Robert Salvador salvador@htlinn.ac.at

Conway´s "Game of Life"

- Mathematische / Fachliche Inhalte in Stichworten

  Matrix- und Zeichenketten-Manipulationen
  Programmieren mit der Mathcad-Programmierumgebung

- Kurzzusammenfassung

  Verschiedene Varianten von Conway´s "Game of Life" werden mit Mathcad realisiert.

- Lehrplanbezug (bzw. Gegenstand / Abteilung / Jahrgang)

  Der Schwerpunkt der Arbeit liegt eher im Informatik- als im Mathematik-Bereich.
  Mit einigermaßen interessierten Schülern kann das "Game of Life" in C(++) in Informatik im 2. Jahrgang (Konsolen-Programm) und/oder grafisch anspruchsvoller mit C# oder Java im 3. Jahrgang programmiert werden.
  Die vorliegende Arbeit soll zeigen, dass es die von Mathcad gebotenen Möglichkeiten erlauben, das Game of Life zu "spielen".

- Mathcad-Version

  Mathcad 14

- Literaturangaben

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

ÜBERSICHT (Doppelklick auf den gewünschten Bereich!):

Einleitung

zurück zur Übersicht

Wer kennt ihn nicht, den evolutionären - und immer noch populären - Algorithmus-Klassiker des Mathematikers *John Horton Conway* aus dem Jahr 1970?
Die Bezeichnung "Spiel" ist allerdings etwas irreführend (wenn man sich auch ausgiebig damit "spielen" kann!), handelt es sich doch eher um eine Simulation!

Die vorliegende Arbeit soll zeigen, wie man mit Mathcad die von Conway (und nach ihm vielen anderen) behandelte Aufgabe angehen könnte.

Details sind auf der angegebenen Internet-Seite (siehe Link auf der Startseite dieser Arbeit!) zu finden. Es soll daher hier nur das Wesentliche ganz kurz erläutert werden:

Die Startsituation
Ein rechteckiger Bereich ist (entsprechend den Zeilen und Spalten einer Matrix) in Felder aufgeteilt. Jedes Teilfeld ist von einer Zelle (= "lebende Zelle") besetzt oder nicht (= "tote Zelle"). Ich bezeichne diese Matrix als *Populationsmatrix*. Lebende Zellen werden in dieser Arbeit durch eine 1 in der Populationsmatrix beschrieben, tote Zellen durch eine 0.

Die Entwicklung
Abhängig von bestimmten Regeln (Überlebens-, Todes- und Geburts-Regeln), welche die Anzahl der lebenden Nachbarn eines Feldes betreffen, ändert sich im allgemeinen die Population von einer Generation zur nächsten. Dabei wird jeweils zunächst die Zahl aller lebenden Nachbarn aller Felder der aktuellen Population berechnet (= *Nachbarn-Matrix*) und auf dieser Basis - durch Anwendung der erwähnten Regeln - die nächste Generation (die nächste Populationsmatrix).

Der Spielfeld-Rand
Was den Rand des "Spielfelds" betrifft, gibt es im wesentlichen zwei Möglichkeiten:
- Der Rand bestimmt das Ende des Zellenbereichs, d. h. außerhalb gibt es nichts.
- Das Spielfeld ist horizontal und vertikal "endlos", d. h. linke und rechte Kante sowie obere und untere Kante grenzen aneinander. Es gibt also eigentlich gar keinen Rand, und die Zellenpolpulation lebt in Wirklichkeit nicht in einem Rechteck, sondern auf einem Torus.

Für diese Arbeit wurde die "Torus-Variante" gewählt. Das bedeutet etwa: Wenn sich eine Teilpopulation z. B. über den rechten Rand hinausbewegt (was durchaus passieren kann, wie wir sehen werden), kommt sie am linken Rand wieder herein (dasselbe gilt natürlich auch für oben und unten).

Die Regeln

zurück zur Übersicht

Die Regeln, denen das Leben und Sterben der Zellen gehorcht, waren ursprünglich (nach Conway) folgende:

- Eine lebende Zelle mit 2 oder 3 (lebenden) Nachbarn bleibt in der nächsten Generation am Leben.
- Eine lebende Zell mit weniger als 2 oder mehr als 3 Nachbarn stirbt (an Vereinsamung bzw. Überbevölkerung).
- Aus einer toten Zelle entsteht eine lebende Zelle, wenn sie genau 3 Nachbarn hat.

