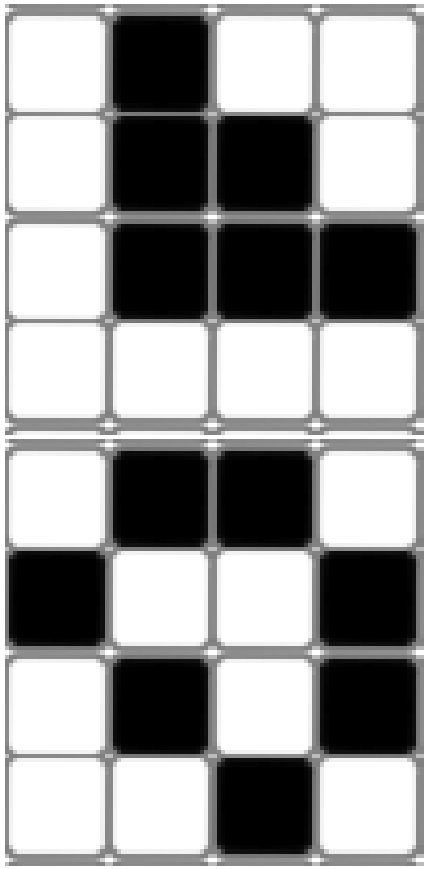
Conway's Game of Life

Valentino Bergamotto

31.01.2024

Einleitung

- Lebende Zellen mit weniger als 2 lebenden Nachbarn sterben.
- Lebende Zellen mit 2 oder 3 lebenden Nachbarn überleben.
- Lebende Zellen mit mehr als 3 lebenden Nachbarn sterben.
- Tote Zellen mit genau 3 lebenden Nachbarn werden lebendig.



Struktur

- Dateistruktur wie die Vorlage in src, examples und test unterteilt.
- Das README und eine Lizenz sind auch im "main"-Ordner

src

- install_packages.jl, installiert alle notwendigen Packages
- cell.jl, enthält eine struct "Cell" und verschiedene Funktionen, für diese
- gameboard.jl, enthält eine struct "Gameboard" und eine Funktion update_game()
- options.jl, enthält eine Funktion get_option(), um die Startkonfiguration zu bestimmen und ein paar Unterfunktionen, die verschiede Konfigurationen zurückgeben
- main.jl, führt alles wichtige aus und implementiert den letzten Schritt für die Interaktivität

Cell

- Cell speichert alle wichtigen Informationen, die eine Zelle braucht, sowie Funktionen, die diese verändern und aufrufen können
- Dazu gehören neighbours, alive, update und button
- Informationen, wie die Anzahl lebender Nachbarn werden über Hilfsfunktionen in der Datei bestimmt

Gameboard

- Gameboard speichert eine Matrix von Zellen und implementiert die Visualisierung dieser
- Für die Visualisierung wurde GLMakie verwendet
- Die Initialisierung wird mit einer Matrix mit 0 und 1 ausgeführt, alle Zellen erhalten dann ihre relevanten Informationen
- Die Reihenfolge. in welcher man einer Zelle ihre Nachbarn zuweist, ist sehr wichtig. Deswegen sieht der Algorithmus dafür sehr komisch aus

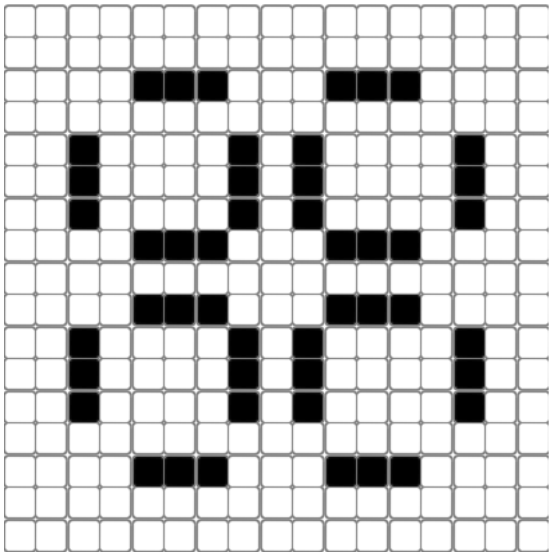
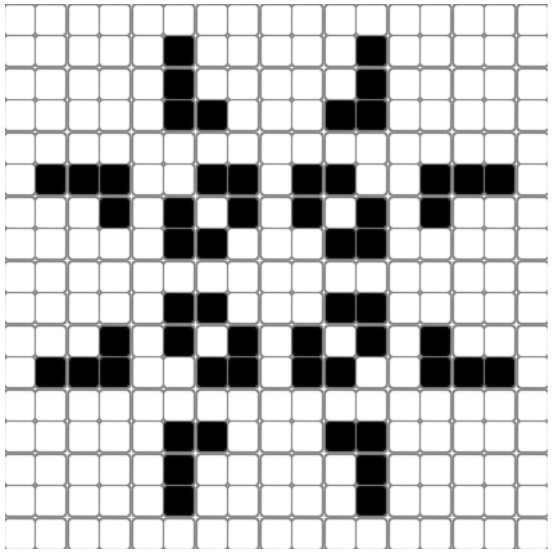
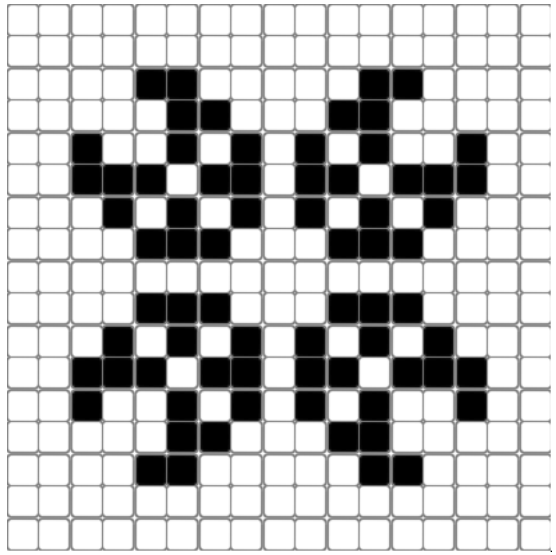
Festlegen von Nachbarn

```
for k in 0:1
    for l in 0:2
        if k == 1 && l > 0
            continue
        end
        set_neighbours(current_state[i,j], current_state[i+k-1,j+l-1])
    end
end
set_neighbours(current_state[i,j-1], current_state[i,j])
set_neighbours(current_state[i-1,j], current_state[i,j])
set_neighbours(current_state[i-1,j-1], current_state[i,j])
set_neighbours(current_state[i-1,j+1], current_state[i,j])
```

Resultate

- Ursprüngliche Initialisierung hat Komplexität $O(y \cdot x)$ (40x40 braucht ca. 90 Sekunden)
- Danach ist jeder Schritt auch für große Matrizen sehr schnell (für 40x40 sofort fertig)

Resultate

Fazit

- Mit einer anderen Visualisierungsmöglichkeit, kann man die Initialisierung vermutlich stark verbessern. Dabei geht aber die Interaktivität verloren.
- Conway's Game of Life kann mit vier sehr simplen Regeln, sehr komplexe Gebilde erschaffen
- Deswegen ist es auch noch nach über 50 Jahren eins der beliebtesten "Programmierspiele" und viele Menschen suchen nach neuen Gebilden

Vielen Dank für eure Aufmerksamkeit