

Portfolio – Werkstück A – Alternative 3

Schiffe Versenken

1. Einführung

1.1 Zusammenfassung

Dieses Dokument enthält im Wesentlichen den Lösungsprozess des Schiffe Versenken-Projekts. Die strukturierte Beschreibung der Spielaufbau und Struktur der Entwicklung aller wichtigen und notwendigen Funktionen für den Code, als auch wie diese richtig implementiert sind.

Sowie die Beschreibung aller Probleme, die während des Prozesses aufgetreten sind, und der Maßnahmen, die zu deren Lösung ergriffen wurden. Ein weiterer Punkt, der im Dokument enthalten ist, ist die Bewertung des Ergebnisses, sowie Verbesserungen.

1.2 Einleitung

Die gestellte Aufgabe bestand darin, ein Spiel des Schiffe-Versenken mit Einzelspieler- und Mehrspielermodus mit „grafischer Darstellung“ in der Shell zu realisieren.

Von Anfang an gab es einige Probleme und Herausforderungen, denen man sich stellen musste. Die Tatsache, ein Spiel zu programmieren, das gleichzeitig grafisch dargestellt werden musste, bedeutete von Anfang an, dass mehrere Quellen untersuchen, Fehler machen, diese reflektieren und überwinden mussten, um ein funktionierendes Spiel zu erstellen.

Es ist bereits bekannt, dass Python eine Sprache mit weniger Schlüsselwörtern im Vergleich zu anderen Programmiersprachen ist. Dadurch wird sie kürzer und möglicherweise für Dritte leichter lesbar. Es kann als minimalistisch und intuitiv angesehen werden, was an die Thema Interessierten die Möglichkeit gibt, schneller im Verständnis voranzukommen. Daher zeigt das Dokument den Denkprozess zur Unterstützung dieser Aussage.

Das Ziel dieses Dokuments ist es also, den Fortschritt der erworbenen Kenntnisse bezüglich Python, der grafischen Darstellung des Spiels an der Shell und der anderen wichtigen Komponenten dieses Projekts zu zeigen.

2. Körper

2.1 Aufbau des Spieles

Das erste, was man benötigt und eine Voraussetzung für das Projekt ist, ist die Definition der Größe des Boards.

Das einzige, was man braucht, ist in der Lage zu sein, den Bereich (oder die Min und Max) oder die X- und Y-Achse zu generieren. Das Abrufen des Achsenbereichs aus einem Plot-Diagramm ist innerhalb der Python-Implementierung nicht möglich. Wie man in Abbildung 1 sieht, ein *ArgumentParser* mit Informationen über die Argumente des Programms durch Aufrufe der Methode `add_argument()` gefüllt. Im Allgemeinen sagen diese Aufrufe dem *ArgumentParser*, wie er Zeichenketten aus der Befehlszeile nehmen und in Objekte umwandeln soll.¹ Also, wenn der Benutzer eine Zahl weniger als 5x5 oder größer als 10x10 eingibt, interpretiert der Code die Zahl automatisch als 5x5 bzw. 10x10

```
parser = argparse.ArgumentParser(description='Battleship Spiel') #Initialisieren Sie den Argument-Parser
parser.add_argument('xaxis', type=int, help='Anzahl der Spalten im Spielplan ( 5 < xaxis <=10)') # xaxis argument
parser.add_argument('yaxis', type=int, help='Anzahl der Zeilen im Spielplan( 5 < yaxis <=10)') # yaxis argument
args = parser.parse_args()
```

Abbildung 1: Argumente um Spielboardgröße zu

Das Spiel hat zwei Spielmodi nämlich „Einzelspieler“ und „Mehrspieler“. Um eine davon auszuwählen, wird beim Öffnen des Spiels ein Menü angezeigt, das die verfügbaren Optionen anzeigt. Zusätzlich zum Einzelspieler- und Mehrspielersmodus wurde eine „Exit“-Option hinzugefügt, die das Programm schließt, wenn sie ausgewählt wird. Dies hat man mit einer Liste gemacht und der Benutzer kann mit der Tastatur eine der drei Optionen wählen. Um zu bestimmen, welcher der drei Alternativen vom Benutzer gewählt hat, verwendet man eine Funktion. Diese enthält if-Schleifen, die überprüfen, welches der Elemente in der Liste ausgewählt wurde (0, 1 oder 2).