Das Ganze kann durch eine zweiteilige Regel (für (a) lebende und (b) tote Zellen) so formuliert werden:

- Eine lebende Zelle bleibt genau dann am Leben, wenn sie 2 ODER 3 Nachbarn hat (sonst stirbt sie).
- Eine tote Zelle wird genau dann wiedergeboren, wenn sie 3 Nachbarn hat (sonst bleibt sie tot).

Häufig wird das als "23/3"-Regel bezeichnet. Damit haben wir auch schon die Kurz-Formulierung der Regeln eingeführt: In der Bezeichnung

"23/3"

beschreiben die Ziffern vor dem Schrägstrich die (Über)lebensbedingung(en) einer lebenden Zelle, der Ziffernteil dahinter steht für die Geburtsbedingung(en) einer toten Zelle. Jede Ziffer entspricht dabei einer Teilbedingung; die einzelnen Teilbedingungen sind ODER-verknüpft.
Z. B. bedeutet die (Über)lebensbedingung "23" von oben: (Nachbarnzahl = 2) ODER (Nachbarnzahl = 3)
oder in Mathcad-typischer Schreibweise: $(nachbarnzahl = 2) + (nachbarnzahl = 3)$

Im Lauf der Zeit wurden die verschiedensten Regeln angewendet. Zwei davon werden in dieser Arbeit untersucht. Die daraus für die Programmierung resultierenden Bedingungen werden durch Analyse eines vorgegebenen "Regel-Strings" (z. B. eben "23/3") gewonnen, wie noch gezeigt wird.

Die Funktionen

zurück zur Übersicht

In Teilprobleme zerlegt ergibt sich folgende Vorgangsweise:

1. Initialisierung (Herstellung der Startpopulation): Die Startpopulation (beschrieben durch die "Populationsmatrix") kann durch Zufallszahlen (0 oder 1) festgelegt werden oder durch bestimmte Muster, die interessante Effekte im Zusammenhang mit bestimmten Regeln ergeben können.
2. Berechnung der Nachbarn-Matrix: Die Elemente der Nachbarn-Matrix, welche die gleiche Dimenension hat wie die Populationsmatrix, enthalten die Zahlen der lebenden Nachbarn aller Felder der aktuellen Populationsmatrix. In unserem Fall hat jede lebende oder tote Zelle *immer* 8 Nachbarplätze (das Spielfeld ist ja randlos!).
3. Jetzt kommen die Regeln ins Spiel: Aus der Nachbarn-Matrix wird unter Berücksichtigung der Regeln die nächste Populationsmatrix (= nächste Generation) berechnet.

Die Punkte 2 und 3 werden dann bis zum Erreichen einer Abbruchbedingung wiederholt.

Die verwendeten Funktionen:

Die Funktion

$$zufall(z,s) := rund(rnd(1))$$

erzeugt - zusammen mit der Mathcad-Funktion *matrix* - mittels Zufallsgenerator die Zahlen 0 oder 1. Mit ihr kann daher eine zufällige Startpopulation generiert werden.

$$null(z,s) := 0$$

ist eine Hilfsfunktion zur Erzeugung einer "leeren" Startmatrix (alle Zellen tot). Sie wird verwendet, wenn mit einem bestimmten Startmuster gearbeitet werden soll.

Die Funktion

$$\begin{aligned}
&nb(pop) := \\
&\quad S \leftarrow spalten(pop) \\
&\quad Z \leftarrow zeilen(pop) \\
&\quad nb \leftarrow matrix(Z,S,null) \\
&\quad pop1temp \leftarrow erweitern\left(pop^{\langle S-1\rangle}, pop\right) \\
&\quad pop1 \leftarrow erweitern\left(pop1temp, pop^{\langle 0\rangle}\right) \\
&\quad S1 \leftarrow spalten(pop1) \\
&\quad Z1 \leftarrow zeilen(pop1) \\
&\quad pop2temp \leftarrow stapeln(submatrix(pop1, Z1-1, Z1-1, 0, S1-1), pop1) \\
&\quad pop2 \leftarrow stapeln(pop2temp, submatrix(pop1, 0, 0, 0, S1-1)) \\
&\quad \textbf{for}\ s \in 1..S \\
&\quad\quad \textbf{for}\ z \in 1..Z \\
&\quad\quad\quad nb_{z-1,s-1} \leftarrow 0 \\
&\quad\quad\quad \textbf{for}\ zaehler \in -1..1 \\
&\quad\quad\quad\quad nb_{z-1,s-1} \leftarrow nb_{z-1,s-1} + 1 \ \textbf{if}\ \left(pop2_{z-1,s+zaehler} = 1\right) \\
&\quad\quad\quad \textbf{for}\ zaehler \in -1..1 \\
&\quad\quad\quad\quad nb_{z-1,s-1} \leftarrow nb_{z-1,s-1} + 1 \ \textbf{if}\ \left(pop2_{z+1,s+zaehler} = 1\right) \\
&\quad\quad\quad nb_{z-1,s-1} \leftarrow nb_{z-1,s-1} + 1 \ \textbf{if}\ \left(pop2_{z,s-1} = 1\right) \\
&\quad\quad\quad nb_{z-1,s-1} \leftarrow nb_{z-1,s-1} + 1 \ \textbf{if}\ \left(pop2_{z,s+1} = 1\right) \\
&\quad nb
\end{aligned}$$

