Sogo

January 2018

Inhaltsverzeichnis

1 Dinge, die wir brauchen

- Funktion Start
  - startet das Spiel, zeigt den Grundplot
  - enthält alle weiteren Funktionen
- Funktion weiß
  - erlaubt Eingabe der Position + plottet weiße Kugel
- Funktion schwarz
  - erlaubt Eingabe der Position + plottet schwarze Kugel
- Plot weiße Kugel
  - kann weiße Kugeln an bestimmte Stellen plotten
- Plot schwarze Kugel
  - kann schwarze Kugel an bestimmte Stellen plotten
- Funktion Gewinner
  - testet nach jedem Zug, ob jemand gewonnen hat
- Random
  - man kann nur den Stab auswählen
    - daraus folgt Funktion Gravitation
- Funktion Gravitation
  - testet ob unter der gewählten Position schon etwas ist

2 Erklärung der Funktionen

2.1 Erstellung der Voraussetzungen

Bevor die erste Funktion überhaupt definiert werden kann, wird pylab importiert. Außerdem wird die Bedienungsanleitung ausgedruckt.

2.2 Funktion Start

In der Funktion *start* werden alle weiteren Voraussetzungen festgelegt. Darunter die 4×4 Matrizen A, B, C, D , die mit Nullen gefüllt sind und das leere Spielfeld darstellen. Weiterhin definiert sie die Listen Zeileweiß, Spalteweiß, Höheweiß, Zeileschwarz, Spalteschwarz und Höheschwarz, in die später die Koordinaten für den 3D Plot gespeichert werden.

2.2.1 Erstellung des Grundplots

Der Grundplot ist nichts weiter als das leere Spielfeld. Man sieht einen 3D Plot dessen 3 Achsen jeweils von 0 bis 3 gehen, sodass der Spieler an den Zahlen ablesen kann, an welcher Position seine Kugel landen wird. Um es dem Spieler noch eindeutiger zu machen, heißt die x-Achse Zeilen, die y-Achse Spalten und die z-Achse Höhe. Das ist besonders wichtig, wenn es möglich ist den Plot zu drehen, weil man sonst durcheinander kommt.

2.2.2 Erstellung der Gewinnbedingung

Der Gewinner des Spiels wird mit Hilfe einer while-Schleife mit einem zusätzlichen break-Befehl bestimmt. Die while-Schleife läuft solange, bis Schwarz gewonnen hat, denn die Abbruchbedingung muss Schwarz lauten, damit die Schleife aufhört. Allerdings muss die Schleife auch abgebrochen werden, wenn Weiß gewonnen hat und das nach dem Zug von Weiß. Um das zu gewährleisten, ist in die while-Scheife eine if-Bedingung eingebaut, die sofort abbricht, sobald Weiß gewonnen hat.
Die Funktionen *weiß*, *schwarz* und *Gewinner* werden jeweils in ihrem eigenen Abschnitt näher beleuchtet.
Mit dem Wert von Abbruchbedingung wird als Letztes der Sieger der Partie ausgegeben.

2.3 Funktionen Weiß und Schwarz

Die Funktionen *weiß* und *schwarz* ermöglichen dem Spieler seine Spielsteinposition anzugeben.
Die beiden Funktionen hängen vom Wert eingabe ab, der in der Funktion *start* definiert ist . Der Wert eingabe enthält einen input-Befehl, der es dem Spieler ermöglicht, die Position seiner Kugel einzugeben, in dem er als erstes die Zeile und durch ein Komma getrennt die Spalte angibt.
Das ist Teil 1 des Versuches, das Spiel realistisch zu machen, denn durch diese Methode, kann der Spieler nur den Stab wählen, wie auch beim nicht digitalen Vorbild.
Die Funktion *weiß* wandelt die Eingabe in den Zahlenwert Integer um. Diese Integers werden genutzt, um an die angegebene Stelle eine 1 in die Matrix A zu setzten. Anschließend sorgt die Funktion *Gravitation* dafür, dass die Kugel nicht schwebt, wie das genau funktioniert, wird im entsprechenden Abschnitt erläutert.
Als letzten Schritt trägt die Funktion *weiß* die Werte von Zeile, Spalte und Höhe in die dafür vorgesehenen Listen ein. Aus diesen Listen werden in den Funktionen *Plotweiß* und *Plotschwarz* die Koordinaten für den Plot ausgelesen.
Das gleiche passiert auch in der Funktion *schwarz* jedoch mit dem Unterschied, dass die Stelle in der Matrix A auf 2 gesetzt wir.

