Einführung in die Programmierung

Prof. Dr. Detlef Krömker
Alexander Wolodkin

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt Nr. 5[1]

Ausgabe: 30.11.2017
Abgabe: 16.12.2017
16:00 Uhr

Pah Tum

Hinweis: *Dieses Aufgabenblatt ist mit einer Bearbeitungszeit von zwei Wochen dafür ausgelegt in einem Zweierteam gelöst zu werden.* ***Bitte beachten Sie, dass die/der Programmierpartner*in einmal gewechselt werden muss, also für diese Aufgabe oder aber spätestens für EPR 06.*** *Neben der Implementierung wird bei der Bearbeitung Wert auf die folgenden Punkte gelegt: Dokumentation[2], Strukturierung und Einhalten des Style-Guides[3].*

Achtung: Achten Sie darauf die Variable `__author__` in **allen** Quellcode Dateien **korrekt** zu setzen. Abgaben die nicht dieser Vorgabe entsprechen werden **nicht** bewertet! Quellcode muss als *.py* Datei und alles Weitere als *.pdf* Datei abgegeben werden. Bei Abgaben mehrer Dateien müssen diese als *.zip* zusammengefasst werden. Abgaben, die nicht diesen Regeln entsprechen, werden ebenfalls **nicht** bewertet! Außerdem muss ihr Name in jeder abgegebenen *.pdf* Datei zu finden sein.

Beachten Sie das Programmierhandbuch, für Aufgaben ab der Quartalwoche 47 ist die Verwendung des Headers verpflichtend. Abgaben, die nicht dieser Vorgabe entsprechen, werden nicht bewertet.

Σ __ / **20**

Aufgabe 5.1: Einleitung

Wir wollen in dieser Aufgabe anhand eines klassischen Spiels verschiedene User-Interfaces systematisch entwickeln. Die folgende Spielbeschreibung entstammt der Wikipedia:

Pah Tum *ist ein strategisches Brettspiel für zwei Spieler. Es stammt aus Mesopotamien und Assyrien und ist mindestens 3800 Jahre alt. Damit zählt es zu den ältesten bekannten Spielen.*

Regeln
Pah Tum wird auf einem Brett mit 7 × 7 Feldern gespielt. Ein Spieler setzt weiße, der andere schwarze Steine auf das Brett. Zu Beginn des Spiels wird eine ungerade Zahl von Feldern blockiert, was meist durch einen Zufallsgenerator erfolgt. Die Spieler setzen abwechselnd je einen Stein auf ein freies, nicht blockiertes Feld, Weiß beginnt.

Hinweis: Für die Umsetzung dieser Aufgabe ist die Anzahl der blockierten Felder vom Benutzer/Spieler entgegen zu nehmen. Diese darf aus ungeraden ganzen Zahlen aus dem Intervall $[5, 13]$ bestehen. Die Auswahl der zu besetzenden Felder

[1] Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein.
[2] `http://www.python.org/dev/peps/pep-0257`
[3] `https://www.python.org/dev/peps/pep-0008/`

erfolgt **gleichverteilt**.

Das Spiel endet, wenn es kein freies Feld mehr gibt. Die Spieler bekommen dann Punkte für die waagerechten und senkrechten Reihen, die sie aus ihren eigenen Steinen gebildet haben und die nicht durch gegnerische Steine oder blockierte Felder unterbrochen sind. Diagonale Reihen zählen nicht. Die Reihen dürfen sich überkreuzen, d. h. ein Stein kann gleichzeitig Teil einer waagerechten und senkrechten Reihe sein. Für jeden Spieler werden die Punkte die er für seine Reihen erhält, addiert. Der Spieler mit der höheren Punktzahl gewinnt, bei Gleichstand ist das Spiel unentschieden.

