BSRN-Alt. 3 Buzzword Bingo

Aouatif Adnane, Saida Covrk, Aisha Khan, Merwa Tahanur, Mariam Zadran

Gliederung

01

Umsetzung

Wie haben die Funktionen implementiert?

02

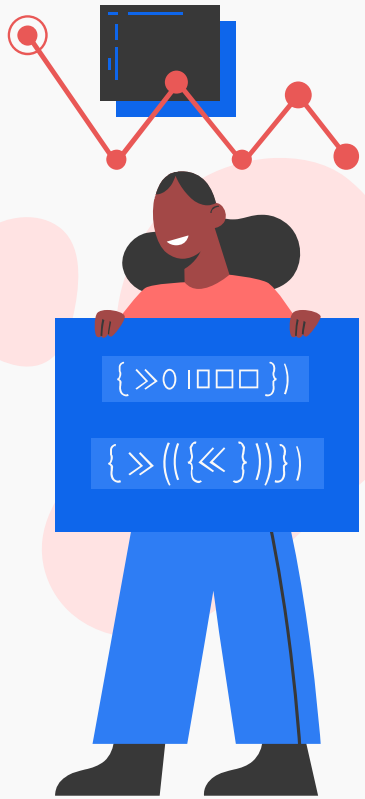
Probleme

Welche Hindernisse hatten wir?

03

Showcase

Eine Runde gemeinsam spielen.



Benutzeroberfläche

Interprozesskommunikation

Gewinnfunktion

Log-Funktion

Umsetzung

Probleme

Showcase

Vielen Dank für Ihre Aufmerksamkeit

Quellen

Benutzeroberfläche

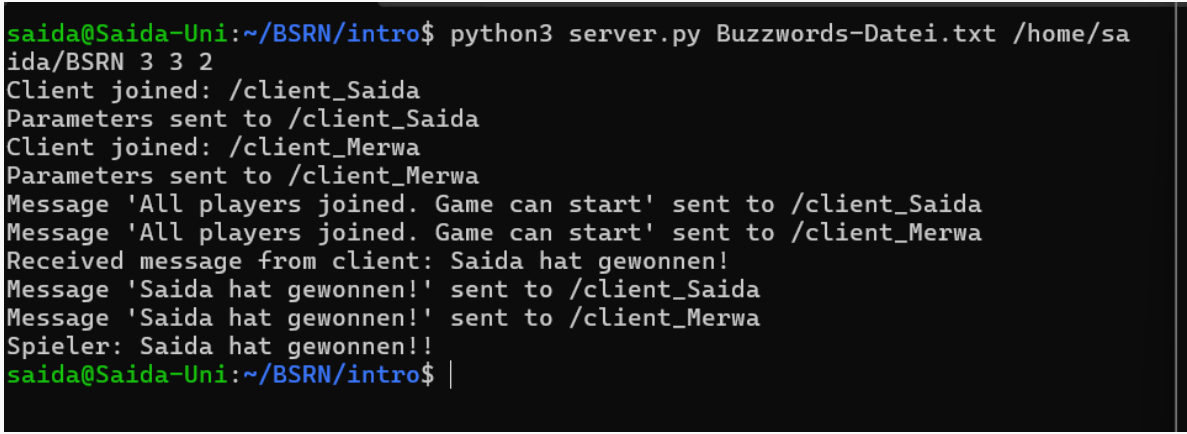
Benutzeroberfläche

Server wird gestartet → Angabe aller Parameter

- Buzzwords-Datei
- Logdatei Pfad
- Zeilen/Spalten
- maximale Spieleranzahl

Ziel:

- Separater Server für mehr Struktur



Benutzeroberfläche

Entwicklung:

- curses Bibliothek
- Buttons, Label

Ziel:

- Einfach und intuitiv für Spieler
- Intro mit Bedienhilfe

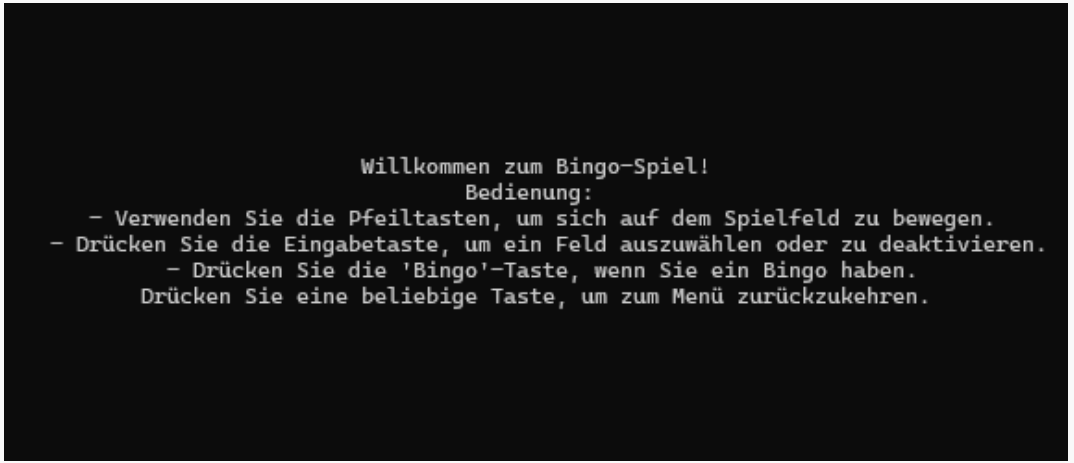
Benutzeroberfläche

Entwicklung:

- curses Bibliothek
- Buttons, Label

Ziel:

- Einfach und intuitiv für Spieler
- Intro mit Bedienhilfe



Benutzeroberfläche

Entwicklung:

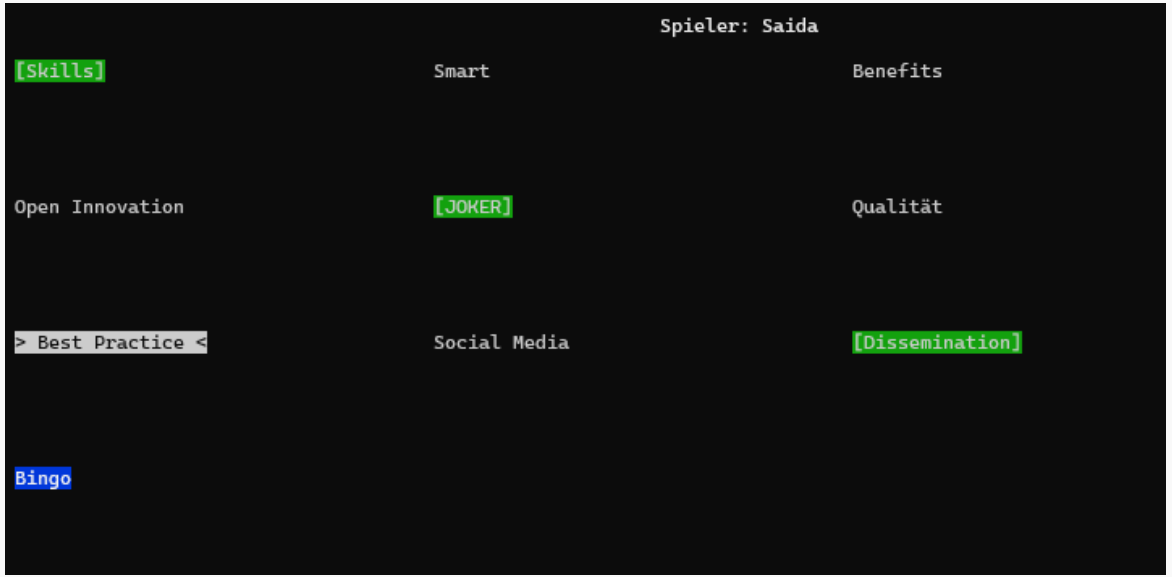
- curses Bibliothek
- Buttons, Label

Ziel:

- Einfach und intuitiv für Spieler
- Intro mit Bedienhilfe

Bingo-Feld:

- Grün: gedrückter Button
- Weiss: hover
- Joker



Interprozesskommunikation

Interprozesskommunikation (IPC)