2.4 Funktion Gravitation

Diese Funktion ist Teil 2 des Versuchs, das Spiel realistisch zu machen, denn aus Teil 1 (Man wählt nur den Stab), ergibt sich ein Problem.
Die Spieler geben dem Programm nur zwei Informationen, nämlich die Zeile und die Spalte, aber es fehlt noch die Ebene, damit das Programm eine eindeutige Position findet.
Die Funktion *Gravitation* testet, ob sich eine Ebene tiefer (Der Matris darunter) an der gleichen Stelle eine 0 befindet, wenn ja, tauschen 0 und der Wert die Plätze. Dieser Vorgang wird wiederholt, bis der Wert darunter nicht mehr 0 ist oder die unterste Ebene erreicht ist. Realisiert wird dieser Vorgang durch eine Treppe von if-Bedingungen und else-Bedingungen.
Dadurch wird gewährleistet, dass die Kugeln im Plot nicht schweben, was in der Realität nicht funktionieren würde.
Außerdem gibt die Funktion *Gravitation* als Output 3 Werte an, nämlich die Höhe, die Zeile und die Spalte, diese Werte sind wichtig, damit der 3D-Plot möglich ist.

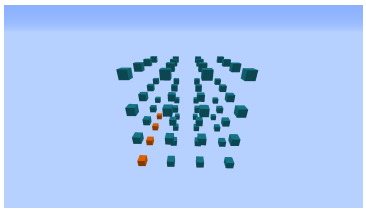
2.5 Plot

Die Position der 1, die nach dem Ausführen der Funktion *Gravitation*, feststeht, soll sich im Plot wiederfinden. Dazu braucht man die Funktion *Plotweiß*, die mit Hilfe der Positionsdaten der 1 in der Matrix eine Kugel in den Grundplot plottet.
Damit das funktioniert, existieren die Listen Zeileweiß, Spalteweiß und Höheweiß. Diese Listen werden von der Funktion *weiß* gefüllt. Sie fungieren als x, y und z Koordinaten für den Plot, die als Scatterplot von der Funktion *Plotweiß* mit roten Punkten dargestellt werden.
Des Weiteren wird in ihr das Subplotfenster definiert und die Achsen beschriftet.
Die Funktion *Plotschwarz* agiert genau gleich, allerdings liest sie die Daten aus den Listen Zeileschwarz, Spalteschwarz und Höheschwarz aus, die von der Funktion *schwarz* gefüllt werden. Diese Werte finden sich als blaue Kugeln im Plot wieder.

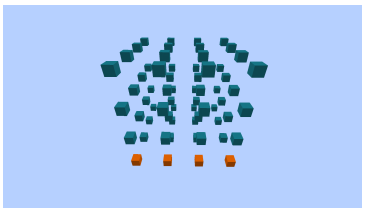
2.6 Funktion Gewinner

Gewonnen ist das Spiel, wenn man in egal welcher Konstellation vier Kugeln einer Farbe in einer Reihe hat. Das heißt, wenn vier gleichfarbige Steine senkrecht übereinander, waagerecht nebeneinander, diagonal in einer Ebene oder aber auch diagonal nach oben oder von Ecke zu Ecke gesetzt werden.
Daraus folgt die Vorgehensweise der Funktion. Zuerst werden Zeilen und Spalten jeder Matrix abgesucht, wie in Bild 1 und Bild 2. Anschließend werden die Diagonalen , in Bild 3, Bild 4, Bild 12 und Bild 13 zu sehen, überprüft. Als nächstes werden die Flächen untersucht, in Bild 5, Bild 6 und Bild 7 dargestellt. Als letztes werden die 4 Diagonalen , Bild 8 bis Bild 11, von der Funktion beachtet.
Dieses Verfahren wird verwirklicht, indem sich alle Gewinnerkombinationen in der Gewinnerliste befinden, die innerhalb der Funktion *Gewinner* definiert wird. Anschließend prüft die Funktion *Gewinner* jede Stelle der Gewinnerliste.
Findet sie auf diese Weise vier zusammenhängende Werte,das heißt die Liste [1,1,1,1] oder [2,2,2,2], gibt sie einen String aus. Dieser String (entweder Weiß oder Schwarz) setzt in der Funktion *start* die Abbruchbedingung und beendet so das Spiel. Außerdem wird mit Hilfe dieses Strings der Sieger bekanntgegeben. Das wird durch eine einfache if-Bedingung ermöglicht.
Wenn nicht, darf der nächste Spieler seine Position eingeben.

