Informatik I: Einführung in die Programmierung

Prof. Dr. Bernhard Nebel
Tim Schulte, Thorsten Engesser
Wintersemester 2017/2018

Universität Freiburg
Institut für Informatik

Übungsblatt 12

Abgabe: Freitag, 26. Januar 2018, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet12` im Wurzelverzeichnis Ihrer Arbeitskopie des SVN-Repositories an. Ihre Lösungen werden dann in Dateien in diesem Unterverzeichnis erwartet. Beachten Sie bitte bei allen Aufgaben die *Hinweise zur Bearbeitung der Übungsaufgaben* unter der folgenden URL:

`http://gki.informatik.uni-freiburg.de/teaching/ws1718/info1/guide/hinweise.html`

Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 12.1 (Micro Bit: Schiffe versenken; Datei: `battleships.py`; Punkte: 6+3+3)

In dieser Aufgabe soll eine Micro Bit-Variante des bekannten Ratespiels „Schiffe versenken“[1] implementiert werden. Hierbei verfügen zwei Spieler über eine Flotte von Schiffen, deren Position dem jeweils anderen Spieler nicht bekannt ist. Ziel ist es, als Erster die gesamte gegnerische Flotte zu versenken.
Hierzu positionieren beide Spieler zunächst ihre Flotte auf dem eigenen, für den jeweils anderen Spieler nicht einsehbaren, Spielfeld. Das Spielfeld ist ein, dem Micro Bit LED-Display entsprechendes, 5x5 Raster. Die Spielerflotten bestehen jeweils aus zwei Schiffen, einem U-Boot und einem Zerstörer. Das U-Boot belegt zwei horizontal oder vertikal aufeinanderfolgende Rasterfelder, der Zerstörer Drei. Haben beide Spieler ihre Flotte positioniert, greifen sie sich abwechselnd an. Hierbei ist ein Spieler immer Angreifer, der andere Verteidiger. In jeder Runde bestimmt der Angreifer eine Position auf dem Spielfeld des Verteidigers, und teilt sie diesem mit. Der Verteidiger überprüft anschließend, ob eines seiner Schiffe die gewählte Position belegt und somit getroffen wurde, und teilt dem Angreifer das Ergebnis (`"HIT"` oder `"MISS"`) mit. Anschließend wechseln die Spieler die Rollen, der Angreifer wird zum Verteidiger und umgekehrt, und die nächste Runde startet. Ein Schiff gilt als *versenkt*, wenn der Gegner jedes Feld des Schiffes mindestens einmal getroffen hat. Gewonnen hat der Spieler der zuerst beide Schiffe des Gegners versenkt hat. Anders als im original Spiel, greifen die Spieler immer abwechselnd an, und dürfen nicht erneut angreifen, wenn sie einen Treffer gelandet haben.
Zum einfacheren Verständnis haben wir ein kurzes Video erstellt, welches die Grundlagen des Spiels erläutert und die Funktionsweise der zu implementierenden Funktionen aufzeigt. Betrachten Sie zunächst das Video `battleships_demo.m4v`. Laden Sie dann das Template `battleships.py` von der Kurswebseite herunter und machen Sie sich mit dem Code vertraut.

(a) Implementieren Sie die Funktion `spawnships()`. Diese erzeugt eine zufällig bestimmte, legale Anordnung der zwei Schiffe des Spielers, und gibt sie als Image-Objekt zurück. Die Schiffskoordinaten sollen im erzeugten Image-Objekt durch die Pixelhelligkeit `BRIGHTNESS_SHIP` dargestellt werden. Allen anderen Koordinaten haben die Pixelhelligkeit 0. Eine Anordnung ist legal, wenn die Schiffe sich nicht überschneiden

[1] `https://de.wikipedia.org/wiki/Schiffe_versenken`

oder berühren, also mindestens ein Rasterfeld Abstand zueinander haben. Außerdem müssen die Schiffe die vorgegebenen Längen haben. Weiterhin sollten alle legalen Anordnungen generierbar sein.