Wählt man eine der ersten beiden Optionen, zeigt das Programm das Spielfeld und positioniert die Schiffe zufällig. Diese Positionierung kann aber auch von dem Benutzer geändert werden.

Um die Schiffe zufällig zu positionieren, haben wir das Modul *random* verwendet, wie Sie in Abbildung 2 sehen können. Dieses *random*-Modul bietet Zugriff auf Funktionen, die viele Operationen unterstützen. Am wichtigsten ist, dass es Zufallszahlen generieren kann.²

In diesem Fall willt man, dass der Computer eine Zufallszahl in einem vorgegebenen Bereich auswählt.

```
def schiffe_platzieren(self, gewinnen):
    gewinnen.clear() #Geben Sie die Möglichkeit, die Schiffe manuell zu platzieren oder mit der Standardpositionierung fortzufahren.
    h, w = gewinnen.getmaxyx()
    gewinnen.addstr(0, 0, self.name + " ! Drücken Sie Enter, um mit dieser Platzierung fortzufahren.")
    gewinnen.addstr(1, 0, self.name + " ! Oder drücken Sie " + "Leertaste" + " um mit der manuellen Platzierung von Schiffen zu beginnen.")
    for type in SCHIFF_TYPEN:
        self.platz_ship_random(type)
    gewinnen.refresh()
```

Abbildung 2: Random für die aleatorisches Platzierung

¹ Python Software Foundation. (2021). *argparse - Parser for command-line options, arguments and sub-commands*.
argparse - Parser for command-line options, arguments and sub-commands - Python 3.9.5 documentation.
<https://docs.python.org/3/library/argparse.html#argparse.ArgumentParser>.

² Python Software Foundation. (2021). *random - Generate pseudo-random numbers*. random - Generate pseudo-random numbers - Python 3.9.5 documentation. <https://docs.python.org/3/library/random.html>.

Zur Programmierung der Schiffe erstellt man eine Klasse („Ship“), diese enthält die relevanten Schiffsinformationen, wie z. B. den Schiffstyp, seine Größe, Ausrichtung, Koordinaten und „Hits“. Außerdem programmiert man eine Liste („Shiff_Typen“), die die verschiedenen verfügbaren Schiffstypen und ihre Menge enthält.

Dies ist besonders wichtig für die Positionierung. Wenn sie zufällig ist, verwendet das Programm eine Funktion, die eine if- und while-Schleife enthält. Diese überprüfen ob die zufällig generierten Schiffe der Liste „Shiff_Typen“ in das Spielfeld passen. Dies erreicht man, indem die Größe der Schiffe (in der Klasse gefunden) mit der Anzahl der verfügbaren Zeilen bzw. Spalten verglichen wird. Wenn es nicht passt, wird der Vorgang wiederholt.

Wenn der Benutzer die Positionierung seiner Schiffe ändern möchte, zeigt das Programm das leere Spielfeld an und ermöglicht es dem Spieler, mit der Maus oder der Tastatur das Feld auszuwählen, in dem er das Schiff und dessen Richtung platzieren möchte. Um das Schiff zu positionieren, muss der Benutzer die Tasten „v“ (vertikal) bzw. „h“ (horizontal) drücken. Wenn das Programm diese Information erhält, zählt es die Zellen nach unten bzw. rechts (mit Schleifen), um zu prüfen, ob das Boot in dieser Stelle passt. Auch hier verwendet man die Information der Klasse „Ship“ und die Liste „Shiff_Typen“.

Man verwendet Schleifen, um festzustellen, ob es bereits einen Gewinner gibt. Zuerst gibt es eine while-Schleife, die berechnet, ob Modulo 2 der Anzahl der Züge 0 ist (denn das bedeutet, dass die Anzahl der Züge gerade ist, d.h. beide Benutzer haben bereits ihren Zug gemacht). Wenn dies der Fall ist, prüft das Spiel, ob es bereits einen Gewinner gibt. Dafür werden if-Schleifen verwendet. Diese überprüfen die Gewinnbedingungen um festzustellen ob es schon ein Sieg oder ein Unentschieden gibt.

```
225 while True:
226     # wenn die Anzahl der Züge gleich ist, bedeutet dies, dass beide Spieler ihren Zug gespielt haben
227     if turns % 2 == 0:
228         # Prüfung für Game Over
229         if aktuelles_spieler.hat_verloren() and gegner.hat_verloren():
230             unentschieden(aktuelles_spieler, gegner, gewinnen)
231         elif aktuelles_spieler.hat_verloren():
232             game_over(gewinner=gegner, loser=aktuelles_spieler, gewinnen=gewinnen)
233         elif gegner.hat_verloren():
234             game_over(gewinner=aktuelles_spieler, loser=gegner, gewinnen=gewinnen)
```