berechnet und liefert die zu einer Populationsmatrix *pop* gehörende Nachbarn-Matrix.

Dazu wird - wegen der dann einfacheren Rechnung - temporär die als Parameter vorliegende Populations-Matrix um zwei Zeilen und zwei Spalten vergrößert: rechts wird die Spalte von ganz links angehängt und links die Spalte von ganz rechts. Entsprechendes passiert oben und unten. Damit ist die Zählung der Nachbarn einfacher.

Die Funktion $pop\_neu(pop\_alt, nb, regel)$ errechnet aus den drei Parametern

- Populations-Matrix pop_alt ,
- Nachbarn-Matrix nb
- und der Zeichenkette $regel$

die neue Populationsmatrix.

Zuerst wird der Regel-String am Trennzeichen "/" in zwei Teile zerlegt, aus denen die entsprechenden Teilregeln ("(Über)lebensregel" und "Geburtsregel") konstruiert werden (siehe Erklärung weiter oben!).

$$
\begin{aligned}
&pop\_neu(pop\_alt, nb, regel) := \\
&\quad regel\_länge \leftarrow zflänge(regel) \\
&\quad slash\_pos \leftarrow strtpos(regel, "/", 1) \\
&\quad regel1 \leftarrow subzf(regel, 0, slash\_pos) \\
&\quad RL1 \leftarrow zflänge(regel1) \\
&\quad regel2 \leftarrow subzf(regel, slash\_pos + 1, regel\_länge - slash\_pos) \\
&\quad RL2 \leftarrow zflänge(regel2) \\
&\quad S \leftarrow spalten(nb) \\
&\quad Z \leftarrow zeilen(nb) \\
&\quad \text{for } s \in 0..S-1 \\
&\quad\quad \text{for } z \in 0..Z-1 \\
&\quad\quad\quad Zelle\_lebt \leftarrow \left(pop\_alt_{z,s} = 1\right) \\
&\quad\quad\quad Geburt \leftarrow \sum_{l_2=0}^{RL2-1} \left(nb_{z,s} = zfinzahl\left(subzf\left(regel2, l_2, 1\right)\right)\right) \\
&\quad\quad\quad Leben \leftarrow \sum_{l_1=0}^{RL1-1} \left(nb_{z,s} = zfinzahl\left(subzf\left(regel1, l_1, 1\right)\right)\right) \\
&\quad\quad\quad pop\_neu_{z,s} \leftarrow Zelle\_lebt \cdot Leben + (\neg Zelle\_lebt) \cdot Geburt \\
&\quad pop\_neu
\end{aligned}
$$

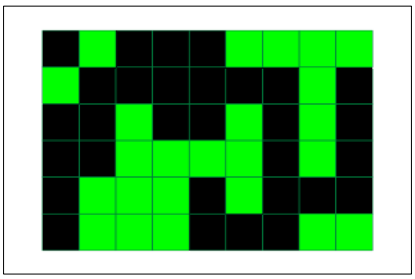
Beispiel:
Die Grafiken werden jeweils in gesperrten Bereichen untergebracht (ohne Kennwort!), um sie vor unbeabsichtigter Veränderung durch versehentliches Daraufklicken und/oder Verschieben mit der Maus zu schützen!

