

Gegeben ist das folgende NetLogo-Programm:

```

patches-own [ aktuell neu ]

to setup
  clear-all
  ask patches [
    set aktuell 0
  ]

  ask n-of 100 patches [
    set aktuell 1
    set pcolor blue
  ]

  reset-ticks
end

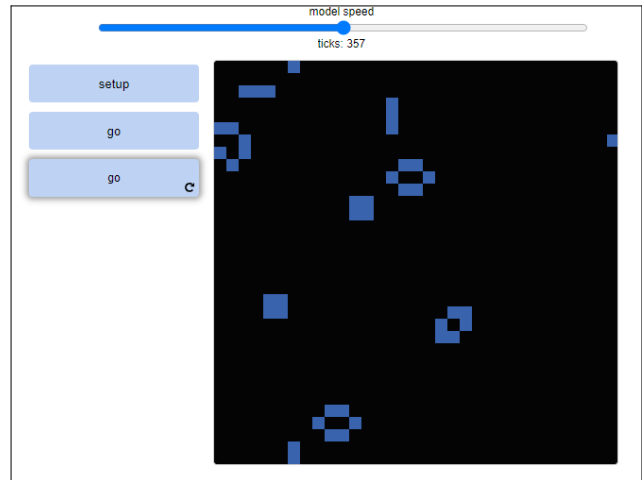
to go
  ; bitte alle lebendigen (aktuell=1) patches:
  ask patches with [aktuell = 1] [
    set pcolor blue
    let nachbarn count neighbors with [aktuell = 1]

    if (nachbarn < 2 or nachbarn > 3) [
      set neu 0
    ]
  ]

  ; hier fehlt die Geburtsregel:

  ask patches [
    set aktuell neu
    ifelse aktuell = 0 [
      set pcolor black
    ][
      set pcolor blue
    ]
  ]
  tick
end

```



Hinweise

- Dieses Logo-Programm verwendet nur Patches, also den Hintergrund, und gar keine Turtles.
- Mit patches-own kann man Variablen definieren, die zu Patches gehören sollen. Jedes Patch hat dann sein eigenes Exemplar dieser Variablen.
- Das „Game of Life“ ist jetzt schon 50 Jahre alt und der Versuch, das Leben und Sterben von Zellen in einer Zellkultur zu simulieren. Die Spielregeln findet man sicher online – ein Teil (die Sterberegel) ist schon im Programm eingebaut.

Aufgaben

1. Suche die Spielregeln des Game of Life heraus und ergänze die Geburtsregel. Prüfe, ob Klötze und Blinker korrekt funktionieren.
2. Ergänze Schieberegler, mit den man die Zahlen in den Geburts- und Sterberegeln einstellen kann (Abschnitt „Abweichende Regeln“ im Wikipedia-Artikel)
3. Informiere dich, z.B. im Projekt Paint, wie man das Spielfeld größer bekommt und Sorge für richtig Platz.
4. Schön wäre ein Eingabemodus, in dem man Anfangskonfigurationen zeichnen kann. Das kann entweder ein dauerhafter Modus sein, in dem jeder Mausklick eine Zelle zum Leben erweckt, oder über einen Schalter gesteuert werden. Auch hierfür findest du Hinweise im Paint-Projekt.
5. In der Originalversion leben Zellen ewig, und in gewisser Weise trifft das auf Einzeller auch real zu. Versuche trotzdem, ein Älterwerden und Sterben für Zellen in die Simulation einzubauen. Jedes neu geborene Patch bekäme dann ein Alter, das mit jeder Runde...

Erweitere das Programm zu einem kleinen Zeichenprogramm

- Die setup-Prozedur sollte die Größe des Bildes einstellen können. Dazu braucht man die Funktionen `resize-world` und `set-patch-size` die für Anzahl und Größe der Patches zuständig sind. Wie genau sie benutzt werden, findest du heraus.
- Mache die Bildgröße über Schieberegler oder Input-Felder einstellbar, ohne das Programm ändern zu müssen. Als Variablennamen dafür bieten sich `breite` und `hoehe` an. Die müssen allerdings nicht global definiert werden, das machen Regler/Input automatisch selbst.
- Hilfreich ist es, wenn die Patchgröße nicht auf einen festen Wert eingestellt ist, sondern von der Weltgröße abhängig berechnet wird. Ein Beispiel: Der Bildschirm ist ca. 1000 Pixel hoch. Bei einer Welthöhe von 30 Patches (`max-pycor - min-pycor`) sollte die Patchgröße nicht mehr als 30 Pixel betragen, eher kleiner sein.

Zusätzliche Anregungen

- Die aktuelle Zeichenfarbe kann auch mit einem Turtle angezeigt werden. Dazu erzeugt man ein Turtle und setzt es in der Zeichenprozedur immer an die Position der Maus: `setxy mouse-xcor mouse-ycor`. Sicher findet man auch ein passendes Shape (Vorschlag: Kreis) und kann dann die Farbe beim Zeichnen zuweisen.
- Die Pinselgröße könnte man an einen Schieberegler binden. Dann wird nicht nur das Patch auf der Koordinate des Mauszeigers gefärbt, sondern auch alle umliegenden. Dafür gibt es in NetLogo die Funktion `in-radius`:
`ask patches in-radius ... [ set pcolor farbe ]`
- Mit einem Schalter kann man beim Zeichnen zwischen Vorder- und Hintergrundfarbe umschalten. Dazu braucht man eine zweite Variable und muss dafür sorgen, dass beim Hintergrund-Ändern auch nur die Patches in Hintergrundfarbe betroffen sind.