Informatik I: Einführung in die Programmierung

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

Universität Freiburg
Institut für Informatik

Übungsblatt 9

Abgabe: Freitag, 22. Dezember 2017, 20:00 Uhr

Wichtige Hinweise: Zur Bearbeitung der Übungsaufgaben legen Sie bitte ein neues Unterverzeichnis `sheet09` 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`

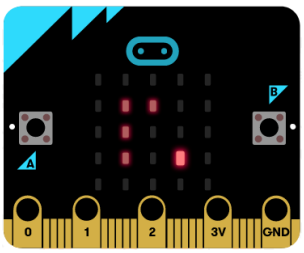
Bewertet wird bei allen Aufgaben die letzte Version, die zur Deadline des Übungsblattes auf dem SVN-Server eingereicht ist.

Aufgabe 9.1 (Micro Bit: Snake Part II; Dateien: `snake2.py`; Punkte: 3+2+4)

Nachdem wir auf dem letzten Übungsblatt die Grundlage für ein kleines *Snake*[1]-Spiel geschaffen haben, wollen wir dieses nun vervollständigen. Dazu sind folgende Änderungen erforderlich: (1) Der Spieler ist eine Schlange und wird, anstatt durch eine einzelne Koordinate, durch eine Liste von Koordinaten (die Schlangenglieder vom Kopf bis zum Schwanzende) repräsentiert. (2) Zusätzlich zum Spieler befindet sich immer eine Maus auf dem Spielfeld, welche von der Schlange gefressen werden kann. (3) Die Schlange wächst wenn sie die Maus frisst, woraufhin eine weitere Maus an einer zufällig bestimmten Stelle auftaucht. Die Schlange schrumpft wenn sie frontal in eine Wand läuft. Die Bewegungsrichtung der Schlange wird weiterhin durch Drücken des A- bzw. B-Knopfes verändert. Laden Sie zunächst das Template `snake2.py` von der Kurswebseite herunter. (Falls sie Schwierigkeiten bei der Bearbeitung der vorangegangenen Aufgabe hatten, versuchen Sie zuerst die bereitgestellte Musterlösung `snake_simple.py` zu verstehen).

(a) Implementieren Sie die Funktion `render(player, dot)`. Diese erhält die Koordinatenliste des Spielers, sowie die Koordinaten der Maus als Argument und gibt ein `Image`-Objekt zurück. Die Pixelhelligkeit im erzeugten `Image`-Objekt soll an den Spielerkoordinaten den Wert `PLAYER_BRIGHTNESS` und an den Mauskoordinaten den Wert `DOT_BRIGHTNESS` haben. Sie können Ihre Funktion testen, indem Sie den Main-Loop vorübergehend auskommentieren und Ihre Funktion in der REPL des mu-Editors mit `display.show(render(player, dot))` aufrufen, z.B:

```
>>> player = [(2,1), (1,1), (1,2), (1,3)]
>>> dot = (3,3)
>>> display.show(render(player, dot))
```



Informationen zu den `Image` und `Display` Modulen finden Sie unter folgenden URLs:
`http://microbit-micropython.readthedocs.io/en/latest/image.html`
`http://microbit-micropython.readthedocs.io/en/latest/display.html`

[1] `https://de.wikipedia.org/wiki/Snake_(Computerspiel)`

(b) Implementieren Sie die Funktion `spawn_dot(player)`. Diese erhält die Koordinatenliste des Spielers und gibt eine neue zufällig bestimmte Koordinate zurück. Diese muss eine gültige Position auf der LED Matrix repräsentieren und von den Spielerkoordinaten verschieden sein. Verwenden Sie hierzu die Funktion `randint` aus dem Modul `random`.