2.1.1 Einzelspielermodus

In diesem Spielmodus tritt der Spieler gegen das Programm an. Das Programm muss also eine andere Person simulieren und in den jeweiligen Runden Spielzüge erzeugen.

Dazu verwendet man eine Funktion („cpu_bewegung“). Diese benutzt Schleifen und den Random Modul von Python, um zufällige Werte, die sich im Spielfeld befinden, zu erzeugen. Das heißt, dass die Zahle kleiner oder gleich als die x- bzw. y-Achse sein sollen. Wenn dies der Fall ist, gibt das Programm zwei Zahlen aus, die die beiden Koordinaten entsprechen. Dann prüft man, ob sich ein Schiff an den vom Programm gewählten Koordinaten befindet. Dies geschieht über eine weitere if-Schleife, die die zuvor angegebenen Daten überprüft. Wenn das Programm eines der Schiffe trifft, werden die Nachbarzellen des getroffenen Schiffes im nächsten Zug ausgewählt.

```
def cpu_bewegung(self, gegner):
    for y in range(ZEILEN): #CPU-Bewegung definieren
        for x in range(SPALTEN):
            if gegner.board[y][x] == CODE_HIT:
                cell = Koordinate(x, y)
                neighbors = [cell.left(), cell.right(), cell.up(), cell.down()]
                random_cell = random.choice(neighbors)
                if random_cell.x < 0 or random_cell.x >= SPALTEN or random_cell.y < 0 or random_cell.y >= ZEILEN:
                    continue
                return random_cell
    randomy = secrets.randbelow(ZEILEN)
    randomx = secrets.randbelow(SPALTEN)
    return Koordinate(randomx, randomy)
```

Abbildung 3: CPU Spielvorgehen

Wenn der Benutzer dran ist, zeigt das Programm das leere Spielfeld an und ermöglicht es dem Nutzer, eine Zelle auszuwählen. Hier wird eine weitere Funktion verwendet, um die Benutzereingabe zu verarbeiten. Diese hat eine if-Schleife, die prüft, ob diese Koordinate bereits ausgewählt wurde, ob das Schiff bereits versenkt wurde oder ob es leer ist. Wenn keines davon zutrifft, bedeutet dies, dass ein Schiff getroffen wurde. Wenn dies geschieht, wird 1 zur Anzahl der Treffer addiert, die das Schiff erhalten hat, und es wird überprüft, ob es mit diesem Treffer gesunken wurde. Dies wird erreicht, indem die Anzahl der Treffer mit der Größe des Schiffes verglichen wird (diese Information befindet sich in der Klasse „Ship“).

2.1.2 Multispielermodus

In diesem Spielmodus spielen zwei verschiedene Benutzer gegeneinander, jeder von denen hat sein eigenes Spielfeld, und dieses ist vor dem Gegner verborgen. Die auf dem Bildschirm dargestellten Informationen wechseln also je nach Spielrunde. Um die Spielzüge zu trennen, zeigt man auf dem Bildschirm eine Meldung an, die darauf hinweist, dass der nächste Spieler dran ist, und die erst verschwindet, wenn man eine Taste drückt. Diese Meldung programmiert man in die Schleifen, in denen das Spiel prüft, ob bereits zwei Runden vergangen sind und ob es bereits einen Gewinner gibt. Dadurch wird sichergestellt, dass sie nur bei Bedarf angezeigt wird.

Der allgemeine Verlauf dieses Spielmodus ist der des Einzelspielers sehr ähnlich.

