Gondela Łukasz

Multiscale Modeling - 1 report

Introduction

Cellular Automata methods can be used for various simulations in different fields of science. This application contains few implementations types of cellular automata that allows to adjust simulations as you please. This report presents one of the many approaches to this problem.

Technology used

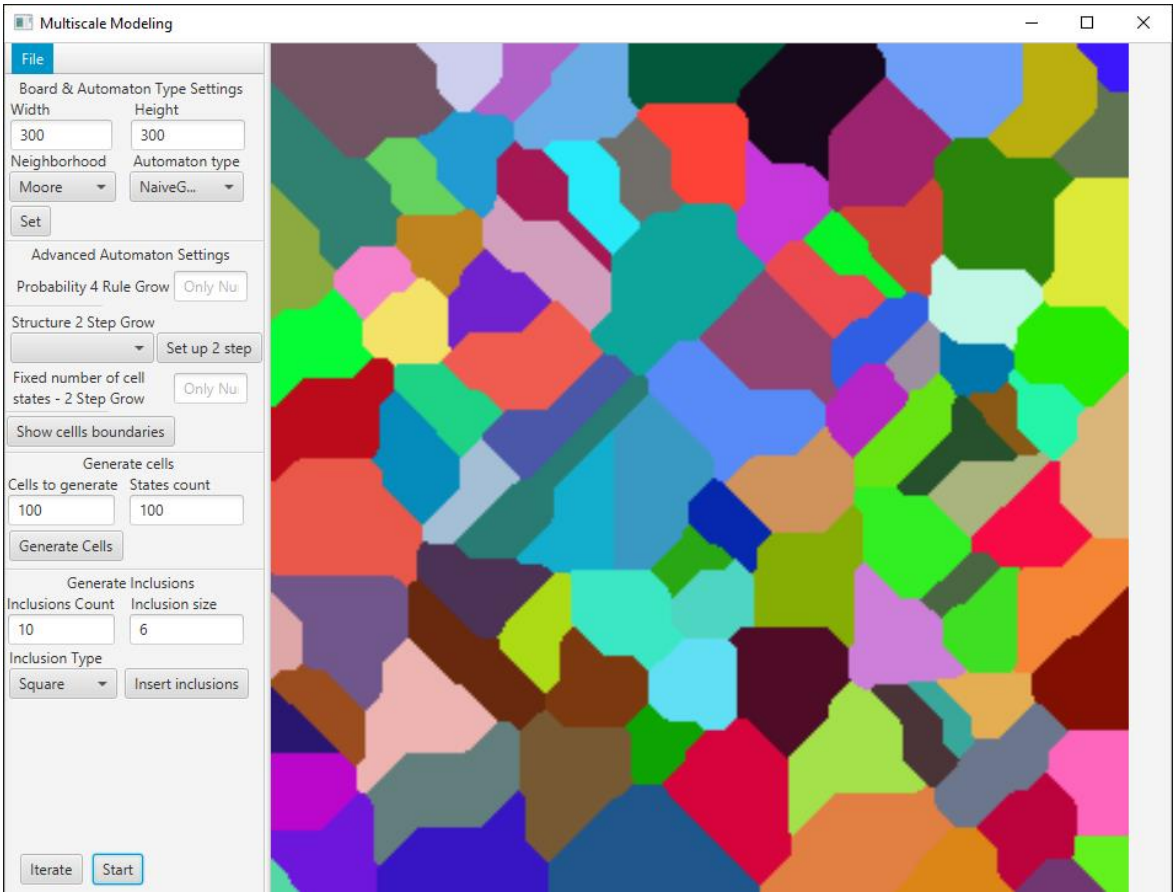
This project has been implemented using Java programming language. The main reason behind this choice was ease of writing complex programs with graphical user interface that can be run on different platforms. Simulations implemented in this program relay on classes that support functional-style operations on streams of elements, such as map-reduce transformations on collections. Thanks to that, multithreading support was added.

The representation of cell grid is based on flyweight design pattern. Grid is stored in one dimensional array that contains references to specific elements from array of unique cells. Those unique cells are generated during initialization of grid prior to starting simulation.

For the UI JavaFX was chosen, mostly because this it is intended to replace Swing as the standard GUI library. Also Oracle provides really great tools for fast building appearance of applications (it is called SceneBuilder).

UI – Description

Picture number one shows how main window look like. On this picture we can see all available options and settings and also visualized result of simulation.



Picture 1 Main window of application

On the right hand side there is visualized board (using imageView) that contains current state of calculations.