- **Funktion**: "`handle_win_message(win_message)`"
- **Zweck**: Verteilung der Gewinnnachricht an alle Clients
- **Hauptaufgaben**:
  - Initialisieren und Entfernen der Server-Warteschlange
  - Senden der Nachricht an alle Clients
  - Fehlerbehandlung
  - Beenden des Spiels

```
95  def handle_win_message(win_message):
96      queue_server = posix_ipc.MessageQueue(QUEUE_SERVER)
97      for client_queue_name in clients:
98          try:
99              client_queue = posix_ipc.MessageQueue(client_queue_name)
100             client_queue.send(win_message.encode())
101         except posix_ipc.ExistentialError:
102             print(f"Error: Could not open client queue '{client_queue_name}'")
103     queue_server.close()
104     queue_server.unlink()
105     close_game()
```

Interprozesskommunikation (IPC)

- **Funktion**: "`send_game_params(client_queue_name, roundfile, log_path, zeilen, spalten, max_players)`"
- **Zweck**: Senden von Spielparametern an einen Client
- **Hauptaufgaben**:
  - Öffnen der Client-Nachrichtenwarteschlange
  - Senden der kodierten Spielparameter
  - Schließen der Warteschlange
  - Fehlerbehandlung bei nicht vorhandener Warteschlange

```python
def send_game_params(client_queue_name, roundfile, log_path, zeilen, spalten, max_players):
    try:
        client_queue = posix_ipc.MessageQueue(client_queue_name)
        parameters = f"{roundfile}|{log_path}|{zeilen}|{spalten}|{max_players}"
        client_queue.send(parameters.encode())
        client_queue.close()
        print(f"Parameters sent to {client_queue_name}")
    except posix_ipc.ExistentialError:
        print(f"Failed to open client queue '{client_queue_name}'")
```

Interprozesskommunikation (IPC)

- IPC mit **"POSIX Message Queues"**
- 1. Funktion: **"main player"**
- **Gründe für Verwendung:**
  - **Initialisierung**: Setzt den Spielername und ruft die Hauptlogik auf
  - **IPC**: Erstellt und sendet Nachrichten an die Server-Queue, um dem Server den Beitritt des Spielers mitzuteilen.

- Initialisiert Spielername.
- Erstellt Client-Queue und verbindet sich mit der Server-Queue.
- Sendet Beitrittsnachricht an den Server.
- Wartet auf Startnachricht oder Gewinnnachricht vom Server.
- Startet das Spiel über curses.wrapper(run_game_wrapper).

Interprozesskommunikation (IPC)

- IPC mit **"POSIX Message Queues**
- 2. Funktion: "**notify_all_clients (clients, message)"**
- **Gründe für Verwendung**:
  - **Broadcast:** Benachrichtigt alle Spieler über wichtige Ereignisse (z.B. Spielbeginn, Gewinner)
  - **IPC:** Nutzt Message Queues, um sicherzustellen, dass alle Clients zeitgleich informiert werden.

- Schleife über alle Client-Queues.
- Sendet Nachricht an jede Client-Queue.
- Schließt jede Client-Queue nach dem Senden.

Interprozesskommunikation (IPC)

- IPC mit **"POSIX Message Queues"**

- 3. Funktion: **"wait_for_player_join(queue_server, roundfile, log_path, zeilen, spalten, max_players"**
- **Gründe für Verwendung**:
  - **Synchronisation**: Stellt sicher, dass das Spiel erst beginnt, wenn alle Spieler beigetreten sind.
  - **Parameterübermittlung**: Sendet die Spielparameter an die Clients, sobald sie beitreten.
  - **IPC**: Nutzt Message Queues, um Nachrichten über Spielerbeitritte und Spielparameter zu empfangen und zu senden.

- Wartet auf Nachrichten in der Server-Queue.
- Verarbeitet Beitrittsnachrichten von Clients.
- Fügt Client-Queues zu einer Liste hinzu.
- Sendet Spielparameter an neue Clients.
- Benachrichtigt alle Clients, wenn alle Spieler beigetreten sind.
- Wartet auf Gewinnnachrichten von Clients und ruft handle_win_message auf.

Interprozesskommunikation (IPC)

- **Warum Message Queues?**
  - **Zuverlässigkeit**: Stellt sicher, dass Nachrichten zuverlässig zugestellt werden.
  - **Asynchronität**: Erlaubt asynchrone Kommunikation zwischen Server und Clients.
  - **Skalierbarkeit**: Unterstützt mehrere Clients und sorgt für synchronisierte Kommunikation.

