Dokumentation Projektarbeit - GameOfLife

Groth, Frederick

20. März 2016

Inhaltsverzeichnis

1 Ziel des Projektes **3**

2 Durchführung **3**

2.1 prepGoL 3

2.2 GoLdataimport 3

2.3 GoLStep 3

2.4 GoLplot 4

1 Einleitung

2 Ziel des Projektes

Ziel des Projektes war die Programmierung des GameOfLife in Python. Dies bedeutet, dass an Hand vom Benutzer vorzugebener Regeln die Entwicklung des Feldes berechnet und dargestellt werden soll. Ferner wurden einige Sonderfunktionen programmiert, welche dem Benutzer weitere Einstellungen ermöglichen.

3 Durchführung

Das Projekt wurde in mehrere Arbeitsschritte eingeteilt. Zunächst wurde GitHub als Versionskontrollsystem eingerichtet, um das Projekt zu sichern und den Verlauf zu dokumentieren. Anschließend begann die Programmierarbeit mit dem Erstellen der Grundfunktionen, die für das Programm nötig sind. Ich entschied mich dafür, die Strukturen von Numpy zu verwenden, insbesondere also arrays und den zugehörigen Operationen zu arbeiten. Die genaue Funktionsweise ist dem Programm und insbesondere den Funktionsbeschreibungen zu entnehmen. Hier soll nur ein kleiner Einblick in die vorangegangenen Überlegungen gegeben werden.

3.1 prepGoL

Ein wichtiger Aspekt des Programms sollte die Vorgabe einiger Bedingungen vom Nutzer sein. Dazu zählen der Regelstring, welcher den Verlauf bestimmt, die Randbedingung, Systemgröße, sowie die Anfangsbedingung. Zu diesem Zweck schrieb ich die Funktion prepGoL. Diese verlangt genau diese Eigenschaften als Argumente, welche in verschiedenen Formaten vorgegeben werden. Die erste Idee war, alle Argumente in eine Text-Datei zu schreiben. Dies wäre allerdings beim Datenimport deutlich komplizierter gewesen, da die verschiedenen Argumente verschiedene Variablen-Typen benötigen. Die Funktion prepGoL erstellt also mehrere Text-Dateien, welche noch einmal verändert werden können. Jede Datei enthält ein Argument. Die Dateien werden zum Ausführen der Funktion GameOfLife in genau der vorgegeben Form benötigt.

3.2 GoLdataimport

Zur weiteren Verwendung müssen die Daten wieder als Variablen importiert werden. Dies geschieht mit der Funktion GoLdataimport, welche eine Null-Matrix der vorgegebenen Größe erzeugt. Sie wird an den in der Anfangsbedingung vorgegebenen Stellen mit 1en gefüllt. Diese stehen im weiteren Verlauf für lebende Zellen, 0en für tote/nicht vorhandene Zellen. Das nun fertige Anfangsfeld wird gemeinsam mit der Randbedingung zurückgegeben.

3.3 GoLStep

Nun war also die Vorbereitung für den Import der Benutzervorgaben abgeschlossen. Ich habe mich anschließend der Berechnung der folgenden Matrix zugewandt. Dazu muss zunächst einmal die Anzahl der Nachbarzellen jeder Zelle berechnet werden. Da ich mich, wie schon erwähnt, für die Arbeit mit Matrizen, genauer numpy arrays, entschieden hatte, bot sich hierfür eine Matrixmultiplikation (M.dot(M1)) an. Nach näher Betrachtung ergab sich, dass die Multiplikation mit Matrizen mit 1en auf der Hauptdiagonale sowie den beiden nächsten Nebendiagonalen das gewünschte Ergebnis lieferte. Auf der Linken Seite erwirkt dies eine Addition der untereinander liegenden Werte, auf der rechten Seite eine Addition der nebeneinander liegenden. Es muss schließlich noch die Feld-Matrix abgezogen werden, damit die Zellen selbst nicht mitgerechnet werden.

Somit lässt sich der Verlauf für eine begrenzte Randbedingung berechnen, bei der die Außenbereiche als tot angenommen werden.

Später habe ich die Funktion für andere Randbedingungen ausgeweitet. Dazu gehören noch ein torusförmiges bzw. periodisches und ein unendliches Feld. Für eine torusförmig gekrümmte Fläche, oder wie ich sie genannt habe, periodische, müssen in den beiden Matrizen noch 1en in den Ecken ergänzt werden.

Die Berechnung für eine unendliche Randbedingung gestaltet sich etwas schwieriger. Hier muss die Größe der Matrix dynamisch angepasst werden. Ich entschied mich dafür, die Matrix nur zu erweitern, nicht jedoch zu kürzen, falls in den Randbereichen keine lebenden Zellen sein sollten. Es wird also geprüft, ob sich in den Randbereichen lebende Zellen aufhalten. Dies geschieht ebenfalls mit Matrixmultiplikationen.

Ist dies der Fall, wird eine weitere Zeile oder Spalte an der entsprechenden Seite ergänzt.

3.4 GoLplot

Da nun die Ausgangsmatrix sowie eine Funktion zum Berechnen der weiteren Schritte vorhanden war, habe ich mir überlegt, wie ich die Veränderungen animiert darstellen kann. Nach einer Suche auf den matplotlib Seiten stieß ich auf die Funktion FuncAnimation von matplotlib.animation. Diese erschien mir für meine Zwecke geeignet und so erarbeitete ich mir aus den Beispielen und der Dokumentation die Funktionsweise. Dies nutzte ich dann, um die Matrix animiert als imshow auszugeben. Als Parameter müssen dabei einerseits die Geschwindigkeit, Anzahl der Schritte, Wiederholungen und die Figur, in der geplottet wurde, übergeben werden. Andererseits wird eine Funktion benötigt, welche den Verlauf vorgibt. Dies gestaltete sich als schwerster Teil, da sie nur von der Schrittnummer abhängen darf und sie die Figur wieder zurückgeben muss.

Zunächst einmal musste gespeichert werden, wie die letzte Matrix aussah. Um die Geschwindigkeit bei Wiederholungen zu erhöhen, werden alle Felder in einer Liste gespeichert und abgerufen. Sollte der Wert noch nicht in der Liste vorhanden sein, wird er aus dem vorhergehenden mit der Funktion GoLStep berechnet.

Eine weitere Schwierigkeit trat auf, als ich die Darstellung aktualisieren wollte, wobei eine Veränderung der dargestellten Werte nicht ausreichte. Bei der unendlichen Rand-

bedingung wird nur ein Ausschnitt der Anfangs-Größe angezeigt. Meine Lösung war schließlich die immer neue Erzeugung der imshow, nachdem der Darstellungsbereich geleert wurde.

Abschließend fügte ich noch eine wie i fortlaufende Variable hinzu, welche Pausen und Stopps möglich macht.

Die ebenfalls in der Funktion GoLplot vorhandenen Sonderfunktionen werden in einem späteren Abschnitt besprochen.

3.5 GameOfLife

Als Abschluss fügte ich noch die Funktion GameOfLife hinzu, welche, nach der Ausführung von prepGoL oder einem manuellen Erstellen der Text-Dateien die, alle weiteren Funktionen aufruft und so den Ablauf vom GameOfLife ermöglicht.

Nach dem Datenimport werden die Variablen an GoLplot übergeben, sodass die Animation startet.