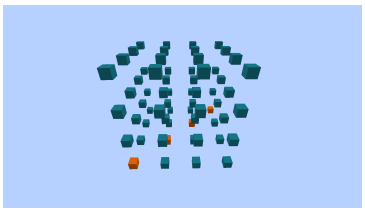
2.6.1 Bilder



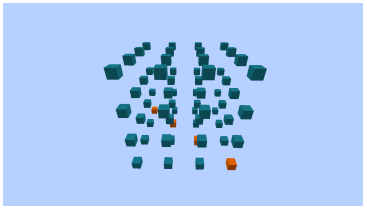
1



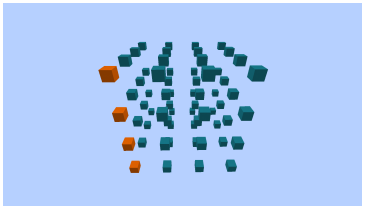
2



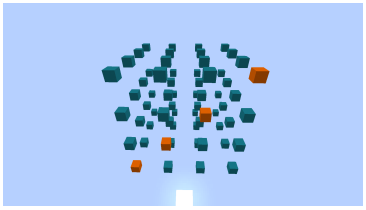
3



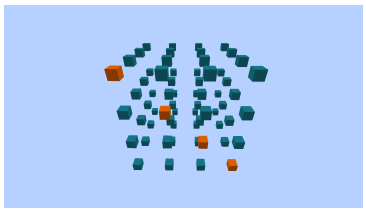
4



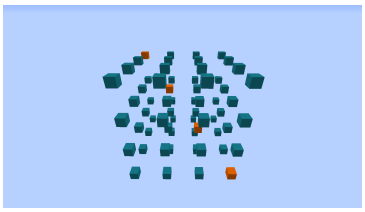
5



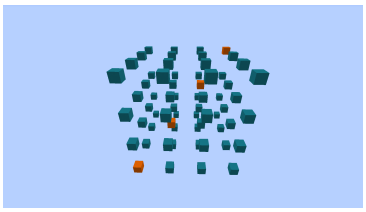
6



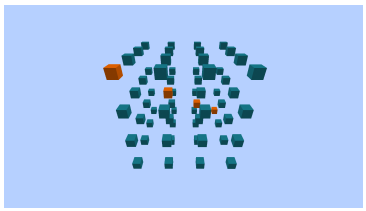
7



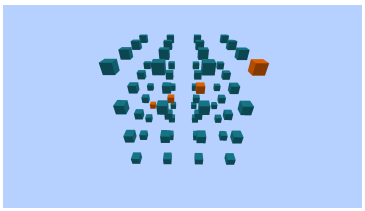
8



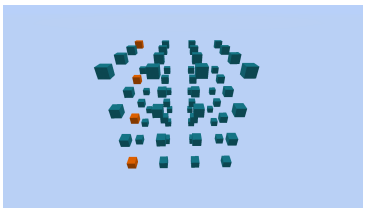
9



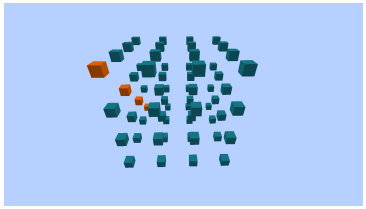
10



11



12



13

3 Erläuterung Gesamt(Beta).py

3.1 Technische Daten

	Name	Version	3D Plot
Betriebssytem	Windows 10 Home	10.0.16299 Build 16299	—
Läuft auf	Spyder	3.2.5	—
	IPython	3.61 Anaconda costum (64-Bit)	—
	Jupyter Notebook	5.2.2	—
Browser für Jupyter	Firefox Quantum	59.0.1 (64-bit)	—

3.2 Grundidee

Für die Betaversion war es das Ziel nur den Grundstein zu legen, das heißt, sie sollte nur den blanken Code enthalten. Das Spiel sollte spielbar sein, allerdings noch ohne Grafik und der Code sollte vorrangig funktionieren, es wurde also keine Rücksicht auf effiziente Programmierung genommen.