On the left hand side there is:

- File menu that contains options (for all of them you can select file from any directory):
  - Import board state from CSV
  - Import board state from BMP
  - Save board current state to CSV
  - Save board current state to BMP
- In section “Board & Automaton Type Setting” we can specify:
  - Dimensions for simulation
  - Automaton type – which is type of simulation
    - Naïve grain grow

- Four rule grain grow
- Two step naïve grain grow

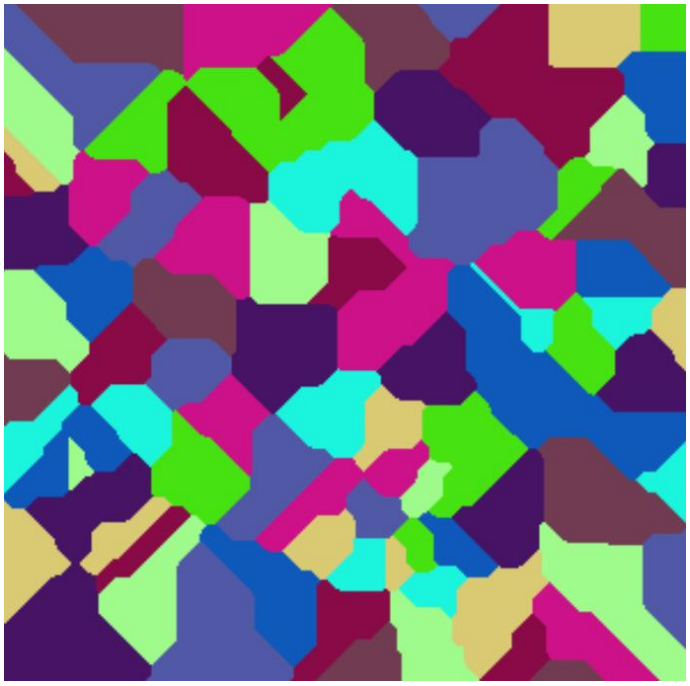
When we have provided those setting we need to initialize simulation using “Set” button

- In section “Advanced Automaton Settings” we have:
  - Field with percentage probability used in “four rule grain grow”
  - Some settings for “Two step naïve grain grow” – which is partially implemented
  - Option to show boundaries of all cells (this should be run after finalizing simulation)
- Section “Generate cells” as it says is used for generating inputted number of cells and placing them on to the board for simulation
- “Generate Inclusions” is used for inserting in to the board chosen amount of inclusions
  - In “Inclusion Type” we can set if we want to use circular or squared inclusions
- Iterate button advances simulation by one step
- Start button is starting multithreaded simulation

Examples of implemented functionalities

Naïve grain grow

Picture number two shows result of naïve grain grow simulation with settings that are specified below this picture.



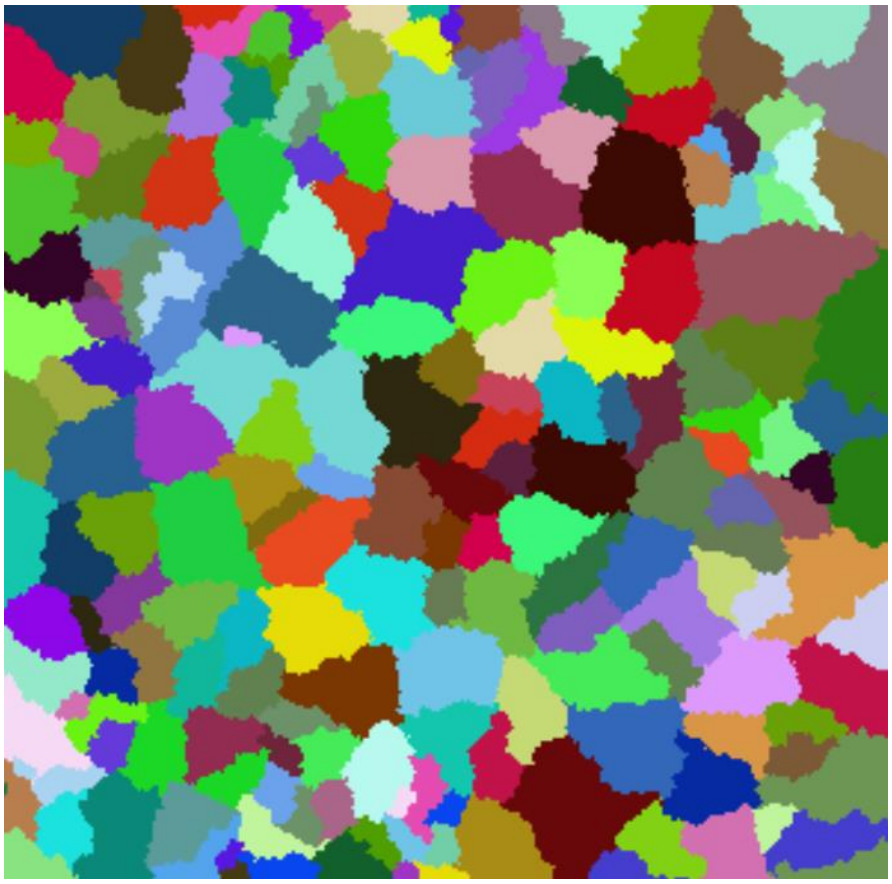
Picture 2 Example of naive grain grow simulation

Parameters of simulation:

- Dimensions of grid: 300x300 cells
- Moore neighborhood
- 10 unique cells that was randomly spread with count of 100

Four rule grain grow with 5% probability

Picture number three shows result of four rule grain grow simulation with settings that are specified below this picture.



