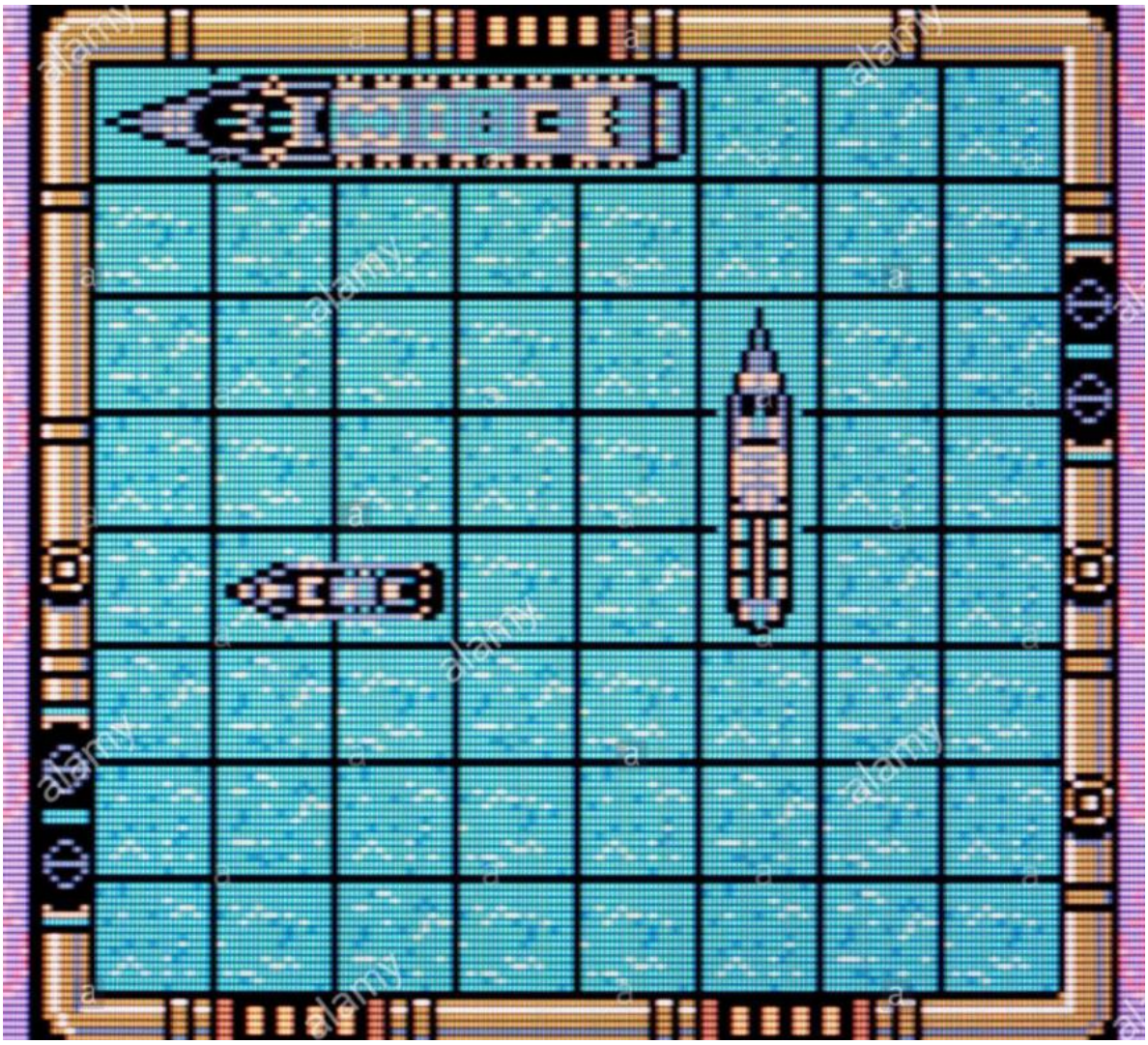
Projektstart: 14.04.2021

Projektende: 28.06.2021

Schiffe-Versenken



Zakaria Mowafek , Sam Acosta & Merve Ilpesin

Inhaltsverzeichnis:

1. Warum wir uns für dieses Projekt entschieden haben:

Wie jeder weiß, ist „Schiffe versenken“ ein Klassiker, den jeder aus seiner Kindheit kennt. Gerade deshalb war die Wahl dieses Projekts eindeutig für uns – ein nostalgisches Spiel wie dieses virtuell zu programmieren, schien für uns eine knifflige und zugleich interessante Herausforderung.