Aus der Start-Population $pop\_start\_0 := matrix(6, 9, zufall)$

als Matrix:

$$pop\_start\_0 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

grafisch dargestellt (hell = 1 = lebende Zelle):

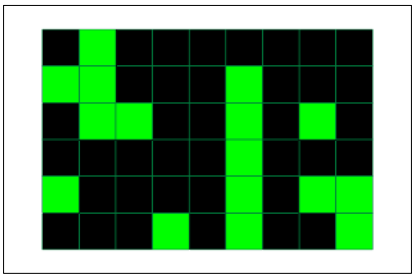


pop_start_0

ergibt sich zunächst die Nachbarn-Matrix $nachbarn\_0 := nb(pop\_start\_0)$
und daraus wiederum die nächste Generation unter Anwendung der "23/3"-Regel:
$pop\_1 := pop\_neu(pop\_start\_0, nachbarn\_0, "23/3")$

$$nachbarn\_0 = \begin{pmatrix} 5 & 3 & 4 & 2 & 2 & 1 & 4 & 5 & 5 \\ 2 & 3 & 2 & 1 & 2 & 3 & 6 & 4 & 5 \\ 1 & 3 & 2 & 4 & 4 & 2 & 5 & 2 & 4 \\ 1 & 4 & 5 & 5 & 5 & 3 & 5 & 1 & 2 \\ 3 & 4 & 7 & 6 & 6 & 2 & 4 & 3 & 3 \\ 5 & 4 & 6 & 3 & 4 & 3 & 5 & 4 & 3 \end{pmatrix}$$

$$pop\_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$



pop_1

Nun muss nur noch die Iterations-Schleife realisiert werden. Dies geschieht mit der Funktion *evolution*. Ihre Parameter sind

- die Matrix der Start-Population
- die anzuwendende Regel in Form einer Zeichenkette (z. B. "23/3")
- die Zahl der Schleifendurchläufe (= Zahl der zu berechnenden Generationen)

evolution gibt einen Vektor zurück, der die einzelnen Populations-Matrzen als Vektorkomponenten enthält (das hat sich - insbesondere, wenn man eine Animationsdatei erstellen will - als günstig erwiesen):
Die vorgegebene Startpopulation ist darin pop_0, die letzte berechnete Populationsmatrix ist pop_{tiefe}:

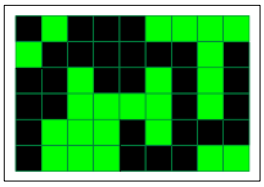
$$evolution(pop\_start, regel, tiefe) := \left| \begin{array}{l} pop_0 \leftarrow pop\_start \\ \text{for } i \in 0..tiefe-1 \\ \quad \left| \begin{array}{l} nb\_temp_i \leftarrow nb(pop_i) \\ pop_{i+1} \leftarrow pop\_neu(pop_i, nb\_temp_i, regel) \end{array} \right. \\ pop \end{array} \right.$$

Das Original

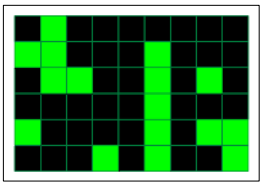
zurück zur Übersicht

Beispiel:
Mit der Start-Population von vorher und der $Regel := \text{"23/3"}$ erhält man die ersten $Tiefe := 11$ Generationen

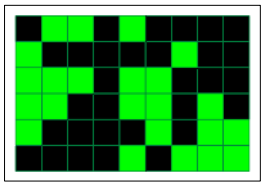
$pop := evolution(pop\_start\_0, Regel, Tiefe)$