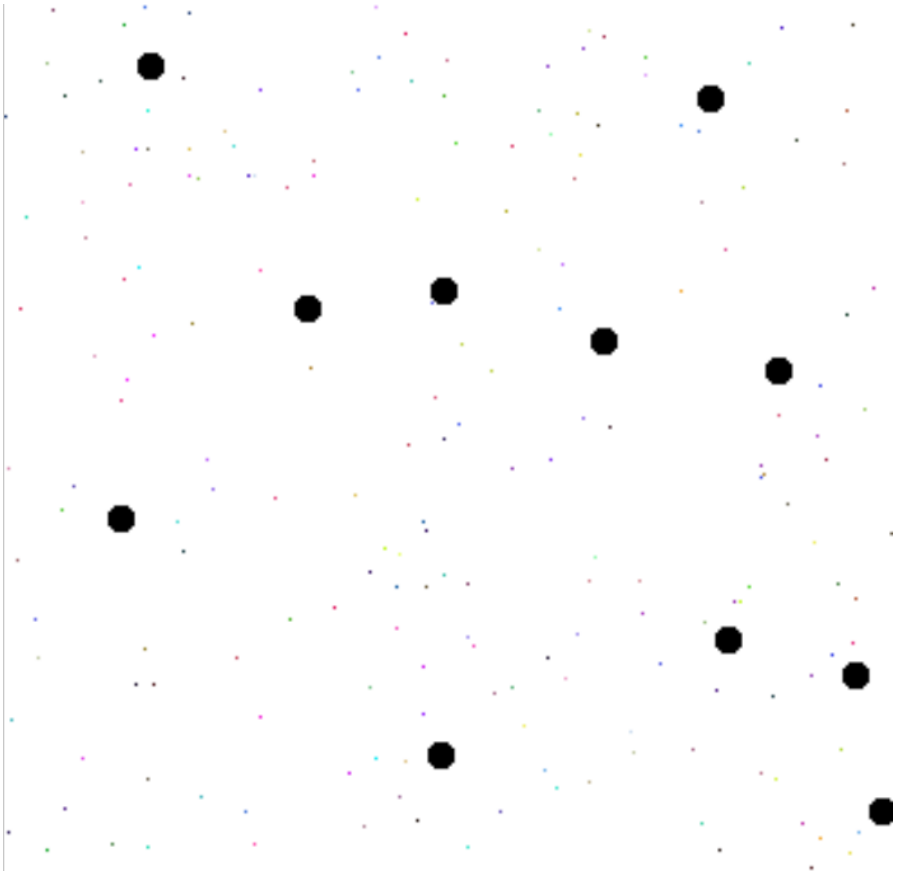
Picture 3 Example of four rule naive grain grow simulation

Parameters of simulation:

- Dimensions of grid: 300x300 cells
- Moore neighborhood
- Probability for fourth rule 5%
- 100 unique cells that was randomly spread with count of 200

Inclusions with prepared board for simulation

Picture number four shows outcome of populating cellular grid with settings that are specified below this picture.



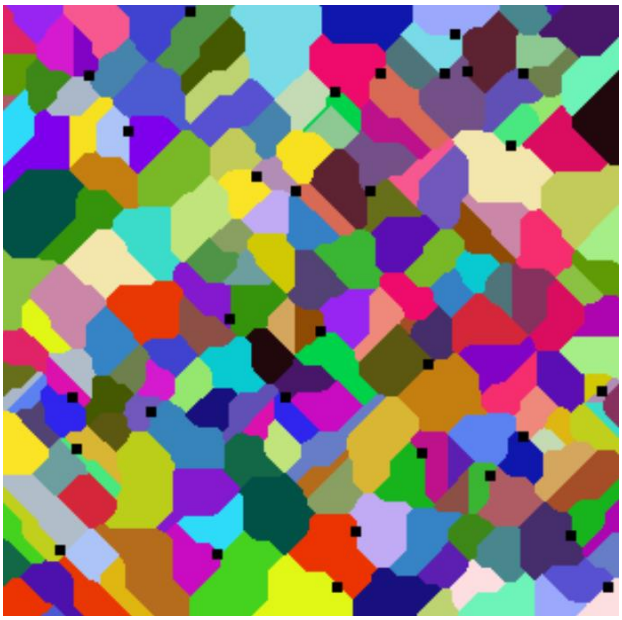
Picture 4 Example of prepared grid with inclusions for simulation

Parameters of example:

- Dimensions of grid: 300x300 cells
- Eleven randomly generated circular inclusions with radius 5 cells
- 100 unique cells that was randomly spread with count of 200

Inclusions set after simulation

Picture number five shows outcome of naïve grain grow and on top of that there are generated inclusions with settings that are specified below this picture.