Die erste Positionierung von Schiffen funktioniert auf die gleiche Weise, aber der Auswahlprozess wird für einen zweiten Benutzer wiederholt.

```
#Schiffe für Spieler 1 platzieren
spieler1.schiffe_platzieren(gewinnen)
gewinnen.clear()
h, w = gewinnen.getmaxyx()
msg = spieler1.name + " ! Ihre Schiffe wurden platziert. Drücken Sie eine beliebige Taste, um fortzufahren."
gewinnen.addstr(h//2, w//2 - len(msg)//2, msg)
gewinnen.refresh()

gewinnen.getch()

# Schiffe für Spieler 2 platzieren
spieler2.schiffe_platzieren(gewinnen)
gewinnen.clear()
h, w = gewinnen.getmaxyx()
msg = spieler2.name + " ! Ihre Schiffe sind platziert worden. Drücken Sie eine beliebige Taste, um fortzufahren."
gewinnen.addstr(h//2, w//2 - len(msg)//2, msg)
gewinnen.refresh()
```

Abbildung 4: Platzierung von Schiffen

Das Gleiche gilt für jede der Spielrunden, da das Programm nur die Option hinzufügen muss, dass ein zweiter Benutzer Daten eingeben kann. Zu diesem Zweck man verwendet die Klasse „*Spieler*“, in der die relevanten Informationen jedes Spielers gespeichert werden, die es ermöglichen, die einzelnen Spieler voneinander zu unterscheiden und die von ihnen ausgeführten Züge zu speichern. Daher wurden auch Funktionen und Variablen hinzugefügt, die denjenigen des ersten Players entsprechen, aber andere Werte speichern.

2.1.3 Bugs

Zu erwähnen sind einige Bugs, die im Laufe der Arbeit aufgetreten sind, z.B. die Anzeige des Spielergebnisses. Vor der Behebung des Fehlers wurde die Meldung "Game Over, Spieler 1 gewinnt" auf eine korrekte Koordinate gesetzt, so dass die Meldung perfekt sichtbar war, wenn das Spiel endet und die beiden Boards erscheinen. Bei der Auswahl eines größeren Boards befand sich die Nachricht jedoch unter den Boards. Dies löst man durch, dass man die Meldung mit Hilfe eines *addstrs* viel weiter nach rechts verschiebt, so dass sie bei größeren Boards sichtbar sind. Wenn Sie jedoch zu kleineren Boards zurückkehren, erscheint die Meldung etwas weit entfernt von den Boards, was die Visualität des Spiels beeinträchtigt.

Es gibt auch andere Bugs im Spiel, die nicht behoben werden konnten, aber nur gelegentlich auftreten, wie z.B. das Einfrieren des Bildschirms im Hauptmenü, im vs. CPU-Modus und bei der Auswahl der Positionierung der Schiffe.

2.2 Graphische Darstellung

Die grafische Darstellung ist eine Herausforderung und eine sehr wichtige Anforderung für das Projekt. Es wurde beschlossen, die *Curses*-Bibliothek für die Implementierung von Farben, die Verwendung von Mause für die Auswahl von Koordinaten auf dem Board, Bildschirmflackern, wenn ein Spieler gewinnt, unter anderem, zu benutzen.

Zunächst sollte aber kurz und bündig erklärt werden, was *curses* ist. Die *curses*-Bibliothek liefert eine terminalunabhängige Bildschirmdarstellung und Tastaturbedienung für textbasierte Terminals.

Der Inhalt eines Fensters kann auf verschiedene Arten verändert werden - Text hinzufügen, löschen, Aussehen ändern - und die *curses*-Bibliothek findet heraus, welche Steuercodes an das Terminal gesendet werden müssen, um die richtige Ausgabe zu erzeugen.³

³ Python Software Foundation. (2021.). *Curses Programming with Python*¶. Curses Programming with Python - Python 3.9.5 documentation. <https://docs.python.org/3/howto/curses.html>.

```
def main(stdscr):  
    curses.curs_set(0) #den Cursor ausblenden  
  
    # alle in meinem Spiel verwendeten Farbschemata  
    curses.init_pair(1, curses.COLOR_BLACK, curses.COLOR_WHITE)  
    curses.init_pair(2, curses.COLOR_YELLOW, curses.COLOR_BLACK)  
    curses.init_pair(3, curses.COLOR_GREEN, curses.COLOR_BLACK)  
    curses.init_pair(4, curses.COLOR_RED, curses.COLOR_BLACK)  
    curses.init_pair(5, curses.COLOR_WHITE, curses.COLOR_YELLOW)  
    curses.init_pair(6, curses.COLOR_WHITE, curses.COLOR_RED)  
    curses.init_pair(7, curses.COLOR_MAGENTA, curses.COLOR_CYAN)  
  
    stdscr.border(0) # Rand des Standardbildschirms  
    curses.mousemask(1) # um das Abhören auf Mausereignisse zu aktivieren  
  
    main_menu(stdscr, 0)
```

Abbildung 5: Hier kann man die Verwendung von verschiedenen Funktionen im Zusammenhang mit Curses in der main-Methode des Codes sehen.