2. Code:

Wichtig:

Bitte folgende Python Packages installieren:

Für „random2“ : pip install random2

Für „os“: pip install os

Für „cfonts“: pip install python-cfonts

Für „colorama“: pip install colorama

Die weiteren importierten Python Packages sollten in Python bereits installiert sein.

```
1  import random2
2  import os
3  from cfonts import render, say
4  import time
5  import colorama
6  from colorama import Fore, Back, Style
7  colorama.init(autoreset=True)
```

2.1 Menü

```
def mainMenu() :
    ok = 1
    while ok == 1:
        try:
         pvp = ("Player vs Player")
         pve = ("Player vs Computer")
         ext = ("Exit")
         print("\n1." + pvp)
         print("2." + pve)
         print("3." + ext +"\n")
         eingabe = input("Zum Auswählen der Optionen bitte 1, 2 oder 3 drücken und mit Enter bestätigen.")
         int(eingabe)
         liste = [1,2,3]
         if "1" in eingabe:
             print("Eingabe erfolgreich - es wurde " + pvp + " ausgewählt.")
             ok == 2
             break
         elif "2" in eingabe:
             print("Eingabe erfolgreich - es wurde " + pve + " ausgewählt.")
             ok == 2
             break
```

```
        elif "3" in eingabe:
            print("\nEingabe erfolgreich - es wurde " + ext + " ausgewählt.")
            ok == 2
            break

        else:
            print("\nFehler - Bitte 1, 2 oder 3 auswählen.")

        except:
            print("\nFehler - Bitte 1, 2 oder 3 auswählen.")

    return eingabe


def finale(check):
    try:
        if "1" == check:
            #pvp
            bestaetigung = input("Ausführen? [j/n]:")
            if bestaetigung == "j":
                #hier kommt pvp rein
                print("Spiel würde jetzt ausgeführt werden.")
                bs = "n"

            elif bestaetigung == "n":
                bs = "n"

            elif bestaetigung != "j" or bestaetigung != "n":
                print("Bitte nur j für ja und n für nein bestätigen.\n")
                bs = "n"

        elif "2" == check:
            #pve
            bestaetigung = input("Ausführen? [j/n]:")
            if bestaetigung == "j":
                # hier kommt pve rein
                print("Spiel würde jetzt ausgeführt werden.")
                bs = "n"
```

```
            elif bestaetigung == "n":
                bs = "n"

            elif bestaetigung != "j" or bestaetigung != "n":
                print("Bitte nur j für ja und n für nein bestätigen.\n")
                bs = "n"

        elif "3" == check:
            #ext
            bestaetigung = input("Möchten Sie das Spiel wirklich beenden? [j/n]:")
            if bestaetigung == "j":
                print("Vielen Dank fürs Spielen.")
                bs = "j"

            elif bestaetigung == "n":
                bs = "n"

            elif bestaetigung != "j" or bestaetigung != "n":
                print("Bitte nur j für ja und n für nein bestätigen.\n")
                bs = "n"
        else:
            print("Bitte nur j für ja und n für nein bestätigen.\n")
    except:
        print("")
        print("Fehler - Eingabe überprüfen.")

    return bs


check = mainMenu()
bs = finale(check)
def aufseher(bs):
    merve = "n"
    while merve == "n":
        check = mainMenu()
        bs = finale(check)
        if bs == "j":
            merve = "y"
            break
        elif bs == "n":
            print("Wiederholung der Schleife.")
```

```
if bs != "j":
    aufseher(bs)
else:
    print("Vielen Dank fürs Spielen.")
```

Anfangs war es in der Überlegungsphase schwer sich gedanklich ein Menü vorzustellen, was sich jedoch später als eine nicht allzu schwere Herausforderung herausgestellt hat, da uns mit der Zeit das Einprogrammieren verschiedener Erweiterungen leichter fiel. Wir haben uns für eine große If-Anweisung entschieden, welche als Menü dient, unendlich geht und sich wiederholt, um dem Spieler einen reibungslosen Übergang zwischen den Modi zu simulieren. Dabei mussten wir uns auch Gedanken über einen „Exit“ machen, um das Spiel zu beenden.