3.3 Spielweise

In dem Programm sind viele Variablen global definiert, um so Fehlern, die mit unbekannten Variablen verbunden sind, zu vermeiden. Weiterhin kann das Spiel nur zu zweit gespielt werden.
Den Spielern wird nach dem Ausführen des Codes eine Spielanleitung ausgegeben. Anschließend kann das Spiel gestartet werden.
Der erste Spieler wird aufgefordert seine erste Position zu wählen, das tut er wie oben genannt. Seine eingebenen Werte werden von der Funktion *weiß* in Integers umgewandelt. Diese beiden Zahlen geben dann eine Position in der Matrix A an und setzten diese auf 1. Allerdings ist die Matrix A die höchste Matrix und die gewählte Position hängt in der Luft. Jetzt nimmt sich die Funktion *Gravitation* die beiden Zahlenwerte und testet die drei weiteren Matrizen an dieser Stelle. Da es der erste Zug ist, befinden sich in den anderen Matrizen an der Stelle nur Nullen. Deswegen tauscht die Funktion *Gravitation* die 1 immer mit der 0 darunter, bis die 1 in der Matrix D, also der untersten, angekommen ist. Als Output gibt sie an die Funktion *weiß* die Matrix und die eingegebenen Werte zurück, in die die 1 gesetzt wurde. Dieser Output landet direkt im Output von Funktion *weiß*, denn diese druckt alle 4 Matrizen einfach aus.
Anschließend wird eine Abbruchbedingung definiert mit Hilfe der Funktion *Gewinner*. Diese Funktion enthält eine Gewinnerliste, in der sich alle Gewinnmöglichkeiten befinden. Die Funktion *Gewinner* vergleicht jetzt die Gewinnmöglichkeiten aus der Liste mit den gesetzten Positionen in den Matrizen. Findet sie so eine Gewinnerkombination in den Matrizen gibt sie einen String zurück, der dann als die oben genannte Abbruchbedingung definiert wird. Da bis jetzt nur ein Zug gespielt wurde, hat natürlich noch niemand gewonnen, das heißt die Abbruchbedingung ist nicht erfüllt und die while-Schleife läuft weiter.
Nun kommt der zweite Spieler an die Reihe und kann ebenfalls seine Position eingeben. Sein Wert wird genauso behandelt wie der Wert vom ersten Spieler, allerdings hinterlässt der zweite Spieler eine 2 anstatt einer 1 an seiner Position in den Matrizen.
Die Spieler können nun in den Matrizen sehen, wo sie ihre Zahl gesetzt haben, und spielen bis es einer schafft, 4 seiner Zahlen in eine Reihe zu bekommen.
Ist das geschafft, erkennt die Funktion *Gewinner* eine Gewinnkombination in der Gewinnerliste und verändert den Wert der Abbruchbedingung. Der geänderte Wert der Abbruchbedingung führt zum Abbruch der while-Schleife, die es ermöglicht einen Zug nach dem anderen zu tun. Mit dem Wert der Abbruchbedingung, kann die Funktion *start* mit einem String als Output den Sieger verkünden.

3.4 Probleme

- Das Spiel hat keine Exitbedingung, sodass man zu Ende spielen oder absichtlich etwas Falsches eingeben muss, um aus dem Loop auszubrechen.
- Das Spiel hat keine Resetfunktion, dass heißt, möchte man eine weitere Runde spielen, muss man den gesamten Code wieder durchlaufen lassen.
- Der grafische Output mit den Matrizen ist nicht sehr eindeutig, sodass das Erkennen, welche Position günstig wäre, erschwert ist.
- Bevor der erste Zug gemacht wird, sehen die Spieler das Spielfeld nicht.

3.5 Video

In der Videodatei Gesamt(Beta) befindet sich eine Erläuterung der Spielfunktionen, die die erste spielbare Version beinhaltet.