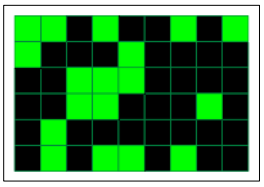
pop_0



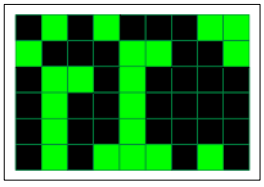
pop_1



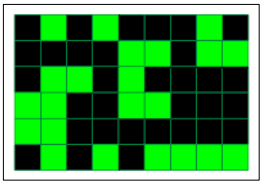
pop_2



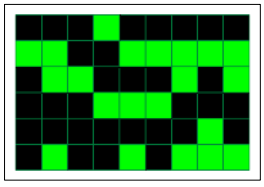
pop_3



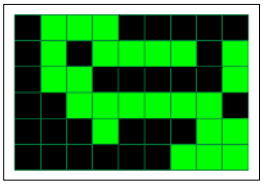
pop_4



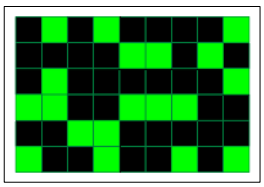
pop_5



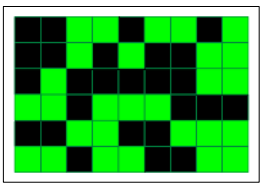
pop_6



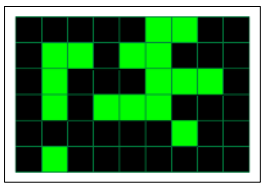
pop_7



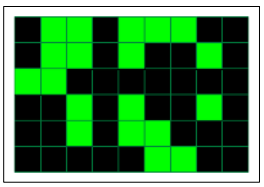
pop_8



pop_9



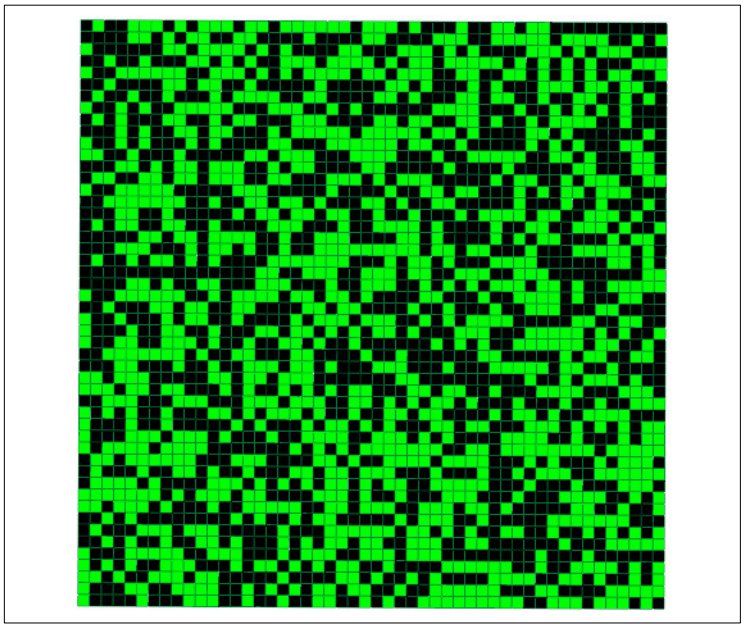
pop_{10}



pop_{11}

Selbstverständlich kann damit nun auch eine Animation erzeugt werden. Wir vergrößern dazu das "Spielfeld" auf 50 x 50 und berechnen 200 Generationen:

$pop\_start\_1 := matrix(50, 50, zufall)$ $pop := evolution(pop\_start\_1, "23/3", 200)$



pop_start_1

Eine so entstandene Animation (.avi-Datei) liegt unter dem Namen **game_of_life_23_3** bei.

Andere Regeln, andere Startmuster

zurück zur Übersicht

Es ist auch ganz interessant, andere Regeln zu untersuchen, z. B. die "35/3"-Regel.
Wir erzeugen dazu ein ganz bestimmtes Startmuster (den sogenannten "Schwimmer"), ein Zellengebilde, das sich zyklisch verändert und dabei imstande ist, sich "fortzubewegen".
Dabei sehen wir die Möglichkeit vor, dass sich der Schwimmer in alle vier möglichen Richtungen bewegen kann.
Seine F□Grundform wird beschrieben durch die Matrix

$$schwimmer\_rechts := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Die anderen "Schwimmer"-Matrizen gewinnen wir daraus:

$$schwimmer\_links := \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$schwimmer\_ab := schwimmer\_rechts^T$

$schwimmer\_auf := schwimmer\_links^T$

Zuerst wird ein "leeres" Spielfeld erzeugt ($schwimmer\_leer := matrix(21,21,null)$), in das mittels der Funktion