Um das Menü besser darzustellen, haben wir uns dazu entschieden, es einzeln in der Dokumentation zu zeigen. Dies ist jedoch nur der Vorgänger zum eingebauten Menü, welches erweiterte Funktionen und letzten Endes auch die Spielmodi enthält.

2.2 Player vs Player:

Um Spieler 2 zu implementieren, haben wir den Spieler 1 modifiziert und leicht abgeändert. Deshalb zeigen wir hier nur Spieler 1, um die Dokumentation kompakter zu halten.

```
# Spieler1 Programmhälfte    Spieler1 Programmhälfte     Spieler1 Programmhälfte:

def pruefung_auf_fehler_Spieler1(auswahl_position_Spieler1, bereits_ausgewaehlt_Spieler1):
    auswahl_position_Spieler1.sort()
    for i in range(len(auswahl_position_Spieler1)):
        num = auswahl_position_Spieler1[i]
        if num in bereits_ausgewaehlt_Spieler1:
            auswahl_position_Spieler1 = [-1]
            break
        elif num < 0 or num > 99:
            auswahl_position_Spieler1 = [-1]
            break
        elif num % 10 == 9 and i < len(auswahl_position_Spieler1) - 1:
            if auswahl_position_Spieler1[i + 1] % 10 == 0:
                auswahl_position_Spieler1 = [-1]
                break
        if i != 0:
            if auswahl_position_Spieler1[i] != auswahl_position_Spieler1[i - 1] + 1 and \
```

```
                        auswahl_position_Spieler1[i] != auswahl_position_Spieler1[i - 1] + 10:
                    auswahl_position_Spieler1 = [-1]
                    break
        return auswahl_position_Spieler1

    # Diese Funktion kreiert für den Spieler 1 seine Spielfeld - ein Feld für die eigenen Schiffe.
    # Hierbei wurde mit Hilfe von If-Anweisungen darauf geachtet, dass es zu keiner doppelplatzierung
    # der Schiffe kommt oder gar zu einer falschen und nicht befindbaren Koordinate (z.B N8 oder Z2).

    def schiff_Spieler1_hinzufuegen(long, bereits_ausgewaehlt_Spieler1):

        ok = True
        while ok:
            try:
                print("          Schiffe Versenken    ")
                print("              Spieler 1     ")
                print("     Bitte plaziere deine Schiffe")
                print("     A  B  C  D  E  F  G  H  I  J")
```

```
                place1 = 0
                for x in range(10):
                    row1 = ""
                    for y in range(10):
                        cz = " _ "
                        row1 = row1 + cz
                        place1 = place1 + 1

                    print(x, " ", row1)
                schiff_Spieler1 = []
                # Fordert den Spieler auf eine Koordinate einzugeben.
                print("Standpunkt von Schiff mit Länge ", long)
                for i in range(long):
                    boat_num = input("Bitte Koordinate eingeben, wo sich dein Schiff befinden soll:")
                    if len(boat_num) != 2:
                        print("\nFehler - Koordinate befindet sich nicht auf dem Feld.")
                        print("...versuche stattdessen Bsp: A1, C8, G3\n")
                    else:
                        eins = boat_num[0:1]
                        # print(eins)
```

```
                        zwei = boat_num[-1:]
                        # print(zwei)

                        if eins == "a" or eins == "A":
                            abc = 0
                        elif eins == "b" or eins == "B":
                            abc = 1
                        elif eins == "c" or eins == "C":
                            abc = 2
                        elif eins == "d" or eins == "D":
                            abc = 3
                        elif eins == "e" or eins == "E":
                            abc = 4
                        elif eins == "f" or eins == "F":
                            abc = 5
                        elif eins == "g" or eins == "G":
                            abc = 6
                        elif eins == "h" or eins == "H":
                            abc = 7
                        elif eins == "i" or eins == "I":
                            abc = 8
```

```
                    elif eins == "j" or eins == "J":
                        abc = 9
                    else:
                        print("Falsche Koordinate! Bitte Koordinate innerhalb des Feldes wählen")

                    xyz = int(zwei)
                    xyz = (xyz * 10)
                    boat_num = abc + xyz

                schiff_Spieler1.append(int(boat_num))
                # Prüft auf das Schiff
            schiff_Spieler1 = pruefung_auf_fehler_Spieler1(schiff_Spieler1,
                                                           bereits_ausgewaehlt_Spieler1)
            if schiff_Spieler1[0] != -1:
                bereits_ausgewaehlt_Spieler1 = bereits_ausgewaehlt_Spieler1 + schiff_Spieler1
                print("...Schiff wurde plaziert")
                print()
                print()
                break
            else:
                print("Fehler - Platziertes Schiff überschneidet sich möglicherweise")
```

```
            except:
                print("        Bitte erneut eingeben")
                print()
        return schiff_Spieler1, bereits_ausgewaehlt_Spieler1

    def schiffe_bauen_Spieler1(bereits_ausgewaehlt_Spieler1, anzahl_schiffe_Spieler1):
        schiffe_Spieler1 = []
        # auswahl_schiffe_spieler1 = [5,4,3,3,2,2]

        for auswahl_schiff_Spieler1 in anzahl_schiffe_Spieler1:
            schiff_Spieler1, bereits_ausgewaehlt_Spieler1 = schiff_Spieler1_hinzufuegen(
                auswahl_schiff_Spieler1, bereits_ausgewaehlt_Spieler1)
            schiffe_Spieler1.append(schiff_Spieler1)

