? Spieler1: “
 - „Wie heißt Spieler2? Spieler2: “
- Erstellung des Boards
 - Methode: **print_grid**:
- Zuweisung erfolgt durch den Befehl **.append**
 - *Name i (Spielername)*
 - *Board i (Board für die eigene Platzierung)*
 - *Guess i (Board für die Angriffe)*
 - *Gone i (Board für Kontrolle gegen Überlappung)*
 - *ships i (Liste mit einsetzbaren Schiffen)*

```
63 #Board in richtiger Formatierung ausgeben.
64 def print_grid(Board):
65     # Buchstaben Notation als Spalte
66     sys.stdout.write(" ")
67     for i in range(boardSize):
68         sys.stdout.write(" " + chr(i + 65))
69     print(" ")
70
71     # Schöner Darstellung
72     row_number = 1
73     for row in Board:
74         if row_number < 10:
75             grid = ("%d |%s|" % (row_number, "|".join(row)))
76         else:
77             grid = ("%d|s|" % (row_number, "|".join(row)))
78         print(grid)
79         row_number += 1
```

```
44 #Spieler Profil, alle Werte zuweisen.
45 Player1.append(Name1)
46 Player1.append(Board1)
47 Player1.append(Guess1)
48 Player1.append(Gone1)
49 Player1.append(ships)
50 Player2.append(Name2)
51 Player2.append(Board2)
52 Player2.append(Guess2)
53 Player2.append(Gone2)
54 Player2.append(ships)
```


Start

Platzierung der Schiffe

Funktion:

- *def place_ship:*

Übergabeparameter: *Board, Gone, Name*

- *Abfrage: Zeile; Spalte;
Horizontal/Vertikal*
 - Jeweils für „Schlachtschiff“;
„Kreuzer“; „Zerstörer“ und „U-
Boot“
- *Speicherung (Koordinaten): Liste:
„Gone“ (detailliertere Erklärung folgt)*
- Eigenes ersichtliches Board ist
„Board“
- *Kontrolle: über „Board3“*
 - Sowohl Kontrolle auf Überlappung
als auch auf Boardüberquerung
 - Jedes Schiff wird vor der
gewünschten Platzierung geprüft
 - Koordinaten werden gespeichert und
mit denen auf „Board“ abgeglichen



Spiel

Attackieren:

Funktion: *def attack*

Übergabeparameter: Board, Guess, Name, Gone

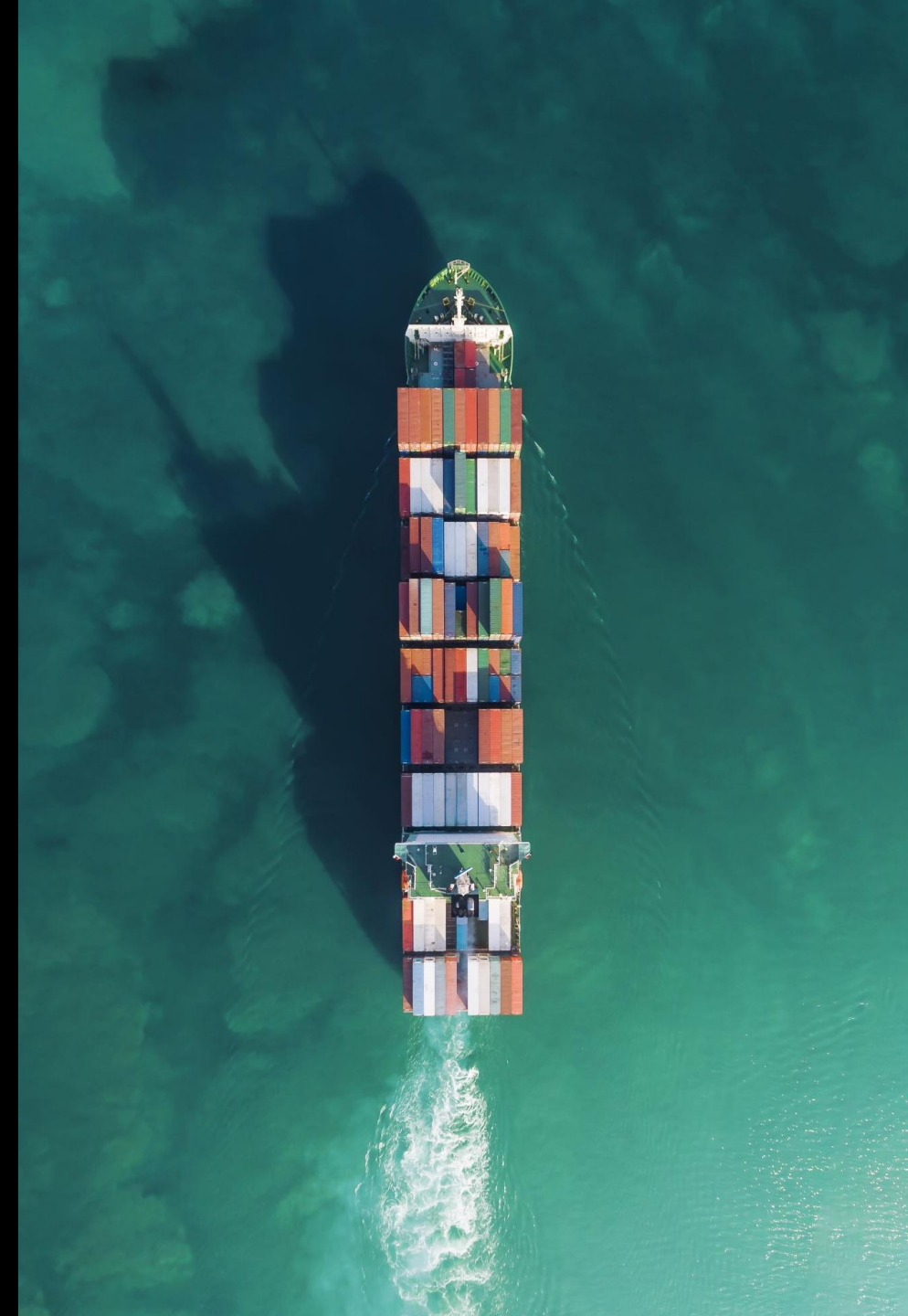
- Abfrage: Zeile/Spalte
 - Die ausgewählten Felder des Gegenspielers werden beschossen
 - Anzeige auf „Guess“
- "x" / "0" : Treffer oder Fehlschuss
- Schiff ist zerstört, wenn alle Schiffsteile getroffen wurden
 - Entsprechende Anzeige
- Kontrolle:
 - Ob eine Stelle bereits auf Schiffe geprüft wurde, mit Abgleich von „x“ und „0“
 - Wenn Ja: Exception