4 Erläuterung Gesamt(Beta 1.0).py

4.1 Technische Daten

	Name	Version	3D Plot
Betriebssytem	Windows 10 Home	10.0.16299 Build 16299	—
Läuft auf	Spyder IPython Jupyter Notebook	3.2.5 3.61 Anaconda costum (64-Bit) 5.2.2	unbeweglich beweglich unbeweglich
Browser für Jupyter	Firefox Quantum	59.0.1 (64-bit)	—

4.2 Grundidee

Ziel dieser Version war es, in den Code der Betaversion den 3D Plot zu integrieren.

4.3 Spielweise

In dem Programm sind viele Variablen global definiert, um so Fehlern, die mit unbekannten Variablen verbunden sind, zu vermeiden. Weiterhin kann das Spiel nur zu zweit gespielt werden.
Gespielt wird wie oben beschrieben, allerdings werden mit den eingegebenen Werten zusätzliche Dinge gemacht.
Die Funktionen *weiß* und *schwarz* agieren genauso wie in der Betaversion, haben aber einen zusätzlichen Schritt, bei dem die eingegebenen Werte in zwei unterschiedlichen Listen geschrieben werden. Auch in dieser Version nimmt die Funktion *Gravitation* die Zeile und die Spalte auf und testet, ob sich die Matrix ändern muss. Der Output enthält in dieser Version anstatt der Matrix den Zahlenwert h. Dieser wird wiederum von den Funktionen *weiß* und *schwarz* in die entsprechende Liste gepackt, sodass für jeden Spieler 3 Listen existieren.
Jeweils nach den Eingabefunktionen folgen die Plotfunktionen in der while-Schleife. Logischer Weise gibt es zwei, die Funktion *Plotweiß* und die Funktion *Plotschwarz*. Diese beiden Funktionen definieren das Plotfenster und legen einen Subplot fest, in den beide ihre Werte plotten. In ihnen werden auch die Maße des Plots und seine Beschriftung festgelegt. Um die Positionen in den Plot zeichnen zu können, greifen diese Funktionen auf die Listen zurück, wobei sie mit Farben unterscheiden aus welchen 3 Listen sie plotten. So werden die Daten vom ersten Spieler in rot und die vom zweiten Spieler in blau geplottet.
Der Rest entspricht der Spielweise von der Betaversion.

4.4 Probleme

- Das Spiel hat keine Resetfunktion, dass heißt, möchte man eine weitere Runde spielen, muss man den gesamten Code wieder durchlaufen lassen.
- Bevor der erste Zug gemacht wird, sehen die Spieler das Spielfeld nicht.
- Je länger ein Spiel geht, desto unübersichtlicher wird das Spielfeld.

5 Erläuterung SOGO.py

5.1 Technische Daten

	Name	Version	3D Plot
Betriebssytem	Windows 10 Home	10.0.16299 Build 16299	—
Läuft auf	Spyder IPython Jupyter Notebook	3.2.5 3.61 Anaconda costum (64-Bit) 5.2.2	unbeweglich beweglich unbeweglich
Browser für Jupyter	Firefox Quantum	59.0.1 (64-bit)	—

5.2 Grundidee

In der endgültigen Version sollte es keine globalen Variablen mehr geben, um den damit verbundenen Problemen aus dem Weg zu gehen.

5.3 Spielweise

Das Spiel wird genauso gespielt wie auch in der Gesamt(Beta 1,0). Die Abläufe in dem Programm sind ebenfalls die selben, es gibt nur ein paar kleinere Veränderungen.
Die Matrizen und die Listen werden lokal in der Funktion *start* definiert. Damit beginnt mit jedem starten des Spiels ein komplett neues, ohne dass man den gesamten Code durchlaufen lassen muss.
Wenn das Spiel gestartet wird, können die Spieler noch bevor sie einen Zug gemacht haben, das Spielfeld sehen. Dadurch ist das Spiel leichter verständlich.
Außerdem wurden die Textausgaben angepasst und die Spielanleitung verbessert.

5.4 Probleme

- Je länger ein Spiel geht, desto unübersichtlicher wird das Spielfeld.

5.5 Video

In der Videodatei SOGO befindet sich eine Darstellung des fertigen Spiels.