Game of Life Project Description

Elias Pichottka

University of Potsdam

February 19, 2020

Contents

1 Game of Life Description

Game of Life describes a deterministic cellular automaton, invented by the british mathematican John Horton Conway in 1970. Each cell of a two-dimensional field $M_{ij}(t)$ can hold the state 'empty' or 'occupied'. The field-configuration is changed in discrete time steps, in compiliance with a set of deterministic rules. Whether a cell will be 'occupied' or 'empty' in time step $t+1$ depends on its state at time point t, as well as the count of adjacent occupied cells $N_{ij}(t)$. The adjacent cells of M_{ij} are defined as M_{i+kj+l} with $k, l \in \{-1, 0, 1\}$. Therefore $0 \leq N_{ij} \leq 8$ holds.
The original rule set includes the following rules:

1. $(M_{ij}(t) = \text{'empty'}) \wedge (N_{ij}(t) = 3) \Rightarrow (M_{ij}(t+1) = \text{'occupied'})$
2. $(M_{ij}(t) = \text{'occupied'}) \wedge (2 \leq N_{ij}(t) \leq 3) \Rightarrow (M_{ij}(t+1) = \text{'occupied'})$
3. $\neg(M_{ij}(t) = \text{'empty'} \wedge N_{ij}(t) = 3) \wedge$
   $\neg(M_{ij}(t) = \text{'occupied'} \wedge 2 \leq N_{ij}(t) \leq 3) \Rightarrow (M_{ij}(t+1) = \text{'empty'})$

2 Task Description

Project goal is a python game of life simulation. The simulation should visualize the matrix $M(t)$ for a time intervall, creating an animation. Matrix size, boundary conditions, initial condition and set of rules should be user inputs. Rules should be formatted as a string. The string starts with the letter 'B' followed by a list of adjacency numbers N_{ij}, representing cell transitions from 'empty' to 'ocuppied'. Subsequently the string continues with '/S' and a second list of adjacency numbers N_{ij}, representing cell transitions from 'occupied' to 'occupied'. Additionaly a few interesting examples should be illustrated using the simulation. The following subsections will briefly describe improvements/extensions that might be implemented.

2.1 Graphical User Interface

A GUI could be realised with the toolkit Tkinter, in order to further improve the user expierence.

2.2 Event Programming

Simulation start and stop could be controlled by keyboard or mouse events (e.g. mouse button click).

2.3 Speed Optimization

Matrix calculations could be done in parallel on multiple CPU/GPU cores or improvements regarding the algorithmic complexity.

3 Approach Description

I will start by realising the core functionalty as mentioned in section 2. States 'occupied' and 'empty' will be modeled as integers (0 and 1). The integer representation should open doors to exploiting matrix addition or multiplication for applying a given set of rules.