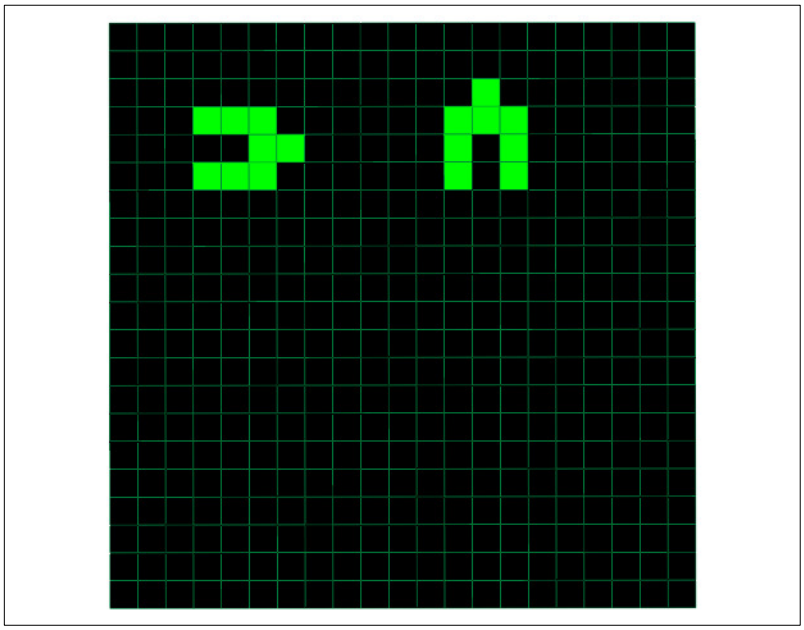
$muster\_einfügen(matrix,muster,zeile,spalte) :=$

$Z \leftarrow zeilen(muster)$
$S \leftarrow spalten(muster)$
$for\ \ z \in 0..Z-1$
  $for\ \ s \in 0..S-1$
    $matrix_{zeile-1+z,\,spalte-1+s} \leftarrow muster_{z,s}$
$matrix$

nacheinander zwei Schwimmer gesetzt werden, die sich in verschiedene Richtungen bewegen:

$schwimmer\_start := muster\_einfügen(schwimmer\_leer,schwimmer\_rechts,4,3)$

$schwimmer\_start := muster\_einfügen(schwimmer\_start,schwimmer\_auf,3,13)$



$schwimmer\_start$

$pop := evolution(schwimmer\_start,"35/3",100)$

Die damit erzeugte Animationsdatei hat den Namen **schwimmer** .

Auf der oben zitierten Internet-Seite können noch viele andere Anregungen dieser Art gefunden werden, etwa auch zyklisch veränderliche Figuren, die an Ort und Stelle bleiben, wie der "Pulsator":

$$pulsator := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

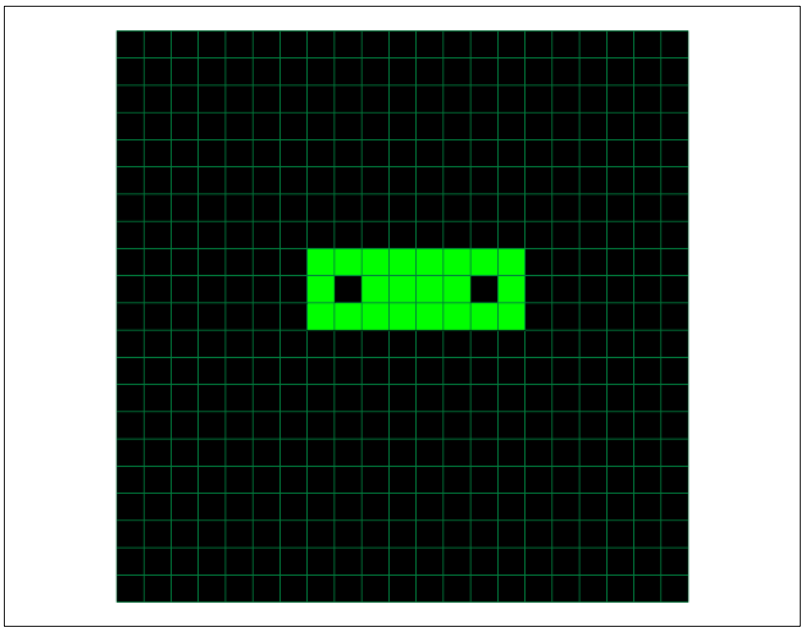
Wir erzeugen wieder zuerst eine leere Populationsmatrix und fügen das Pulsator-Muster ein:

$pulsator\_leer := matrix(21, 21, null)$

$pulsator\_start := muster\_einfügen(pulsator\_leer, pulsator, 9, 8)$

$pop := evolution(pulsator\_start, "23/3", 29)$

So sieht der Pulsator beim Start aus:



pulsator_start

Die Animation (Dateiname: **pulsator**) umfasst 2 Perioden und wird am besten im Endlos-Modus abgespielt.

zurück zur Übersicht