Mancala Stebler & Utz

# Vorgehen

Wir haben via <https://www.github.com> am selben Code gearbeitet. Grundsätzlich hat am Anfang jeder eigenen Ideen ausprobiert, welche schlussendlich in einem gemeinsamen Projekt zusammengefasst wurden.

# Spiel (Regeln & Inplementation)

## Feld

Das Feld besteht grundsätzlich aus 14 Feldern. Jeder Spieler besitzt sieben davon und kann aus sechs ziehen. Im Bild unten sichtbar sind die sechs Felder, aus denen Spieler 1 ziehen kann und das eine (grosse) Feld, welches nicht ziehbar ist und gleichzeitig die Punkte von Spieler 1 beinhaltet.

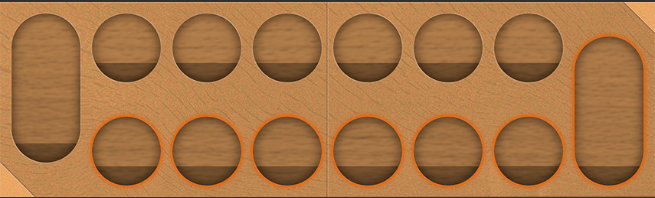


Abbildung Bild von <http://play-mancala.com/#!/1638e3f0bbc6522b7de9e66c0>

## Ziehen

Gezogen wird abwechselnd. Es können nur Felder gezogen werden, die Bohnen beinhalten. Anschliessend werden die Bohnen wir in Abbildung 2 beschrieben verteilt.

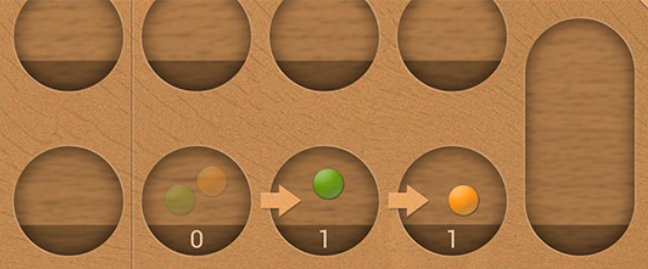


Abbildung Bild von <http://play-mancala.com/#!/1638e3f0bbc6522b7de9e66c0>

Es werden also alle Bohnen des gezogenen Feldes auf die nachfolgenden Felder verteilt. Bohnen werden auch ins eigenen Punktefeld gelegt, nicht aber ins Gegnerische.

## Gegnerische Bohnen schlagen

Man kann gegnerische Bohnen schlagen, wenn beim Verteilen die letzte Bohne in ein eigenes Feld geht, welches leer ist. Dann kann man die gegnerischen gegenüberliegenden Bohnen ins eigene Punktefeld verschieben.

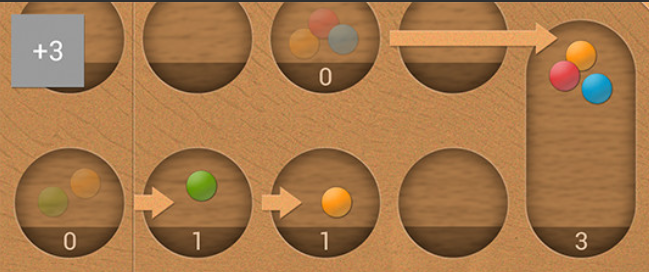


Abbildung Bild von <http://play-mancala.com/#!/1638e3f0bbc6522b7de9e66c0>

## Ende

Das Spiel endet, wenn auch einer Seite des Spielfeldes (nur kleine Felder) keine Bohnen mehr liegen. Derjenige Spieler, der auf seiner Seite noch Bohnen hat, darf diese in sein Punktefeld verschieben.

# Baum initialisieren

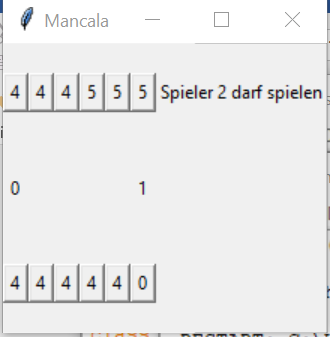
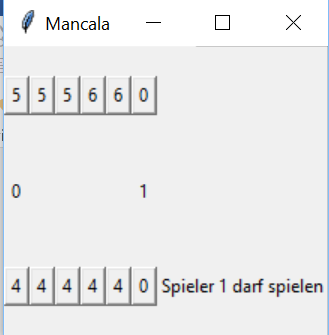
Die Initialisierung des Spielbaumes kann mit beliebiger Tiefe vorgenommen werden. Dann berechnet eine rekursive Funktion den Baum. Grundsätzlich funktioniert das wie folgt.