Gewinnfunktion

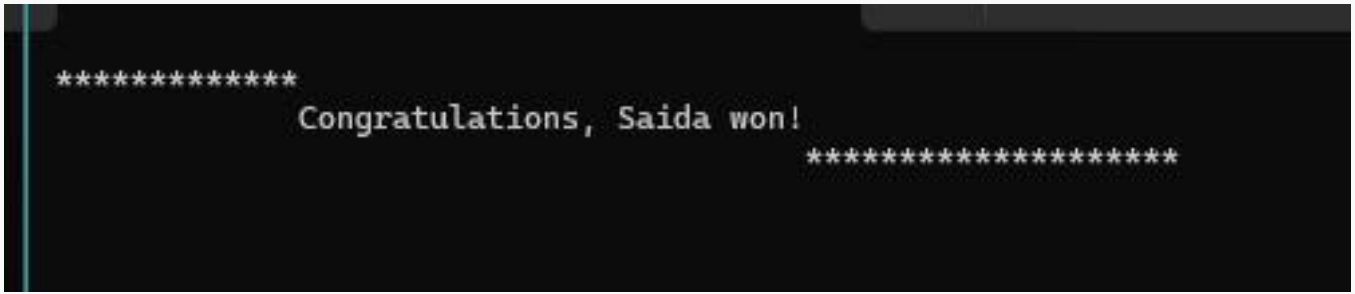
Gewinnkonzept

1. Gewinn bei Ausfüllen einer horizontalen, vertikalen oder diagonalen Reihe
2. Sichern mit Bingo Button
3. Bei Gewinn --> Gewinnnachricht
4. Gewinnanimation mit Spielernamen und Beglückung

```
if self.selected_button_index == len(self.buttons) - 1:  # BINGO button
    if self.check_for_win():
        self.broadcast_win()
        self.gewonnen_animation(self.player_name)
        return
```

Gewinnmeldung

- IPC durch Posix Ipc Message Queues
- Listener_thread um Gewinnnachrichten zu checken
- Gewinnmeldung durch broadcast_win
- Verarbeitung der Gewinnnachricht mit handle_win_message



Log-Funktion

Log-Datei

Aktivitäten eines Spielers werden protokolliert

- Name des Spielers
- Beginn/Ende eines Spiels
- Wer gewonnen hat
- Spielfeldgröße
- Zeile/Spalte
- Anklicken/Zurückklicken

Speicher in Verzeichnis "logs"

```
2024-06-29 19:44:25,020 - INFO - Log-Datei für Peter erstellt.
2024-06-29 19:44:25,020 - INFO - Größe des Spielfelds: Zeilen: 3, Spalten: 3
2024-06-29 19:44:25,020 - INFO - Start des Spiels.
2024-06-29 19:44:44,845 - INFO - Button geklickt: Rating (Zeile: 1, Spalte: 1)
2024-06-29 19:44:45,576 - INFO - Button geklickt: Synergie (Zeile: 2, Spalte: 1)
2024-06-29 19:44:46,604 - INFO - Button geklickt: Zielführend (Zeile: 2, Spalte: 3)
2024-06-29 19:44:47,924 - INFO - Button geklickt: Revolution (Zeile: 3, Spalte: 3)
2024-06-29 19:44:49,111 - INFO - Button geklickt: Geforwarded (Zeile: 3, Spalte: 2)
2024-06-29 19:44:51,573 - INFO - Peter hat gewonnen!
```

Datetime: Verwendung von Zeitstempeln in Protokollereignissen

Umsetzung

Probleme

Übergabe der Parameter im Client

Parameter:

- Buzzwords-Datei
- Log-Pfad
- Zeilen
- Spalten
- Maximale Spieleranzahl

Lösung →

client.py

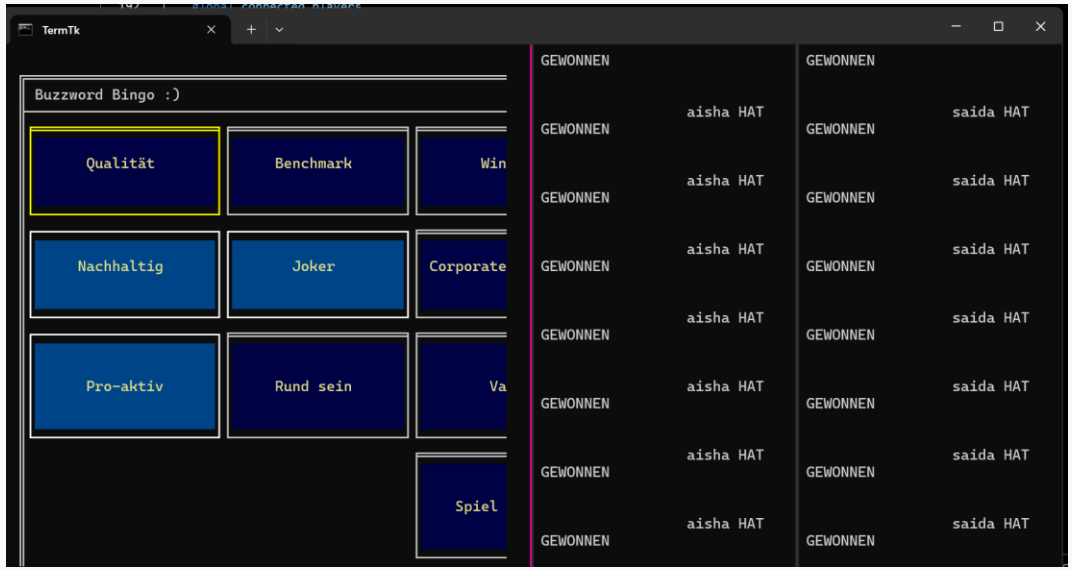
```python
while True:
            if content == "All players joined. Game can start":
                print("Game can start.")
                time.sleep(3)

                def run_game_wrapper(stdscr):
                    if curses.LINES < 20 or curses.COLS < 80:
                        print("Terminal window is too small. Please resize to at least 80x20.")
                        return
                    run_game(stdscr, player_name, zeilen, spalten, roundfile, log_path)

                result = curses.wrapper(intro_menu)
                if result == "start":
                    curses.wrapper(run_game_wrapper)
                elif result == "exit":
                    cleanup()
                    sys.exit(0)
                break
            else:
                params_list = content.split('|')
                if len(params_list) == 5:
                    roundfile = params_list[0]
                    log_path = params_list[1]
                    zeilen = int(params_list[2])
                    spalten = int(params_list[3])
                    max_players = int(params_list[4])
                    print(f"Received parameters from server:")
                    print(f"Roundfile: {roundfile}")
                    print(f"Log Path: {log_path}")
                    print(f"Zeilen: {zeilen}")
                    print(f"Spalten: {spalten}")
```

Gewinnnachricht Problem

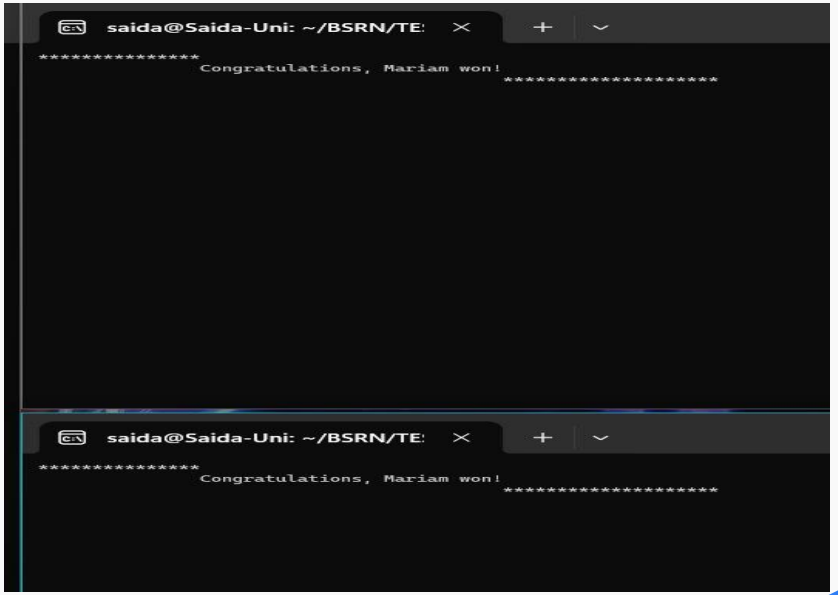
- Jeder Spieler hatte mit pyTermTk die Möglichkeit zu gewinnen
- Gewinnbediengung wurde für jeden Spieler separat geprüft
- Benannte Pipes haben Nachrichten nicht versendet