Spiel

Spielerwechsel:

- Funktion *def clearscreen*
 - Optisch gesäuberte Shell für Übergabe an Gegenspieler
 - Gewährleistung der Fairness
 - Benutzereingabe gesteuert
 - Enterklick



Spiel

Spielstand:

- Liste: „Gone“
 - Besteht aus 10 Unterlisten
 - Beinhaltet Koordinaten von Schiffsteilen
 - Jede Unterliste entspricht einem ganzen Schiff
- Ein Schiffsteil entspricht einem Tupel. Bsp.: (0, 1)

Restliche Schiffsteile:

`[(0, 1), (0, 2), (0, 3)],`

	A	B	C	D	E	F	G	H
1	k k k k							
2								
3								
4								
5								
6								
7								
8								

`[[(0, 0), (0, 1), (0, 2), (0, 3)]]`



7								
8								

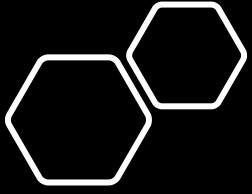
Welche Zeile willst du attackieren?
Zeile: 1
Welche Spalte willst du attackieren?
Spalte: A





Ende

- Treffer: Koordinaten-Tupel aus „Gone“ wird gelöscht
- Löschung einer vollständigen Unterliste in Gone
 - Schiff gesunken
 - Anzeige auf Shell
- Wenn die komplette Liste „Gone“ leer ist:
 - Game Over – Spiel zu Ende.



Timer

- Jeder Spieler hat 15 Sekunden Zeit
- Nach Ablauf des Timers, wird ein Zufallsschuss generiert
- Möglichkeiten mit threading
- Event wird gesetzt, um Timer zu Stoppen

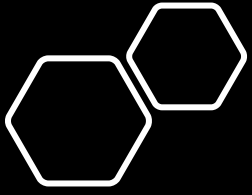
```
def schuss():  
    thread_1 = Thread(target=timer)  
    thread_1.start()
```

```
try:  
    #Eingabe passt  
    int(schuss)  
    exit_event.set()  
    print("Timer Stop")  
    break
```



```
def timer():
```

```
    if exit_event.is_set():  
        break
```




Zufallsschuss

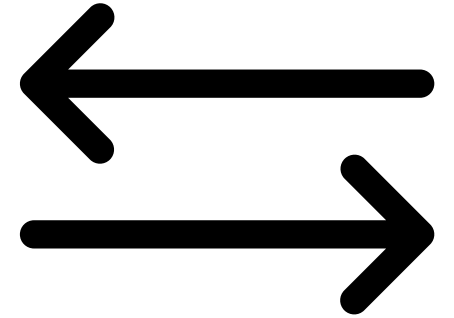
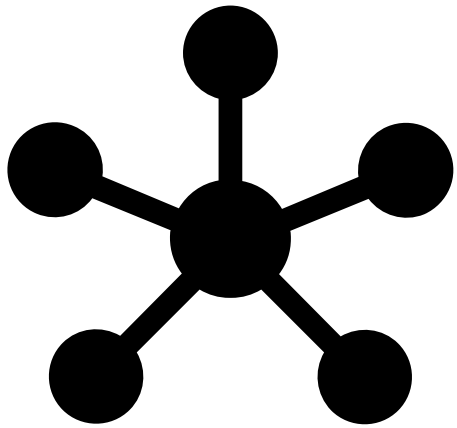
- `import random`
- Zwei Listen mit Variablen
- Berücksichtigung der Spielfeldgröße
- Zwei unabhängige Zufallsausgaben

```
def zufall():  
    list1=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']  
    list2=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]  
    b=random.randint(0, spielfeldgröße)  
    a=random.randint(0, spielfeldgröße)  
  
    zeile = list2[b]  
    spalte = list1[a]
```

```
--> 00:01 --> 00:00  
Ihre Zeit ist abgelaufen!  
Es wird auf die Koordinate: 2A geschossen
```



Ausgabe der
Zeile + Spalte



Interprozesskommunikation



Vielen Dank für Ihre Aufmerksamkeit

Habib Mustafa Koca, Divya Kharbanda, Eyüp Tolgahan Yildirim, Akash Mehra, Felix-Markus Koschitzky