        return schiffe_Spieler1, bereits_ausgewaehlt_Spieler1

    # Die folgende Funktion zeigt dem Spieler 1 sein Spielfeld mit den platzierten Schiffen.

    def show_board_c_Spieler1(bereits_ausgewaehlt_Spieler1):
        print("               Spieler 1")
        print("Hier sind deine Schiffe in einer Übersicht")
         print("     A  B  C  D  E  F  G  H  I  J")

         place = 0
         for x in range(10):
             row = ""
             for y in range(10):
                 ch = " _ "
                 if place in bereits_ausgewaehlt_Spieler1:
                     ch = " o "
                 row = row + ch
                 place = place + 1

             print(x, " ", row)

     def show_board_Spieler1(hit_AufSpieler2, miss_AufSpieler2, complete_AufSpieler2):
         schiffe_uebrig = 6
         print("          Schiffe Versenken    ")
         print("               Spieler 1")
         print("     A  B  C  D  E  F  G  H  I  J")
```

```
            place = 0
            for x in range(10):
                row = ""
                for y in range(10):
                    ch = " _ "
                    if place in miss_AufSpieler2:
                        ch = " x "
                    elif place in hit_AufSpieler2:
                        ch = " o "
                    elif place in complete_AufSpieler2:
                        ch = " O "
                        schiffe_uebrig -= 1
                    row = row + ch
                    place = place + 1
                print(x, " ", row)
            print("Spieler 2 hat noch " + str(schiffe_uebrig) + " Schiffe übrig.\n")

        # Mit dieser Funktion (Methode) wird der Spieler 1 aufgefordet auf die Koordinaten,
        # auf die als erstes geschossen werden sollen.
        def schuss_VonSpieler1_AufSpieler2(zuvorGeschossenseSchuesse_VonSpieler1):
```

```
            ok = "n"
            while ok == "n":
                try:
                    print("Spieler 1 ist am Zug")
                    anfrage = input("Bitte gebe dein Ziel ein: ")
                    print("Befehl zum Angriff auf folgendes Ziel: " + ">" + anfrage + "<")

                    if len(anfrage) != 2:
                        print("\nFehler! Koordinate befindet sich nicht auf dem Feld.")
                        print("...versuchen Sie stattdessen Bsp: A1, C8, G3\n")
                    else:
                        print("Ziel anvisiert!")
                        eins = anfrage[0:1]
                        # print(eins)
                        zwei = anfrage[-1:]
                        # print(zwei)

                        if eins == "a" or eins == "A":
                            abc = 0
                        elif eins == "b" or eins == "B":
                            abc = 1
```

```
                        elif eins == "c" or eins == "C":
                            abc = 2
                        elif eins == "d" or eins == "D":
                            abc = 3
                        elif eins == "e" or eins == "E":
                            abc = 4
                        elif eins == "f" or eins == "F":
                            abc = 5
                        elif eins == "g" or eins == "G":
                            abc = 6
                        elif eins == "h" or eins == "H":
                            abc = 7
                        elif eins == "i" or eins == "I":
                            abc = 8
                        elif eins == "j" or eins == "J":
                            abc = 9
                        else:
                            print("Falsche Koordinate! Bitte Koordinate innerhalb des Feldes wählen")
```

```
                    xyz = int(zwei)
                    xyz = (xyz * 10)
                    schuss_VonSpieler1 = abc + xyz

                    if schuss_VonSpieler1 not in zuvorGeschossenseSchuesse_VonSpieler1:
                        # print(shot)
                        print("FEUERFREI!!!\n")
                        ok = "y"
                        break

                    else:
                        print("Ziel wurde bereits beschossen, bitte wähle ein anderes.")

                except:
                    print("Falsche Koordinaten eingegeben, bitte erneut versuchen")

            return schuss_VonSpieler1
        # Um das Spiel korrekt spielen zu können, wird mit dieser Funktion (Methode) darauf geprüft,
        # ob es zu einem Miss oder Hit kam.
        def pruefung_schuss_VonSpieler1(schuss_VonSpieler1, schiffe_spieler2, hit_AufSpieler2, miss_AufSpieler2,
                                        complete_AufSpieler2):
            missed_AufSpieler2 = 0
            for i in range(len(schiffe_spieler2)):
                if schuss_VonSpieler1 in schiffe_spieler2[i]:
                    schiffe_spieler2[i].remove(schuss_VonSpieler1)
                    if len(schiffe_spieler2[i]) > 0:
                        hit_AufSpieler2.append(schuss_VonSpieler1)
                        missed_AufSpieler2 = 1
                    else:
                        complete_AufSpieler2.append(schuss_VonSpieler1)
                        missed_AufSpieler2 = 2
            if missed_AufSpieler2 == 0:
                miss_AufSpieler2.append(schuss_VonSpieler1)

            return schiffe_Spieler2, hit_AufSpieler2, miss_AufSpieler2, complete_AufSpieler2

        # Spieler2 Programmhälfte    Spieler2 Programmhälfte     Spieler2 Programmhälfte:

        def pruefung_auf_fehler_Spieler2(auswahl_position_Spieler2, bereits_ausgewaehlt_Spieler2):
            auswahl_position_Spieler2.sort()
```