Offensichtlich muss man *curses* mit

einer Funktion initialisieren, in diesem Fall mit der Funktion *stdscr*. Diese Funktion sendet die notwendigen Konfigurationscodes an das Terminal (nachdem sie den Terminaltyp bereits erkannt hat) und erstellt einige interne Strukturdaten.

Schritt für Schritt ist zunächst zu erwähnen, dass zur Verbesserung der Sichtbarkeit und der Spielbarkeit beschlossen wurde, die Terminal-Cursor (Mauszeiger) auszublenden.

Die Funktion *curses.curs_set()* setzt den Sichtbarkeitsstatus des Cursors, zwischen unsichtbar, normal oder sehr sichtbar. Hier kann man die Verwendung von verschiedenen Funktionen im Zusammenhang mit Curses in der main-Methode des Codes sehen.

Wie man umsetzt, ist in Zeile 49 des Codes im obigen Bild zu sehen, wodurch die Sichtbarkeit des Cursors entfällt (*curses.curs_set(0)*).

Nachdem man die Bibliothek initialisiert hat, sollt man etwas Farbe ins Spiel bringen. Wie man im Bild oben sehen kann, kann man mit der Funktion *curses.init_pair* alle Farbschemata definieren, die man im Spiel verwendet

Diese Funktion, nimmt drei Argumente entgegen: die Nummer des zu ändernden Farbpaars, die Nummer der Vordergrundfarbe und die Nummer der Hintergrundfarbe. Das ändert die Definition des Farbpaars.⁴ Mit dem obigen Codebild wäre ein Beispiel, um die Farbe 2 in gelben Text auf schwarzem Hintergrund zu ändern, es würde aufgerufen werden:
curses.init_pair(2, curses.COLOR_YELLOW, curses.COLOR_BLACK)

Ein weiteres Problem war die Definition die Kanten und wie man es so gestalten kann, dass die Spieler mit der Maus auf einen bestimmten Punkt schießen können.

Für die Randdefinition haben wir zunächst *border()* verwendet, eine Funktion, die einen Rand um die Kanten des Terminals zeichnet. In diesem Fall hat man die Zahl 0 verwendet, um die Standardwerte in *curses.h* zu verwenden⁵

⁴ Python Software Foundation. (2021). *curses - Terminal handling for character-cell displays*. curses - Terminal handling for character-cell displays - Python 3.9.5 documentation. <https://docs.python.org/3/library/curses.html#module-curses>.

⁵ Oracle. (2014, November 11). *curs_border - man pages section 3: Curses Library Functions*. https://docs.oracle.com/cd/E36784_01/html/E36880/curs-border-3curses.html#REFMAN3Ckurs-border-3curses.

Wie bereits erwähnt, war eine Anforderung des Projekts, dass die Spieler mit den Pfeiltasten, der Eingabe der genauen Koordinaten oder mit der Maus schießen können. Dafür wurden verschiedene Methoden und Funktionen verwendet. Die nächsten Schritte zur Lösung dieses Problems werden später in diesem Dokument besprochen.

```
def zelle_zeichnen(coord, offx, offy, code, ausgewähltes, gewinnen):  
    #Grafische Darstellung des Boards mit verschiedenen Farben unter Verwendung von curses  
    if ausgewähltes:  
        gewinnen.attron(curses.color_pair(7))  
        gewinnen.addstr(coord.y*ZEILE_HÖHE + offy, coord.x*SPALTE_BREITE + 1 + offx, str(code))  
        gewinnen.attroff(curses.color_pair(7))  
    else:  
        if code == CODE_HIT or code == CODE_verseut:  
            gewinnen.attron(curses.color_pair(6))  
            gewinnen.addstr(coord.y*ZEILE_HÖHE + offy, coord.x*SPALTE_BREITE + 1 + offx, str(code))  
            gewinnen.attroff(curses.color_pair(6))  
  
            elif code == CODE_MISS:  
                gewinnen.attron(curses.color_pair(1))  
                gewinnen.addstr(coord.y*ZEILE_HÖHE + offy, coord.x*SPALTE_BREITE + 1 + offx, str(code))  
                gewinnen.attroff(curses.color_pair(1))  
  
            elif code == CODE_LEER:  
                gewinnen.addstr(coord.y*ZEILE_HÖHE + offy, coord.x*SPALTE_BREITE + 1 + offx, str(code))  
  
        else:  
            gewinnen.attron(curses.color_pair(5))  
            gewinnen.addstr(coord.y*ZEILE_HÖHE + offy, coord.x*SPALTE_BREITE + 1 + offx, str(code))  
            gewinnen.attroff(curses.color_pair(5))
```

