Schiffeversenken in Python - SS2021

Marc Ullmann, David Ruschmaritsch, Anne Lotte Müller-Kühlkamp

Frankfurt University of Applied Sciences
(1971-2021: Fachhochschule Frankfurt am Main)
Nibelungenplatz 1
60318 Frankfurt am Main
marc.ullmann@stud.fra-uas.de, david.ruschmaritsch@stud.fra-uas.de,
anne.mueller-kuehlkamp@stud.fra-uas.de

Zusammenfassung Hier kommt das Abstract hin. Das Abstract sollte in wenigen Sätzen (weniger als 10 Zeilen) den Inhalt des Dokuments beschreiben. Wikipedia schreibt: Ein Abstract ist eine prägnante Inhaltsangabe, ein Abriss ohne Interpretation und Wertung einer wissenschaftlichen Arbeit. Das Abstract soll so kurz wie möglich sein. Alle wesentlichen Sachverhalte sollen explizit enthalten sein. Das Abstract soll beim Leser Interesse für den Inhalt des Dokuments wecken.

Das Spiel Schiffeversenken, welches früher nur mit Stift und Papier gespielt wurde und sich seit dem in seiner Grundstruktur kaum verändert hat, bietet in der Implementierung die optimalen Grundlagen der objektorientierten Programmierung und grafischen Darstellung.

1 Grundprinzip Schiffeversenken

Für die Implementierung des Spiels wird ein Spielfeld benötigt, welches mit Kommandozeilenargumenten anpassbar sein soll. Die Beschränkung von einer Mindestgröße von 10x10 Feldern und Maximalgröße von 20x20 Feldern erwies sich als sinnvoll, da bei mehr als 20 Feldern die Größe des Standard-Fenster überschritten wird und somit das Programm abstürzt. Bei weniger als zehn Feldern könnte es zu Platzproblemen der Schiffe mit dem Platzier-Algorithmus geben und somit unerwünschte Ereignisse hervorrufen. Deshalb wird bei Eingabe von zwei Kommandozeilenargumenten die Begrenzung abgefragt, die bei Abweichung die Standardgröße von 10x10 wählt. Für die Implementierung von Schiffen ist nun die Grundlage des Spielfelds gegeben. Ein Schiff ist als Objekt zu definieren. Die Schiffanzahl ist von Grund auf festgelegt auf insgesamt zehn Schiffe [1]:

- 1x Schlachtschiff mit Größe 5
- 2x Kreuzer mit Größe 4
- 3x Zerstörer mit Größe 3
- 4x U-Boote mit Größe 2

2 Visuelle Darstellung

Zu Beginn kam es zur Wahl der Bibliothek für die visuelle Darstellung. Zuerst fiel diese auf termbox [2]. Da diese jedoch nicht mehr unterstützt und weiterentwickelt wird, fiel die Entscheidung zuerst auf cursebox [3], für welche die einfachen Befehle und praktische Funktionen sprachen. Da auch diese nicht weiterhin unterstützt wird und wenig Dokumentation zu der Bibliothek vorliegt, wurde das Programm in curses [4] übersetzt.

2.1 Spielfeld

Das Spielfeld wurde mit Hilfe von der Erweiterung numpy [5] entwickelt. Die Erweiterung bietet eine einfache Erstellung mehrdimensionaler Arrays und liefert verschiedene Operationen um diese zu konfigurieren. Ein zweidimensionales Array bietet die optimale Grundlage für das Spielfeld, da dies wie ein 2D-Koordinatensystem betrachtet werden kann. Mit einem Null-Array (numpy.zeroes) ist es nur möglich Zahlen zu verarbeiten. Dies hat zur Auswirkung, dass Felder die beispielsweise mit Schiffen belegt sind, durch eine „1“ und Felder ohne Schiffe eine „0“ repräsentiert werden. Dadurch ist es einfacher Felder zu vergleichen, da diese nur mit Zahlen belegt sind und nicht weiter übersetzt werden müssen. Für die grafische Darstellung des Null-Arrays bedeutet dies jedoch, eine Einschränkung um die Objekte ansehnlich darzustellen. Das Chararray, welches Zeichen speichern kann, bietet sich hier an. Jedoch wird bei der Darstellung des Arrays immer ein „b“ für Byte hinzugefügt, da es Bytestrings sind, die in dem Array gespeichert werden. Dies wurde umgangen indem der Datentyp des Arrays als Zeichenkette übersetzt und dann ausgegeben wird. Für die Darstellung eines Spielfeldes und nicht der Hinternanderreihung einzelner Objekte des Arrays, wird jede Reihe des Arrays einzeln in einer eigenen Zeile ausgegeben.