2.3 Ausführung:

```
        # Abteilung für die Auführung der Methoden - Reihenfolge beachten!
        # Im Anschluss des Codes erscheint ein Text, der den Spieler erwähnt der gewinnt.

        def check_if_empty_Spieler(list_of_lists):
            return all([not elem for elem in list_of_lists])

        hit_AufSpieler2 = []
        miss_AufSpieler2 = []
        complete_AufSpieler2 = []
        zuvorGeschosseneSchuesse_Spieler1 = []
# loop
for i in range(100):
    # guesses
    zuvorGeschosseneSchuesse_Spieler1 = hit_AufSpieler2 + miss_AufSpieler2 + complete_AufSpieler2
    # shot
    schuss_VonSpieler1 = schuss_VonSpieler1_AufSpieler2(zuvorGeschosseneSchuesse_Spieler1)
    # check_shot
    schiffe_Spieler2, hit_AufSpieler2, miss_AufSpieler2, complete_AufSpieler2 = pruefung_schuss_VonSpieler1(
        schuss_VonSpieler1, schiffe_Spieler2, hit_AufSpieler2, miss_AufSpieler2, complete_AufSpieler2)
    # show_board
    show_board_Spieler1(hit_AufSpieler2, miss_AufSpieler2, complete_AufSpieler2)
    # repeat until ships empty
    if check_if_empty_Spieler(schiffe_Spieler2):
        # print("end of game - winner in", i)
        clear = lambda: os.system('cls')  # Lässt die Überschrifft verschwinden
        clear()
        time.sleep(2)
            # clear = lambda: os.system('cls')  # Lässt die Überschrifft verschwinden
            clear()

            break
```

```
        # guesses Spieler2
        zuvorGeschosseneSchuesse_Spieler2 = hit_AufSpieler1 + miss_AufSpieler1 + complete_AufSpieler1
        # shot Spieler 2
        schuss_VonSpieler2 = schuss_VonSpieler2_AufSpieler1(zuvorGeschosseneSchuesse_Spieler2)
        # check_shot Spieler2
        schiffe_Spieler1, hit_AufSpieler1, miss_AufSpieler1, complete_AufSpieler1 = pruefung_schuss_VonSpieler2(
            schuss_VonSpieler2, schiffe_Spieler1, hit_AufSpieler1, miss_AufSpieler1, complete_AufSpieler1)
        # show_board Spieler2
        show_board_Spieler2(hit_AufSpieler1, miss_AufSpieler1, complete_AufSpieler1)
        # repeat until ships empty Spieler2
        if check_if_empty_Spieler(schiffe_Spieler1):
            # print("end of game - winner in", i)
            clear = lambda: os.system('cls')  # killt die ueberschrift
            clear()
            time.sleep(2)
              fourh = render('Spieler 2', font='block', colors=['red', 'white'],
                             align='center')
              third = render('SPIELER 2 - DU HAST GEWONNEN!', font='block', colors=['blue', 'white'],
                             align='center')
              print(third)
              print("                IN", i, "ZÜGEN GEWONNEN")
              time.sleep(9)
              # clear = lambda: os.system('cls')  # killt die ueberschrift
              clear()
              break
    print("fertig")
```