Abbildung 6: :
Wichtige Methode
zur grafischen
Darstellung. Enthält
verschiedene
Funktionen

In Abbildung 6
sieht man, wie
man die
Farbdefinition
(über die man
bereits in den
vorherigen
Abschnitten

gesprochen hat) und die tatsächliche Implementierung im Spiel verbinden. Hierfür haben wir die Funktion `curses.color.pair()` verwendet, nach den Funktionen `getch()`, `attron`, `addstr` und `attroff`.

Etwas Wichtiges, das man bei der Programmierung des Spiels zu berücksichtigen sollte, ist die Änderung der Farben, wenn man einen Schuss trifft oder nicht, oder wenn man ein Schiff versenkt hat. Alle diese Änderungen sollten unabhängig sein und sich nicht auf die anderen Farben auf der Board auswirken.

Als letztes brauchten wir `getch()`, damit man ein Zeichen aus dem zum Fenster gehörenden Terminal liest, also es aktualisiert den Bildschirm und wartet es, bis man eine Taste drückt. In unserem Fall wäre das jedes Mal der Fall, wenn der Spieler eine Zelle zum Schießen drückt.

Man verwendet diese Funktionen, um die Attribute des Bildschirms zu manipulieren. Im Grunde ein Ein- und Ausschalter für die Farben. Die Funktion `attron` schaltet die genannten Attribute ein, ohne die anderen zu beeinflussen, und die Funktion `attroff` schaltet sie aus.⁶

Diese Funktionen zeigt man zusammen mit `color_pair` an und ändern die Zelle in die angegebene Farbe, je nachdem, ob es sich um einen Treffer oder einen Fehlschuss handelt. Wenn es sich z. B. um einen Treffer handelt, wechselt die Farbe auf das Farbpaar Nummer 6, das (wie bereits in der `main` Methode definiert) weiß mit rotem Hintergrund ist.

⁶ Oracle. (2014, November 11). `curs_attr` - man pages section 3: Curses Library Functions.
https://docs.oracle.com/ed/E36784_01/html/E36880/curs-attr-3curses.html#REFMAN3Curs-attr-3curses.


```
if key == curses.KEY_DOWN:
    ausgewählte_zelle.y += 1
elif key == curses.KEY_UP:
    ausgewählte_zelle.y -= 1
elif key == curses.KEY_LEFT:
    ausgewählte_zelle.x -= 1
elif key == curses.KEY_RIGHT:
    ausgewählte_zelle.x += 1
```

Abbildung 8: Tastatureingabe lesen

In Abbildung 7 sieht man, wie man das Problem des Auslesens von Daten aus der Tastatur löst. Zuerst muss man die Funktion `getch()` aktivieren, es liest ein Zeichen aus dem mit dem Fenster verbundenen

Terminal. Es aktualisiert dann das Bildschirm und wartet dann darauf, dass der Benutzer eine Taste drückt. Bei Verwendung der `if`-Schleife können Sie dann die Tasten und die Folgen des Tastendrucks zuweisen. Dies zusammen mit der Verwendung von `x` und `y`-achse kann man mit den Pfeiltasten durch das Spielfeld scrollen.⁷

```
# Mauseingabe
elif key == curses.KEY_MOUSE:

    click = curses.getmouse()
```

Abbildung 7: Mauseingaben einlesen

Um die Anforderung zu lösen, dass der Benutzer die Zellen mit der Maus auswählen kann. Man verwendet die Funktion `get_mouse`, die nach der Initialisierung der Funktion `getch` ein Signal eines Mausereignisses zurückgibt.