(b) Vervollständigen Sie die Funktion `defend(myboard)`. Diese implementiert das Verhalten des Spielers in der Rolle des Verteidigers und erhält das eigene Spielfeld (Image-Objekt) als Argument. Zunächst wird auf den Erhalt einer Nachricht des Angreifers gewartet, welche die entsprechenden Angriffskoordinaten enthält. Der Verteidiger überprüft dann, ob eines seiner Schiffe getroffen wurde und markiert die betroffene Position auf dem eigenen Spielfeld, indem die Pixelhelligkeit auf `BRIGHTNESS_HITSHIP` (Treffer), bzw. `BRIGHTNESS_SHOT` (Fehlschlag) gesetzt wird. Anschließend wird eine Nachricht an den Angreifer gesendet. Diese enthält den String `"HIT"` im Falle eines Treffers und `"MISS"` sonst.

Bonus: Visualisieren Sie zusätzlich, wo der Angreifer zugeschlagen hat, indem Sie die Pixelhelligkeit an dieser Stelle eine Sekunde lang auf `BRIGHTNESS_CURSOR` setzen und erst dann die Stelle auf dem Spielfeld, wie oben beschrieben, als Treffer oder Fehlschlag, markieren (siehe Video).

(c) Vervollständigen Sie die Funktion `attack(otherboard)`. Diese implementiert das Verhalten des Spielers in der Rolle des Angreifers und erhält das gegnerische Spielfeld (aus Sicht des Angreifers) als Argument. Zunächst wird eine Position auf `otherboard` gewählt und dem Gegner anschließend in einer Nachricht übermittelt. Dann wird auf die Antwort des Gegners gewartet (`"HIT"` oder `"MISS"`) und die betroffene Position auf `otherboard` markiert (`BRIGHTNESS_HITSHIP` bei einem Treffer, `BRIGHTNESS_SHOT` sonst). Zusätzlich soll die erhaltene Nachricht auf dem display angezeigt werden. Verwenden Sie hierzu die Funktion `display.scroll`.

Hinweis: Da zum Testen der Aufgabenteile (b) und (c) ein zweiter Micro Bit benötigt wird, dürfen Sie diese Aufgabe in Gruppen bis zur Größe Vier bearbeiten. Bitte vermerken Sie die Namen aller Gruppenmitglieder und des jeweiligen Tutors als Kommentar am Beginn der Datei `battleships.py`. Sie dürfen (und sollen) dann alle die gleiche Datei in ihr jeweiliges Abgabeverzeichnis committen. Wenn Sie keinen Partner finden, können Sie zum Testen ins betreute Programmieren gehen.

Aufgabe 12.2 (Reguläre Ausdrücke; Punkte: 4+2; Datei: `month.py`)

(a) Schreiben Sie eine Funktion `find_dates`, die aus einem Text alle Datumsangaben der Gestalt wie im Beispiel

```
29. Februar 2016, 8:05 Uhr
```

findet und als eine Liste von Dictionaries zurückgibt. Dabei soll das Leerzeichen vor der Monatsangabe optional sein. Jedes Dictionary habe dabei die Schlüssel `year`, `month`, `day`, `hour`, und `minute`, denen numerische Werte zugewiesen sind. Das Dictionary im Beispiel würde also wie folgt aussehen:

```
{'minute': 5, 'month': 2, 'hour': 8, 'year': 2016, 'day': 29}.
```

(b) Schreiben Sie für die neu implementierte Funktion eine Test-Funktion mit 4 Assertions.

Aufgabe 12.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet12` eine Textdatei `erfahrungen.txt` an. Notieren Sie in dieser Datei kurz Ihre Erfahrungen beim Bearbeiten der Übungsaufgaben (Probleme, benötigter Zeitaufwand nach Teilaufgabe, Bezug zur Vorlesung, Interessantes, etc.).