Um die Vorteile beider Arrays zu benutzen, müssen verschiedene Arrays erstellt werden. Insgesamt gibt es fünf Arrays: Zwei Charrays für die grafische Darstellung und drei Null-Arrays für die logische Verarbeitung.

2.2 Spielfeld: Schiffplatzierung

In diesem Fall wird nur das eine Charray benutzt, welches die ersten beiden logischen Arrays der Schiffplatzierung visuell übersetzt. Dafür wird ein „*“ (logisch=2) für die aktuelle Position des Schiffs und ein „X“ (logisch=1) für schon platzierte Schiffe benutzt. Die leeren Felder werden durch „O“ (logisch=0) dargestellt.

2.3 Spielfeld: Beschuss

Nach der Platzierung der Schiffe wird nun abwechselnt gefeuert. Dafür werden nun durchgehend zwei Spielfelder (durch die beiden Chararrays) angezeigt. Das linke zeigt dabei das Spielfeld des Gegners an, auf welchem die aktuelle Position und schon beschossene Felder angezeigt werden. Dafür werden die bereits Getroffenen als „*“ (logisch=2), Wassertreffer als „Prozentzeichen“ (logisch=3), das Aktuelle als „X“ (logisch=1) und Leere als „O“ (logisch=0) angezeigt. Das rechte Spieldfeld zeigt das eigene Spielfeld, mit allen Schiffpositionen und Schüssen des Gegners, an. An der oberen Seite wird der aktuelle Spieler und dessen verbleibenden Schiffe angezeigt, an der unteren die des Gegners.

2.4 Sonstige visuelle Elemente

Um die Bedienung zu erleichtern, werden am unteren Ende die aktuell möglichen Eingaben angezeigt.

2.5 Die Funktion *update_matchfield()*

Diese Funktion ist für den großteil der visuellen Darstellung verantwortlich. Ihre Aufgabe ist es, den aktuellen Spieler anzuzeigen, die logischen in die visuellen Arrays zu übersetzen und einige visuelle Elemente hinzuzufügen. Der aktuelle Spieler wird der Funktion als Parameter übergeben und dann ausgegeben.

Um ein Spielfeld zu übersetzen, wird der Funktion immer ein Charray (visuelles Spielfeld) und ein Null-Array (logisches Spielfeld) übergeben. Um zwei Spielfelder darzustellen, wird die Funktion zwei mal aufgerufen. Dies stellt ein vor das Problem, dass der aktuelle Bildschirm nicht komplett gelöscht werden kann, sondern aktuelle Elemente nur überschrieben werden können. Das heißt alle Elemente die nicht genau übereinanderliegen, müssen durch ein leeren String ersetzt werden. Da das Spielfeld des Gegners und des aktuellen Spielers immer an der gleichen Position liegen, überschreiben diese sich automatisch. Andere Elemente wie die Anzahl der Schiffe und die Anzeige der Eingaben werden in eigenen Funktionen aufgrufen.

Mit zwei for-Schleifen wird das logische Spielfeld Schritt für Schritt in das Visuelle übersetzt. Zudem gibt es eine Änderung wenn man gegen den Computer spielt. Da das Feld von diesem, auch für den menschlichen Spieler sichtbar ist, werden seine Schiffspositionen verdeckt. Dies ermöglicht dem Spieler direkt zu sehen, worauf der Computer schießt. An den Rändern der Spielfelder wird durch Sonderzeichen das Spielfeld abgegrenzt, wodurch der Überblick erleichtert wird. Zudem sind die Seiten für die Texteingabe durchgehend nummeriert.

3 Mausimplementation

Grundsätzlich soll das Spiel mit der Tastatur spielbar sein, jedoch gab es in den Anforderungen auch die Wahl eine Maussteuerung einzubinden. Die benutzte Programmbibliothek „curses“, welche zur Darstellung zeichenorientierter Benutzerschnittstellen unabhängig vom darstellenden Textterminal ist, bietet hierzu die Funktion „getMouse()“ an. Diese Funktion liefert ein 5-stelliges Tupel. Darin enthalten sind unter Anderem die x- und y-Koordinaten der Maus, welche mit einem Tastenereignis (hier ein Mausklick) abzufragen sind. Nun gilt es die in dem curses-Terminal dargestellten Texte, welche als Knöpfe dienen sollen, per Mausklick zu aktivieren. Dazu müssen die Koordinaten der Maus mit denen der dargestellten Texte übereinstimmen. Dies erwies sich als umständlich und komplex für die weitere Implementierung des Spiels, da die dargestellten Strings keine Objekte sind und damit keine Funktionen zur Abfrage und dem damit verbundenen Vergleich der Mausposition hatten und man diese aufwendig herausfinden müsste. Deshalb funktioniert die Maus nur im Anfangsmenü.