3. Schluss

3.1 Bewertung des Ergebnisses und Verbesserungen

Die Wahrung von Stil und Konsistenz ist der Schlüssel zur Entwicklung von qualitativ hochwertigem und lesbarem Python-Code. die manuelle Durchsetzung eines Code ist ein echtes Problem und kostet wertvolle Entwicklungszeit.

Das manuelle Erzwingen des Code birgt jedoch eine Reihe von Herausforderungen. Erstens ist es zeitaufwändig. (Es braucht Zeit, um sich mit den Regeln des Codestils/Programmiersprache vertraut zu machen)

Zweitens leidet die Codequalität, wenn man der Code manuell erzwingt. Die Kollegen müssen unvermeidliche Fehler im Code beachten und manuell korrigieren. Dies geht zu Lasten der Konzentration auf Probleme mit und Verbesserungen des Codes selbst. Ein Tool, das beim Schreiben des Codes hilft, wäre sinnvoll gewesen.

Drittens in Bezug auf dem Spiel, man hätte die Auswahl der Größe des Spielbretts schon im Programm selbst implementieren können und nicht erst im Terminal. Auch die Einstellung des Textes "Game Over Spieler 1 gewinnt"

⁷ Oracle. (2014, November 11). `curs_getch` - man pages section 3: Curses Library Functions. https://docs.oracle.com/cd/E36784_01/html/E36880/curs-getch-3curses.html#REFMAN3Ccurs-getch-3curses.

sollte mehr auf die Größe der Spielboards reagieren und nicht statisch auf einer bestimmten Koordinate stehen. Sowie die Behebung der oben bereits erwähnten Bugs.

Und schließlich hätte die Koordination des Teams besser sein können, da die Tatsache, dass drei Personen denselben Code programmieren, manchmal zu Verwirrung führt, z. B. bei der Definition von Variablen, der Lokalisierung und Anwendung von Methoden, Objekten usw.. Abgesehen davon, dass nicht jeder den gleichen Wissensstand in der Materie hat, entstehen dadurch auch Verzögerungen im Prozess.

Wahrscheinlich durch ein persönliches Treffen, um die Probleme zu besprechen und somit produktiver und zeitsparender zu sein.

4. Literatur

- Python Software Foundation. (2021). *argparse - Parser for command-line options, arguments and sub-commands*¶. *argparse - Parser for command-line options, arguments and sub-commands - Python 3.9.5 documentation.* <https://docs.python.org/3/library/argparse.html#argparse.ArgumentParser>.
- Python Software Foundation. (2021). *random - Generate pseudo-random numbers*¶. *random - Generate pseudo-random numbers - Python 3.9.5 documentation.* <https://docs.python.org/3/library/random.html>.
- Python Software Foundation. (2021.). *Curses Programming with Python*¶. *Curses Programming with Python - Python 3.9.5 documentation.* <https://docs.python.org/3/howto/curses.html>.
- . Python Software Foundation. (2021). *curses - Terminal handling for character-cell displays*¶. *curses - Terminal handling for character-cell displays - Python 3.9.5 documentation.* <https://docs.python.org/3/library/curses.html#module-curses>.
- Oracle. (2014, November 11). *curs_border - man pages section 3: Curses Library Functions.* https://docs.oracle.com/cd/E36784_01/html/E36880/curs-border-3curses.html#REFMAN3Ccurs-border-3curses.
- Oracle. (2014, November 11). *curs_attr - man pages section 3: Curses Library Functions.* https://docs.oracle.com/cd/E36784_01/html/E36880/curs-attr-3curses.html#REFMAN3Ccurs-attr-3curses.
- Oracle. (2014, November 11). *curs_getch - man pages section 3: Curses Library Functions.* https://docs.oracle.com/cd/E36784_01/html/E36880/curs-getch-3curses.html#REFMAN3Ccurs-getch-3curses.
- Refsnes Data. (2021). Python Tutorial. <https://www.w3schools.com/python/default.asp>.
- Python Software Foundation. (2021). *sys - System-specific parameters and functions*¶. *sys - System-specific parameters and functions - Python 3.9.5 documentation.* <https://docs.python.org/3/library/sys.html>.
- Programcreek. (n.d.). *Python curses.KEY_MOUSE Examples.* Python Examples of curses.KEY_MOUSE. https://www.programcreek.com/python/example/12875/curses.KEY_MOUSE.