Im Folgenden werde ich die Funktionsweise des Spiels in kurzen Sätzen zusammenfassen:

Zunächst muss der Spieler seine Schiffe platzieren, dies geschieht in schiffe_bauen_Spieler1.

Die vom User eingegebenen Koordinaten werden gespeichert und es wird das Spielbrett angezeigt, worauf sich alle seine platzierten Schiffe befinden.

(Das gleiche passiert für Spieler 2)

In einer sich ständig wiederholenden Schleife werden von den Spielern die zuvor geschossenen Schüsse gespeichert, die eingegebenen Schüsse überprüft und gespeichert, und das getroffene Feld wird auf dem Spielfeld angezeigt.

Nach diesen Schritten wird vom Spiel überprüft, ob sich noch Zahlen in der Liste von den Schiffen der jeweiligen Spieler befinden. Wenn eine dieser Listen leer ist (z.B. von Spieler 1), wird dies erkannt und als Gewinner Spieler 2 angezeigt. Der Modus wird beendet und das Menü erscheint erneut.

3. Schwierigkeiten:

Tatsächlich fiel es uns am Anfang nicht einfach einen roten Faden durch das Projekt zu ziehen, da alle Gruppenmitglieder ihre Ideen einbringen wollten und verschiedene Meinungen zu konstruktiven Diskussionen führten. Vor eine gemeinsame Herausforderung gestellt zu werden, stärkte unseren Teamgeist und unser Horizont wurde erweitert. Zudem haben wir gelernt unsere Stärken in die Erstellung des Konzepts miteinfließen zu lassen und zusammen an einer Lösung zu arbeiten.

Diese Projektarbeit hat uns demnach gezeigt, wie Projekte im späteren Leben nach der Universität ablaufen können und hat uns somit auf unser zukünftiges Arbeitsleben vorbereitet.

4. Welche Hindernisse wir beim Programmieren hatten:

Wir haben den Umgang mit einer uns bis dato fremden Software-Plattform und vor allem einer neuen Programmiersprache (Python) erlernt. Zunächst haben uns in PyCharm die Formatierung und viele Fehlermeldungen allgemein Schwierigkeiten bereitet, da diese Punkte nicht wie in Eclipse leicht zu lösen waren, doch durch Übung haben wir diese Software-Umgebung besser kennengelernt.