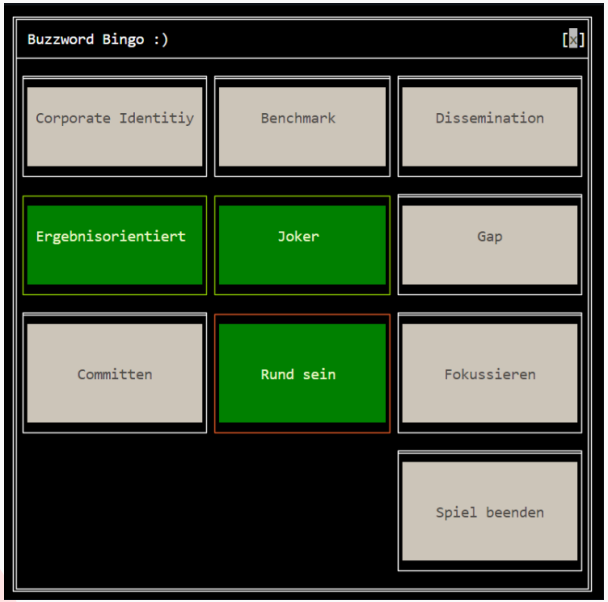
Gewinnnachricht Lösung:

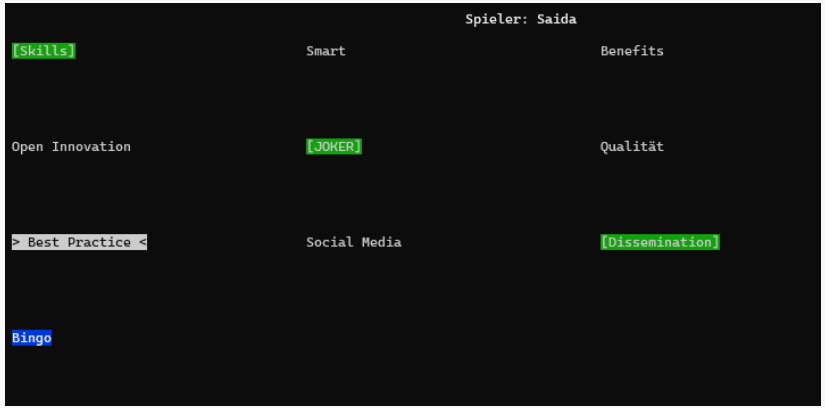
Lösung:

- Wechsel zur curses-Bibliothek
- Nutzung von POSIX-IPC Nachrichtenwarteschlangen
- Einrichten von Threads
- Fehlerbehebungen



Visualiserung





Doppeltes Logging in der Log-Datei

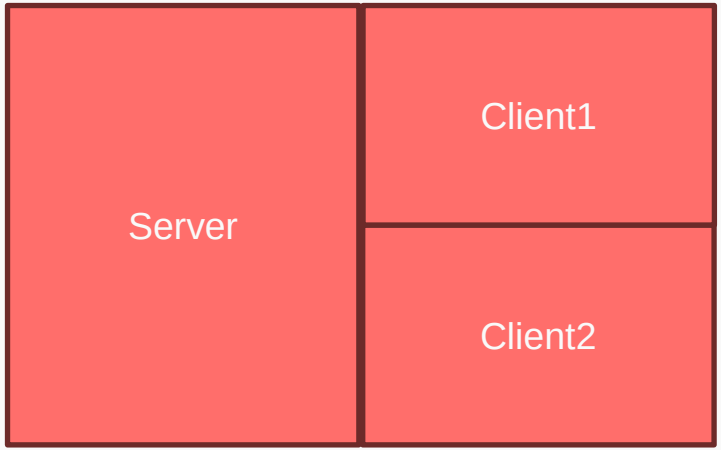
```
2024-06-29 22:03:39,459 - INFO - Log-Datei für mop erstellt.
2024-06-29 22:03:39,459 - INFO - Größe des Spielfelds: Zeilen: 3, Spalten: 3
2024-06-29 22:03:39,459 - INFO - Start des Spiels.
2024-06-29 22:03:43,087 - INFO - lop hat gewonnen!
2024-06-29 22:03:43,087 - INFO - lop hat gewonnen!
2024-06-29 22:04:00,490 - INFO - Ende des Spiels
```