Die Punktzahl für eine Reihe richtet sich nach deren Länge:

- *1 oder 2 Steine: kein Punkt*
- *3 Steine: 3 Punkte*
- *4 Steine: 10 Punkte*
- *5 Steine: 25 Punkte*
- *6 Steine: 56 Punkte*
- *7 Steine: 119 Punkte*

***[Hinweis:]** Diese Punktzahlen ergeben sich, wenn man für eine Reihe von mindestens drei Steinen so viele Punkte gibt, wie Steine in der Reihe sind, und die beiden enthaltenen um eins kürzeren Reihen ebenfalls wertet. Eine Fünferreihe bringt also fünf Punkte, und die darin enthaltenen Viererreihen noch jeweils zehn, also insgesamt 25 Punkte.*

Undo ausführlicher erläutert: In der implementierten Version kann man immer wieder 1 Schritt zurück setzen und eine erneute Auswahl treffen, auch hintereinander. Oder mehrere Schritte gleichzeitig. Der Gegenspieler wird in diesem Fall in seiner Auswahl ebenfalls zurück gesetzt.

Aufgabe 5.2: **Konsolenspiel** Punkte: ____ / 7

Entwickeln Sie ein Konsolenspiel in Python 3.6, so dass zwei Spieler*innen **Pah Tum** auf dem Computer spielen können. Das Programm lässt regelwidrige Eingaben nicht zu. Es gibt angemessene Hilfestellungen für die Spieler*innen.

Entwickeln Sie das Programm so, dass das Spielfeld scheinbar statisch auf dem Bildschirm steht und nicht bei jedem Zug komplett neu (sichtbar) geschrieben wird. ***Hinweis:** Geben Sie sich etwas Mühe mit der graphischen Ausgestaltung des Spiels; hierfür gibt es 1,5 Teilpunkte.*

Die Anzahl der blockierten Felder soll eingelesen werden. Berechnen Sie die erreichte Punktezahl für schwarz und weiß. Realisieren Sie einen Undo-Mechanismus (ggf. mehrfach einen Schritt zurücksetzen).

Geben Sie ein Zustands-Übergangsdiagramm an. Entwickeln Sie eine „Eingabesprache“ und begründen Sie den Entwurf.

Schreiben Sie eine Bedienungsanleitung für dieses Spiel (*1,5 Teilpunkte*).

Aufgabe 5.3: **GUI** Punkte: ____ / 7

Entwickeln ein GUI in Python 3.6 mit TKinter, so dass zwei Spieler*innen **Pah Tum** auf dem Computer spielen können. Das Programm lässt regelwidrige Eingaben nicht zu. Es gibt angemessene Hilfestellungen für die Spieler*innen.

Geben Sie sich etwas Mühe mit der graphischen Ausgestaltung des Spiels
Hinweis: *Auch hierfür gibt es 1,5 Teilpunkte.*

Die Anzahl der blockierten Felder soll eingelesen werden. Schreiben Sie eine Bedienungsanleitung für dieses Spiel (*1,5 Teilpunkte*). Sie dürfen den Code aus Aufgabe 5.1 wiederverwenden oder auch nicht, wie es Ihnen gefällt. Berechnen Sie die erreichte Punktezahl für schwarz und weiß.

Aufgabe 5.4: **Modifikation** Punkte: ____ / 6

Modifizieren Sie die Entwicklungen aus den Aufgaben 5.1 und 5.2 so, dass Sie einen Spieler durch einen Computerspieler (einen implementierten Algorithmus) ersetzen. Geben Sie die gewählte Spielstrategie dieses für die Realisierung genutzten Algorithmus an.

Punktevergabe:

- *Randomisierte Auswahl:* ***1 Punkt***
- *Eine rein Greedy-basierte Auswahl:* ***50% der Punkte***
- *Die KI plant 1 Zug voraus und versucht an der ersten Stelle die Vervielfältigung des gegnerischen Punktestandes (lange Reihe) zu verhindern. An der zweiten Stelle versucht die KI eigene Punkte zu maximieren.* ***100% der Punkte***