Ebenfalls war es nicht einfach dieses Spiel graphisch in der Shell darzustellen. Im ersten Semester war unser Fachgebiet eher das objekt-orientierte Programmieren, wir hatten keinerlei Erfahrung mit größeren Anforderungen im Programmieren, gar mit Spielen.

Womit wir hauptsächlich zu kämpfen hatten war der Player- vs. Computer-Teil. Beispielsweise fiel es uns schwer einen Algorithmus für das gezielte Schießen einzubauen.

Der Algorithmus musste taktisch denken und bei einem Treffer die umliegenden Felder dementsprechend auch beschießen. Der Player- vs. Player-Teil baut auf den vorherigen Teil auf, nur, wurde hier der Algorithmus für das Erstellen der Schiffe entfernt und das Befeuern der gegnerischen Schiffe durch die Eingabe eines Inputs des Users modifiziert. Ebenfalls musste die Reihenfolge angepasst werden, sodass die Spieler abwechselnd schießen und ein Spieler erst seine ganzen Schiffe platziert, damit der andere Spieler in der Zeit weggucken kann und andersrum, da der Computer selbstverständlich nicht hinschaut im Vergleich zu diesem Modus.

Eine weitere Herausforderung war es verschachtelte Schleifen programmieren, wie z.B. beim Aufstellen der mehrdimensionalen Spielfläche (10 mal 10 Feld). Markierungen für Schiffe, die mit dem letzten Schuss versenkt wurden, einzubauen und anzeigen zu lassen, dass das die letzte Zahl in der Lise ist, fiel uns ebenfalls schwer.

Bei einem Spiel ist es außerdem wichtig Sicherheitsrahmen einzubauen. Um das zu bewerkstelligen, mussten wir If-, Elif-, und Else-Anweisungen einbauen, um falsche Eingaben herauszufiltern und das Spiel benutzerfreundlicher zu machen. Beispielsweise darf nicht außerhalb des Spielfelds geschossen werden und das Programm kontrolliert dies kontinuierlich. Somit dienen diese Anweisungen als Filter, sodass am Schluss das Spiel seinen Spielspaß behält ohne Fehlermeldungen. (Bei der Angabe der Koordinaten muss der Buchstabe vorne stehen, die Zahl dahinter und sie darf NUR zweistellig sein!). Groß- und Kleinschreibung spielt keine Rolle, es wurde so programmiert, dass beides geht, um dem Spieler ein angenehmeres Spielen zu ermöglichen.

Unser Basiskonstrukt musste Stück für Stück erweitert werden, um den Anforderungen zu entsprechen. In diesem Prozess kamen dann immer wieder neue Ideen und dementsprechend Methoden und Anweisungen hinzu, die mit der Zeit implementiert haben, bis wir mit dem Ergebnis zufrieden waren.

Das ständige Durchlaufen lassen des Spieles, um Fehler zu finden, hat uns natürlich Nerven gekostet 😊, da wir das Spiel ständig spielen mussten, um den Grund der Fehler ausfindig zu machen. Was uns aber gegen Ende Spaß gemacht hat, war der Fakt, dass sich dieses Spiel nach all der Arbeit wie ein echtes Spiel hat spielen lassen und nur noch geringfügige Kleinigkeiten verändert werden mussten.

Was wir aus der gemeinsamen Projektarbeit gelernt haben:

Die Projektarbeit hatte uns am Anfang vor großen Herausforderungen gestellt und mit weit geöffneten Augen haben wir uns die Anforderungsliste des Projekts durchgelesen. Die Fragezeichen über unseren Köpfen verflogen jedoch mit der Zeit und wurden durch Ehrgeiz geweckt. Letzten Endes kann man sagen, dass es für uns am interessantesten und doch zugleich am schwierigsten war, so ein simples Spiel, das man damals gespielt hat, selbst zu programmieren. Sich hineinzuversetzen, was für solch ein Spiel nötig ist und wie man den Spaßfaktor für den Spieler maximieren kann, war eine lehrreiche Erfahrung. Für unser erstes

selbstprogrammiertes Spiel hat es uns sowohl als Benutzer als auch als Programmierer Spaß gemacht dieses Spiel zu programmieren und zu spielen.