Lösung → Umstrukturierung der *broadcast_win* und *gewonnen_animation* Methode

```
2024-06-30 21:09:39,744 - INFO - Log-Datei für mop erstellt.
2024-06-30 21:09:39,744 - INFO - Größe des Spielfelds: Zeilen: 3, Spalten: 3
2024-06-30 21:09:39,744 - INFO - Start des Spiels.
2024-06-30 21:09:42,623 - INFO - lop hat gewonnen!
2024-06-30 21:09:59,980 - INFO - Ende des Spiels
```

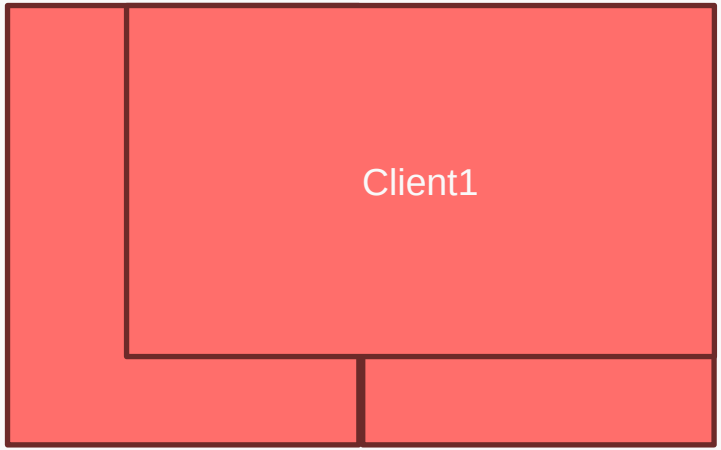
Resize

Problem: Bei einem vergrößern/verkleinern des Terminals wird der Prozess beendet. → Lösung: resize handler, funktioniert nicht wie gedacht



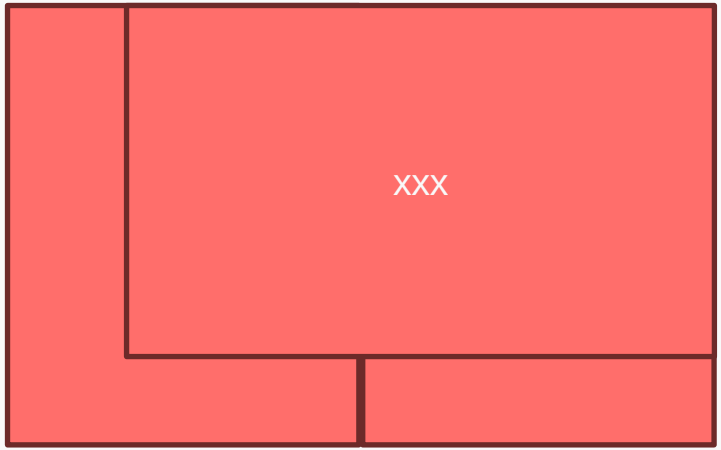
Resize

Problem: Bei einem vergrößern/verkleinern des Terminals wird der Prozess beendet. → Lösung: resize handler, funktioniert nicht wie gedacht



Resize

Problem: Bei einem vergrößern/verkleinern des Terminals wird der Prozess beendet. → Lösung: resize handler, funktioniert nicht wie gedacht



Showcase

Vielen Dank für Ihre Aufmerksamkeit

Quellen

Quellen

Slides Vorlage und damit Grafiken:

- https://slidesgo.com/theme/brackets-lesson-for-coding-and-programming#search-code&position-3&results-33&rs=search

GitHub:

- https://github.com/a-Bit-Of-Saida/BSRN-Gruppenaufgabe

Curses-Bibliothek in Python:

- https://github.com/zephyrproject-rtos/windows-curses