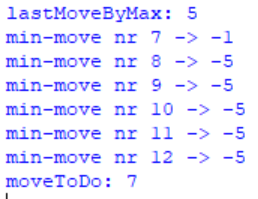
# Spieler

Der Spieler spielt gegen den MinMax-Player. Dieser rechnet nach jedem Zug des Spieler alle seine Möglichen Züge bis zur definierten Tiefe und wird dann den für Ihn besten Zug ausführen.

Move vom Spieler: Reaktion des Min-Max Player:

Berechnung zu der Reaktion des Min-Max Player:

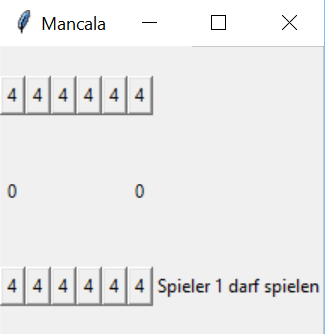
🡨 Letzter vom Spieler getätigter Zug

🡨 Mögliche Züge, bester Zug mit Differenz für Min-Max Player von -1

🡨 Zug der vom Min-Max Player ausgeführt wird

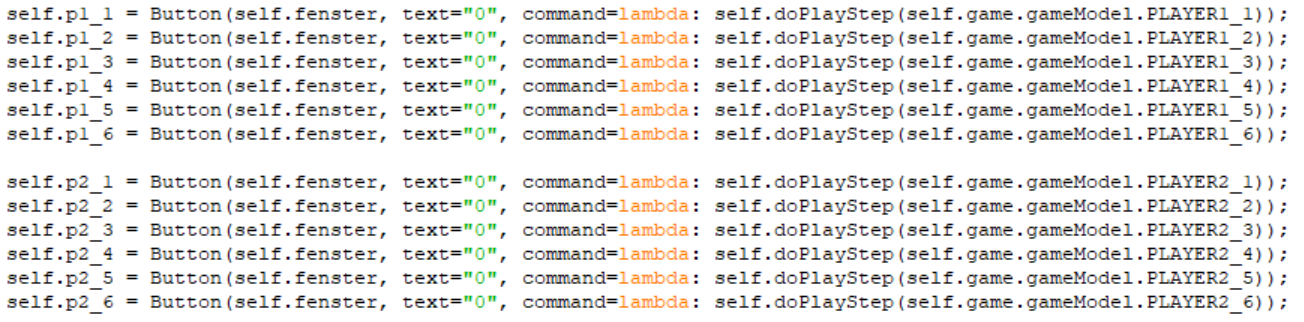
In dieser Situation reagiert der Min-Max Player auf die Möglichkeit das der Player im nächsten Zug sich durch Spielen des 2 Feldes 5 Bohnen holt.

# View

Das Spiel wird über folgende Oberfläche gespielt:

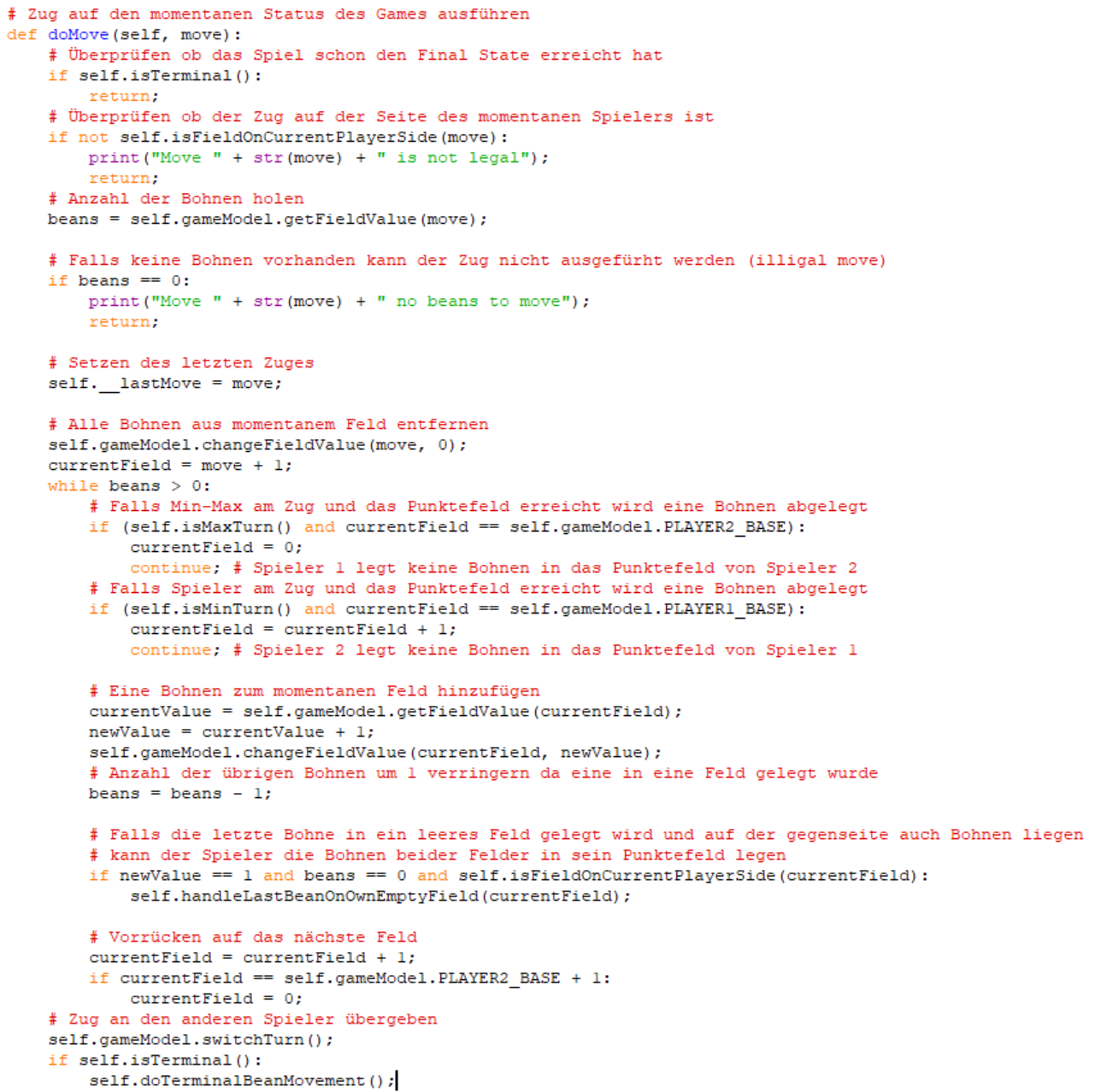
Bohnenfächer Min-Max Player Punktefeld Min-Max Player

Bohnenfächer Player Punktefeld Player

Jeder Spieler hat 6 Fächer mit Bohnen und 1 Punktefeld.   
Implementiert wurden die Bohnenfelder über Buttons. Damit diese Buttons die Funktion zum ausführen des Zuges aufrufen konnten musste ich sie mit einem Lambda Command versehen.  
Nachfolgend der Code dazu:

# Game (Zug ausführen)

Falls der Player sich für einen Zug entscheidet, wird die Methode doMove im Game Object aufgerufen. Diese Methode führ die Überprüfung der Gültigkeit des Zuges und die Ausführung des Zuges durch. Hier der Code mit den Kommentaren der Methode doMove:



# Anleitung zum Spielen

Hier eine kurze Anleitung, wie man das Programm bei sich selbst laufen lassen kann. Voraussetzung ist, dass Python 3.6 und git installiert sind. Anschliessend kann man mit dem folgenden Vorgehen das Programm installieren und starten.

1. Comandozeile öffnen
2. An gewünschte Stelle navigieren (z.B. cd Projektordner)
3. git clone <https://github.com/SoullessStone/mancalaMaster.git>
4. Rechtsklick auf «gameview.py»
5. Edit with IDLE (3.6)
6. Dann mit F5 das Programm starten

# Code

Wir verwenden in der Dokumentation nur kleine Code-Ausschnitte zum Verdeutlichen des Inhalts. Der vollständige Code ist auf Github verfügbar: <https://github.com/SoullessStone/>