4 Benutzereingaben

Für die Entgegennahme der Eingaben des Benutzers wurde die Funktion „userinput()“ entwickelt, welche die gedrückte Taste (oder Maus) entgegen nimmt. Die Eingabeerkennung wird mit der curses-Funktion „get_wch()“ erzeugt. Diese gibt Zeichen für die meisten Tasten zurück. Bei der Eingabe wird festgelegt, welche Tasten für welche Richtungen verantwortlich sind. Die obere Pfeiltaste und die Taste „w“ ist so beispielsweise für die Richtung Norden.

```
1  def userinput(screen):
2  ''' Checks user input '''
3  input_key = ""
4  curinput = ""
5  screen.keypad(1)
6  curses.mousemask(-1)
7
8  curinput = screen.get_wch()
9
10 if curinput == 'd' or curinput == curses.KEY_RIGHT:
11     input_key = "right"
12 elif curinput == 's' or curinput == curses.KEY_DOWN:
13     input_key = "down"
14 elif curinput == 'a' or curinput == curses.KEY_LEFT:
15     input_key = "left"
16 elif curinput == 'w' or curinput == curses.KEY_UP:
17     input_key = "up"
18 elif curinput == '\n':
19     input_key = "enter"
```

```
20 elif curinput == curses.KEY_MOUSE:
21     input_key = "mouse"
22 else:
23     input_key = curinput
24 return input_key
```

Listing 1.1. Funktion userinput() nimmt Eingaben des Benutzers an

5 Die Klasse Ship

Alle Schiffe sind als Objekte definiert. Objekte bieten hier einen erheblichen Vorteil durch deren Attribute. In jedem Schiff werden die Größe, Rotation, Startposition und Koordinaten gespeichert. Die Koordinaten berechnen sich durch die Objektmethode „ship_cords“, welche je nach Ausrichtung (horizontal, vertikal) eine Schleife bis zur Größe des Schiffs durchläuft und von der Startkoordinate aus in die jeweilige Richtung läuft.

6 Schiffplatzierung

Das erste logische Array speichert die temporären Daten, welche für die Berechnungen von der Spielfeldbegrenzung dienen. Das zweite speichert tempöräre Daten, wenn diese die Regeln nicht verletzen. Für die Platzierung der Schiffe wird nun in dem tempörären Array bestimmt, ob die Schiffsgröße die Spielfeldgröße überschreitet. Wenn schon Schiffe platziert wurden gleicht es zudem mit dem zweiten logischen Array die Position dieser ab und berechnet so die Möglichkeit der Platzierung. Bei der Platzierung werden die umliegenden Felder reserviert, da die Schiffe sich nicht direkt berühren dürfen. Das dritte logische Array speichert hingegen nur die Schiffspositionen ab, um diese später zu übergeben.

7 Die Funktion *shoot()*

Nachdem der Spieler die Schiffe platziert hat. Beginnt das eigentliche Spiel, nämlich der Beschuss des Gegners. Der Spieler kann mit WASD, Pfeiltasten oder durch Eingabe e entscheiden, welches Feld beschossen werden soll. Dies wird mit der Taste „enter“ bestätigt. Mit dem Tastendruck wird zuerst geprüft, ob dieses Feld bereits beschossen wurde. Dies gilt nicht für den ersten Schuss, da bisher kein Feld beschossen wurde. Trifft der Schuss kein Schiff aber ein gültiges Feld, so wird das Feld im logischen Array mit einer „3“ belegt. Trifft der Schuss jedoch ein Schiff, wird das getroffene Schiff mit Koordinatenabgleich identifiziert

und das getroffene Feld aus den Koordinaten des Schiffs gelöscht. Sobald alle Felder des Schiffs getroffen wurden wird das Schiffobjekt aus der ship_list_placed Liste gelöscht. Bei zerstörten Schiffen werden die naheliegenden Felder reserviert, da sich dort keine Schiffe mehr befinden können.

Literatur

1. Schiffeversenken. wikipedia.org. 2021
   https://github.com/nsf/termbox
2. nsf. Termbox. Github. 2020
   https://github.com/nsf/termbox
3. Tenchi2xh. Cursebox. github.com. 2018
   https://github.com/Tenchi2xh/cursebox
4. A.M. Kuchling, Eric S. Raymond, Curses, python.org. 2021
   https://docs.python.org/3/howto/curses.html
5. Numpy, numpy.org. 2021
   https://numpy.org/