Analysieren und Visualisieren mit Python

zweite Übungsaufgabe

Marvin, der Sohn Ihrer Nachbarn, hat in diesem Semester angefangen, Biologie zu studieren. Um sich auf sein Studium angemessen vorzubereiten, sitzt er nun häufig im Garten der Eltern und erforscht die ansässigen Ameisen. Da wir uns im Rahmen dieses Seminars aktiv für den Tierschutz einsetzen, sollen Sie zur Bearbeitung dieser Aufgabe einen Ameisensimulator programmieren, der unserem Freund zur Verfügung gestellt werden kann.
Hierfür sollen Sie *Conway's Game of Life* implementieren. Sehen Sie sich vorab den Wikipedia-Artikel https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens an (bei schönem Wetter reichen die ersten beiden Abschnitte). Damit Sie sich auf die Umsetzung des Algorithmus konzentrieren können, finden Sie in der Datei *gol.py* eine Nutzerschnittstelle, mit der sich Ameisen platzieren lassen und mit der Sie Ihre Implementierung testen können. Zur Bearbeitung dieser Aufgabe müssen Sie somit ausschließlich die Datei *golalgo.py* erweitern, beim Start von *gol.py* wird diese dann importiert.

1a) (7 Punkte)
In der Datei *golalgo.py* sollen Sie nun die Funktion *simpleGol(antArray)* erweitern und die klassischen Regeln umsetzen (*23*/*3*-Welt):

- Zellen mit zwei oder drei Nachbarn bleiben erhalten (=> Zellen mit mehr als drei oder weniger als 2 Nachbarn werden also gelöscht).
- Auf leeren Zellen mit genau drei Nachbarn entsteht eine neue Ameise.

Das aktuelle Spielfeld (beliebiger Größe!) wird in der dreidimensionalen Liste *antArray* übergeben (siehe Abbildung 1). Dabei steht die erste Dimension für die Zeile, die zweite für die Spalte und die dritte Dimension für die Ausprägung der Grundfarben rot, grün und blau. [255,255,255] steht also für einen weißen (unbesetzten) Pixel und [0,0,0] für einen schwarzen (Ameisen-)Pixel.

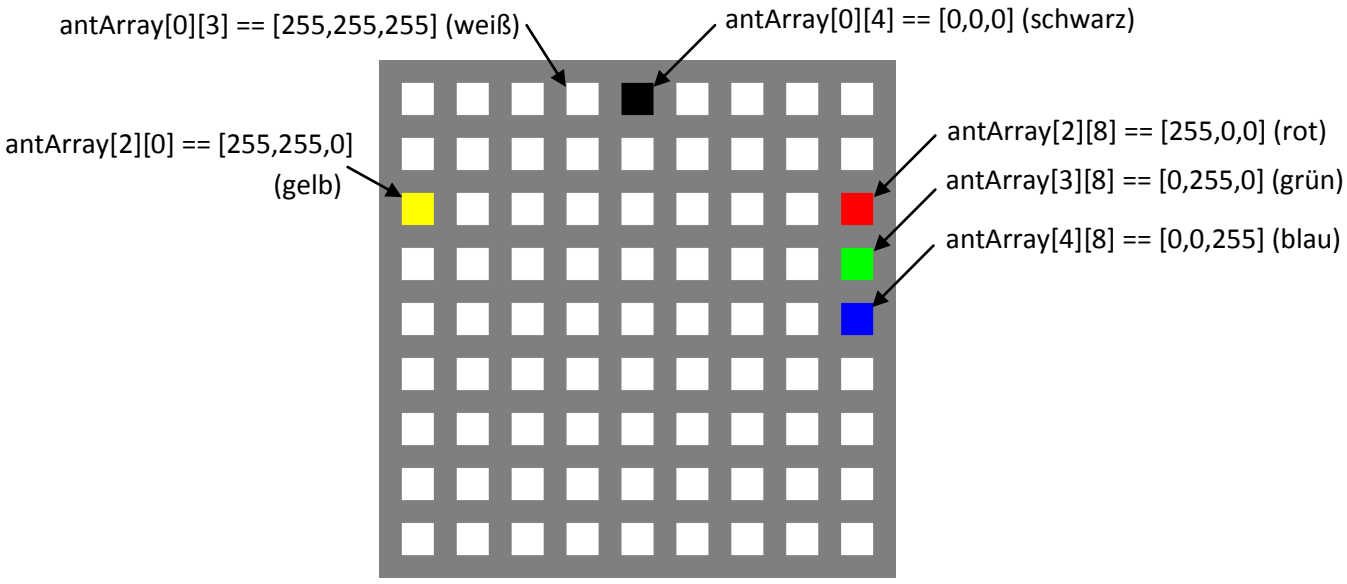

Abbildung 1: Visualisierung der in antArray übergebenen Liste

Für diesen Aufgabenteil reicht es, schwarze (Ameise) und weiße Pixel (keine Ameise) zu unterscheiden. Die Funktion *simpleGol* soll ausgehend vom aktuellen Spielfeld *antArray* genau eine Generation berechnen und das Ergebnis als neue dreidimensionale Liste derselben Form zurückgeben (siehe Abbildung 2).