Picture 5 Example of inclusion insertion on finalized board from simulation

Parameters of example on finalized simulation of naïve grain grow:

- Dimensions of grid: 300x300 cells
- Moore neighborhood
- Thirty randomly generated square inclusions with radius 2 cells
- 100 unique cells that was randomly spread with count of 200

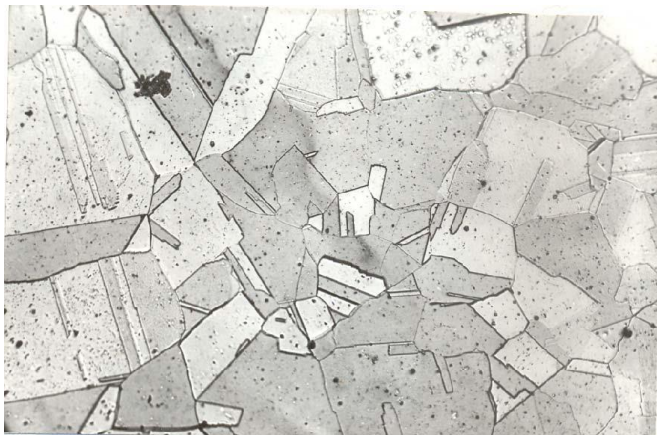
Comparison between simulation and actual metal structure

One phase metal alloy vs naïve grain grow simulation

First example is one phase metal alloy that was undergone a cold deformation process (Picture 6). In comparison picture 7 is representing implementation of naïve grain grow with parameters:

- Dimensions 300x300
- Moore neighborhood
- Ten unique cells that was randomly place in exactly 50 random places

There is a lot of similarity in those two pictures. In implementation we can find those specific squealy grains that are in the structure of real metal.



Picture 6 Equi-axial grains in one phase alloy (source: https://docplayer.pl/40368212-Ksztaltowanie-struktury-i-wlasciwosci-materialow-metalowych-metodami-technologicznymi-1-odlewanie-2-obrobka-plastyczna-3.html)

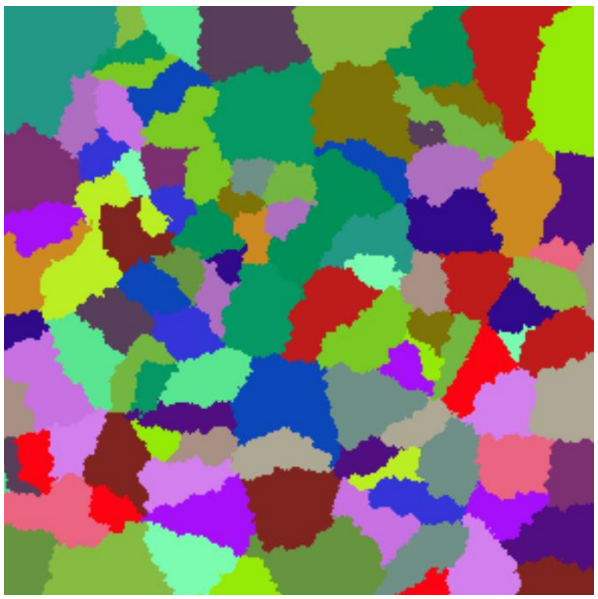
Picture 7 Result of naive grain grow simulation

Stainless steel vs four rule grain grow

Second example is standard stainless steel microstructure (Picture 8). In comparison, picture 9 is representing implementation of four rule naïve grain grow with parameters:

- Dimensions 300x300
- Probability of fourth rule 10%
- Hundred unique cells that was randomly spread

Picture 8 Stainless steel microstructure (source: https://pl.dreamstime.com/zdj%C4%99cie-stock-srebny-grunge-t%C5%82o-stali-nierdzewnej-tekstura-czysty-metalu-diamentu-talerz-p%C5%82ynnie-tillable-struktura-konkretn%C4%85-cze%C5%9B%C4%87-res-image76207821)



Picture 9 Result of four rule naive grain grow simulation

Conclusions

This software with implemented various types of cellular automata can be useful in generating examples of steels microstructure. Both upper simulations quite good resembles actual metal alloy structure which proves correctness of implementation. Unfortunately there is major restriction in size of simulation that this program can generate. The biggest problem here is memory usage and spreading tasks between multiple cores in processor.