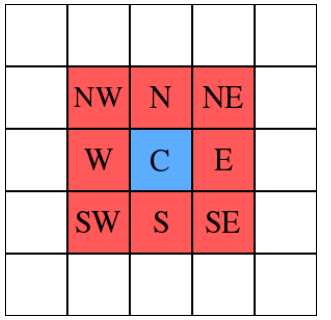
Conway’s Game of Life

Nach der Lernplattform jython.ch

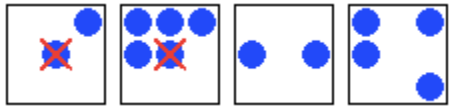
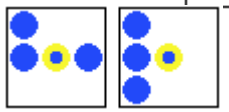
Concept



- 1970 wurde das Spiel vom englischen Mathematiker John Horton Conway (* 26. Dezember 1937 in Liverpool) erfunden.
- Das Spiel basiert auf einem zweidimensionalen zellulären Automaten.
- Zelluläre Automaten dienen der Modellierung räumlich diskreter dynamischer Systeme.
- Die Entwicklung einzelner Zellen zum Zeitpunkt (*t+1*) hängt primär von den Zellzuständen in einer vorgegebenen Nachbarschaft und vom eigenen Zustand zum Zeitpunkt (*t*) ab.
- Es wird eine sogenannte Moore-Nachbarschaft (8er-Nachbarschaft) verwendet.

The 4 rules

Zu Beginn werden zufällig 1000 lebende Zellen bestimmt (grün).

1. Jede lebende Zelle, die keinen oder nur einen Nachbarn hat, stirbt (an Einsamkeit).
2. Jede lebende Zelle, die 4 oder mehr Nachbarn hat, stirbt (an Überbevölkerung). 
3. Jede lebende Zelle, die 2 oder 3 Nachbarn hat überlebt. 
4. Eine tote Zelle, die 3 Nachbarn hat, wird lebendig. 

In jedem Simulationszyklus wird für jede Zelle die Anzahl der Nachbaren bestimmt und die Zelle als lebend (grün) oder als tot (schwarz) markiert.

Gamegrid

- *from gamegrid import **
- Neue Klasse *Game* die aus der Oberklasse *GameGrid* abgeleitet wird.
- *GameGrid.__init__(self, s, s, 800 // s, Color.gray)* äquivalent zu: *makeGameGrid(#x, #y, #Pixel, Linienfarbe, Hintergrundbild, Navigationsbalken)*
- Variable *s=50* (=50 Zellen in jede Richtung)
- Was bedeutet der Operator: // ?
- *Game()*

Reset-Button

- 2 neue Variablen: *z* für die Anfangsbevölkerung und *a* als Vergleichsvariable.
- Neue Methode: *reset(self)*: Hier sollen 1000 lebende Zellen zufällig grün gefärbt werden so dass jedes Mal beim reset-event ein neuer Anfangszustand generiert wird.
- Zuerst werden alle Zellen schwarz gefärbt (Ursprungszustand: *a=0*).
- Danach werden 1000 zufällige Zellen ausgewählt (*a=1*).
- Die Population wird mit der Methode *showPopulation(self)* angezeigt, welche zuerst noch definiert werden muss.
- *if else loop:* für alle *a[x][y]==1* wird die Zelle grün für alle anderen schwarz gefärbt.

Number of Neighbours

- Neue Methode: *getNumberOfNeighbours(self, x, y)* mit neuer Variable *nb=0*.
- Randbedingungen festlegen (damit keine Zellen außerhalb des Fensters betroffen sind) mit *max()* und *min()*.
- Neue integers für x-und y-Richtung (*i* und *k*).
- *If not* Bedingung: *i* und *k* dürfen nicht *x* und *y* sein (sondern Nachbarn von x und y).
- *if* Bedingung: Wenn *a[i][k]==1* dann handelt es sich um einen Nachbarn und die Variable *nb* wird um eins erhöht.

Applying the 4 rules

- Neue Methode: *act(self)*
- Neue Variable *b* (anfänglich äquivalent zu *a*)
- *x* und *y* müssen sich wieder im angezeigten Fenster befinden.
- Die Anzahl Nachbarn wird von der vorherigen Methode übernommen und mit den Lebenden und toten Zellen verglichen (*if else*).
- Die Variable *b* wird je nachdem ob die Kondition zutrifft mit *1* (lebend) oder *0* (tot) gleichgesetzt. Um die vorherigen Schritte zu wiederholen, muss *a* mit *b* gleichgesetzt werden.
- Durch das Ausführen der Methode *self.showPopulation()* werden die neuen Zellen generiert und der Zyklus kann wieder von vorne beginnen.

Objects

https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

Interpretations

- Ökonomisch: Man kauft ein Produkt, wenn einige, aber nicht zu viele und nicht zu wenige Nachbarn es ebenfalls bereits besitzen.
- Biologisch: Die Zellen können wie «lebende» Zellen interpretiert werden.
- Chemisch: Die unterschiedlichen Objekte können mit unterschiedlichen Energieniveaus gleichgesetzt werden (Aufwand an Energie um ein bestimmtes Edukt zu erhalten).
- Physikalisch: Rein deterministisch können (beliebig) kleine Abweichungen der Startbedingung zu ganz unterschiedlichen Ergebnissen führen. Somit lässt sich ein Anfangswertproblem formulieren, worauf chaotisches Verhalten folgt.