(c) Implementieren Sie die Funktion `update(player, direction, dot)`. Diese erhält die Koordinatenliste des Spielers, dessen Bewegungsrichtung, sowie die aktuelle Position der Maus und gibt eine aktualisierte Koordinatenliste des Spielers, sowie die neue Position der Maus zurück. Jeder Aufruf von `update` bewegt den Spieler eine Einheit in die vorgegebene Richtung. Hierzu wird zuerst der Kopf der Schlange eine Einheit in die vorgegebene Richtung bewegt. Anschließend rücken der Reihe nach alle Glieder nach. Dies kann erreicht werden, indem ein neuer Kopf an den Anfang der Koordinatenliste eingefügt wird und das letzte Element entfernt wird. Trifft der Schlangenkopf auf eine Maus (`dot`), so wächst die Schlange um ein Glied. Dies kann erreicht werden indem das letzte Element beibehalten wird. Außerdem soll in diesem Fall, mit Hilfe der `spawn_dot` Funktion, die neue Position der Maus bestimmt werden. Läuft eine, aus zwei oder mehr Gliedern bestehende, Schlange frontal in eine Wand oder in sich selbst, verliert diese ihr letztes Glied. Besteht die Schlange aus einem einzigen Glied (ihrem Kopf), passiert nichts. Betrachten Sie das folgende Beispiel:

```
>>> update(player=[(1,2),(0,2)], direction='E', dot=(3,2))
([(2,2),(1,2)], (3,2))
>>> update([(2,2),(1,2)], 'E', (3,2))
([(3,2),(2,2),(1,2)], (1,4))  # dot wurde gefressen
                              # und zufällig neu gesetzt
>>> update([(3,2),(2,2),(1,2)], 'E', (1,4))
([(4,2),(3,2),(2,2)], (1,4))
>>> update([(4,2),(3,2),(2,2)], 'E', (1,4))
([(4,2),(3,2)], (1,4))  # die Schlange ist gegen eine Wand gelaufen
                        # und eine Einheit geschrumpft
>>> update([(4,2),(3,2)], 'E', (1,4))
([(4,2)], (1,4))
>>> update([(4,2)], 'E', (1,4))
([(4,2)], (1,4))  # die Schlange besteht nur noch aus ihrem Kopf
                  # und kann nicht weiter schrumpfen
```

Aufgabe 9.2 (Ellipse; Dateien: `ellipse.txt`, `hw_geoclasses.py`; Punkte: 1+4+4)

In dieser Aufgabe geht es darum die in der Vorlesung vorgestellten Geometrie-Klassen um eine Klasse `Ellipse` zu erweitern, mit der Ellipsen[2] repräsentiert werden sollen. Wir gehen dabei davon aus, dass die Achsen von Ellipsen parallel zu den Bildschirmkoordinaten sind (die Hauptachse der Ellipse ist also parallel zur x- oder zur y-Achse). Um diese Aufgabe zu realisieren erweitern Sie die in der Vorlesung vorgestellten Geometrie-Klassen. Laden Sie hierzu zunächst die Datei `hw_geoclasses.py` von der Kurswebseite herunter.

(a) Durch welche Parameter kann eine Ellipse sinnvoll und eindeutig beschrieben werden? Von welcher bereits existierenden Klasse sollte sich Ihre Klasse `Ellipse` sinnvoller Weise ableiten und warum?

[2] `http://de.wikipedia.org/wiki/Ellipse`

Schreiben Sie Ihre Antworten zu dieser Teilaufgabe in die Datei `ellipse.txt` und vergessen Sie nicht, diese ins SVN zu committen.

(b) Ergänzen Sie die Datei `hw_geoclasses.py` mit Ihrer Implementierung der Klasse `Ellipse`. Diese Klasse soll neben einer geeigneten `__init__`-Methode, analog zu den vorhandenen Klassen, auch die Methoden `area` (zur Berechnung des Flächeninhalts der Ellipse) und `change_size` (zur prozentualen Veränderung der Größe) enthalten, sowie `stretch_height` und `stretch_width` (zur prozentualen Veränderung von Höhe/Breite, ohne dabei die jeweils andere Diemnison zu verändern).

(c) Nutzen Sie das in der Vorlesung eingeführte Modul `tkinter`, um zwei verschiedene Instanzen der Klasse `Ellipse` nebeneinander als grafische Objekte zu visualisieren.

Aufgabe 9.3 (Erfahrungen; Datei: `erfahrungen.txt`; Punkte: 2)

Legen Sie im Unterverzeichnis `sheet09` 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.).