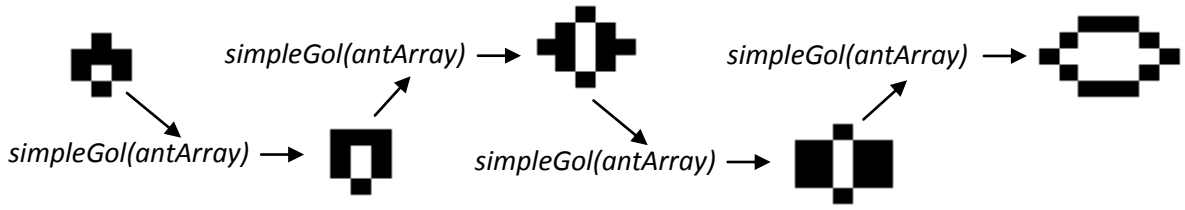

Abbildung 2: Beispiel für fünf Generationen einer Ameisenkolonie

Da unser Spielfeld begrenzt ist, soll davon ausgegangen werden, dass abseits des Spielfeldes kein Leben möglich ist. Die Zelle *antArray[2][0]* kann also beispielsweise maximal fünf Nachbarn haben.

1b) (3 Punkte)

Obwohl Marvin Sie versichert, dass Ihre Ameisensimulation sehr hilfreich für sein Studium sein wird, bemängelt er die beschränkten Interaktionsmöglichkeiten. Im Speziellen hat Marvin geplant, in einem finalen Schlag vier Farbeimer, die er in der Garage gefunden hat, über die inzwischen nachgewachsenen Hügelnester zu verteilen. Retten Sie die Ameisen, indem Sie die Funktion *multicolorAnts* analog zum Aufgabenteil a) erweitern und eine 1234/23-Welt generieren:

- Zellen mit eins, zwei, drei oder vier Nachbarn bleiben erhalten (=> Zellen mit mehr als vier oder mit keinem Nachbarn werden also gelöscht).
- Auf leeren Zellen mit zwei oder drei Nachbarn entsteht eine neue Ameise.

Färben Sie besetzte Zellen dabei in Abhängigkeit der Anzahl der Nachbarn unterschiedlich ein. Verwenden Sie hierfür Ihre Lieblingsfarben (weiß dient zur Kennzeichnung unbesetzter Zellen und stellt deshalb keine gültige Farbe dar). Die Zählung der Nachbarn soll unabhängig von der Farbe der Nachbarameisen erfolgen. Eine Zelle gilt also als besetzt, wenn Ihr Wert ungleich [255,255,255] ist.

Gegebenenfalls müssen Sie zusätzlich die Erweiterungen *Pillow*, *TkInter* und *NumPy* installieren. Nutzen Sie bitte das Nachrichtenforum auf Grips, um sich über auftretende Probleme in verschiedenen Betriebssystemen auszutauschen.

Ergänzen Sie zur Lösung der folgenden Aufgaben die Datei *golalgo.py* und reichen Sie sie über GRIPS bis zum 29.11.17 um 23:55 Uhr ein. Bei Problemen mit der Abgabe können Sie Ihre Lösung bis zur genannten Zeit auch an manuel.ullmann@ur.de senden.

Bitte geben Sie ausschließlich die (unverpackte) Datei golalgo.py ab.

Ihre Abgaben werden automatisiert mit den Abgaben der anderen Teilnehmer und gegebenenfalls mit den Abgaben der letzten Semester verglichen. Auch bei abgeänderten Variablenamen werden gleiche Lösungsansätze von der verwendeten Software erkannt. Im Falle eines Plagiats bekommen alle Gruppenmitglieder die Aufgabe als Plagiat bewertet, unabhängig davon, ob die Lösung weitergegeben oder übernommen wurde. Arbeiten Sie deshalb (nur) gemeinsam mit Ihren Übungspartnern an den Aufgaben. Falls Sie Ihre Übungspartner nicht erreichen, schreiben Sie mir eine E-Mail, sodass wir die Gruppeneinteilung anpassen können. Sollten Sie Code-Blöcke aus dem Internet übernehmen, kennzeichnen Sie die Quellen nachvollziehbar in den Kommentaren Ihrer Abgabe. Falls Sie bei der Bearbeitung der Aufgabe auf Probleme stoßen, können Sie die Übungstermine nutzen und dort Ihre Fragen stellen.

Falls Sie auch gerne Ameisen erforschen, können Sie durch Abändern der Parameter noch zahlreiche andere interessante Ameisenwelten erschaffen!

Viel Erfolg!