Multiscale Modelling

Simple Grain Growth Cellular Automaton

Author: Szymon Trela
Date: 15.11.2019

1. Short introduction

The goal of this project was to implement application simulating grain growth of steel microstructure with use of Cellular Automata methods. Application allows modification of few simulation parameters, for example dimensions or number of initial grains. Report below contains:

- actual look of application with description of user interface
- a few kinds of results of example simulation

2. Used technology

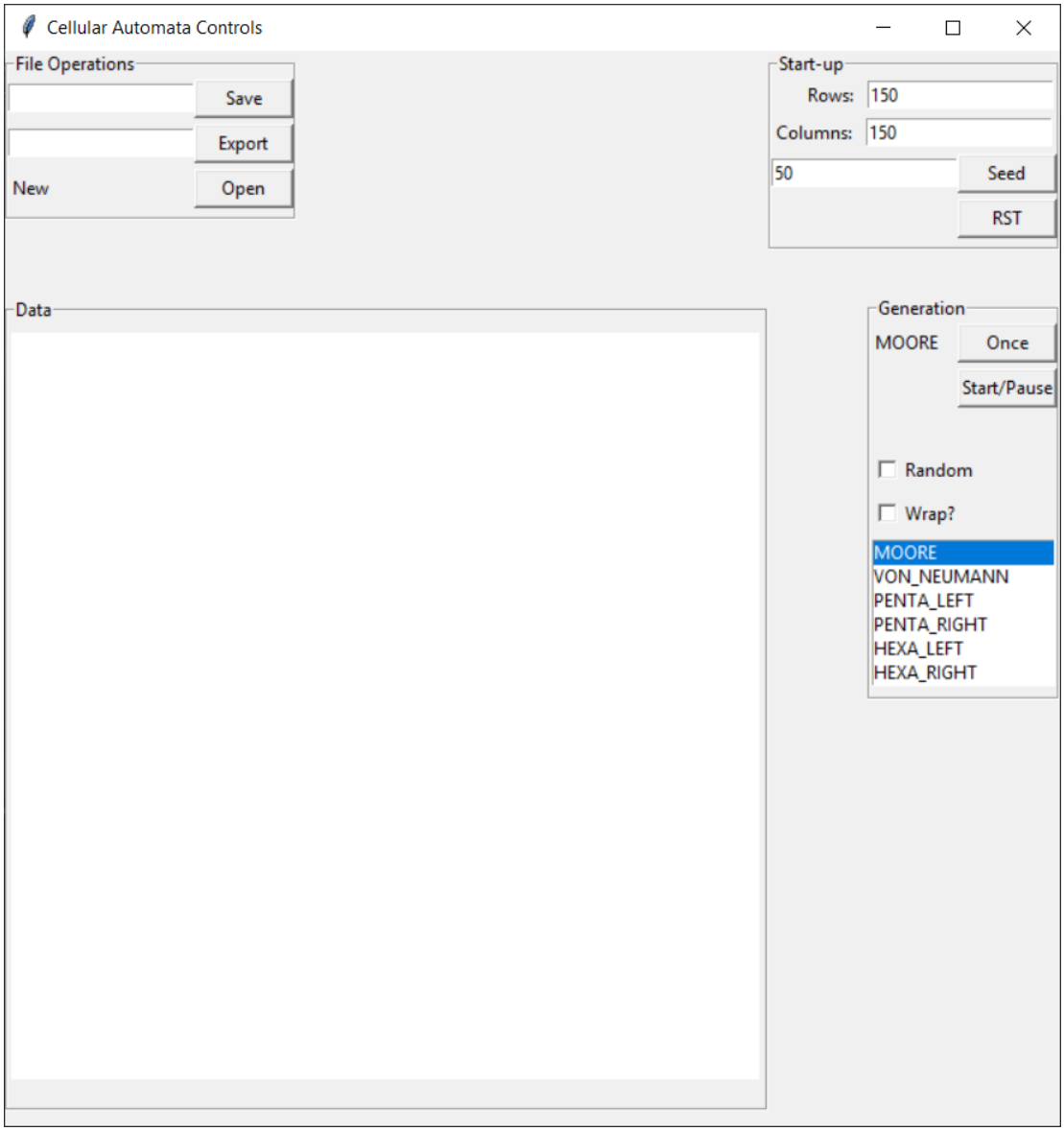
Implementation was done in **Python**. As the language is high-level and object-oriented there was a possibility to implement sophisticated algorithms relatively fast, what was crucial. Furthermore it supports number of libraries used for data structures manipulation and also computations of numerous types. Python offers an easiness of launching application on different systems without any additional tools.

As a version control system the Github is used, because of its reliability.

Link to Github: https://github.com/SzymonIgor/MultiscaleModelling

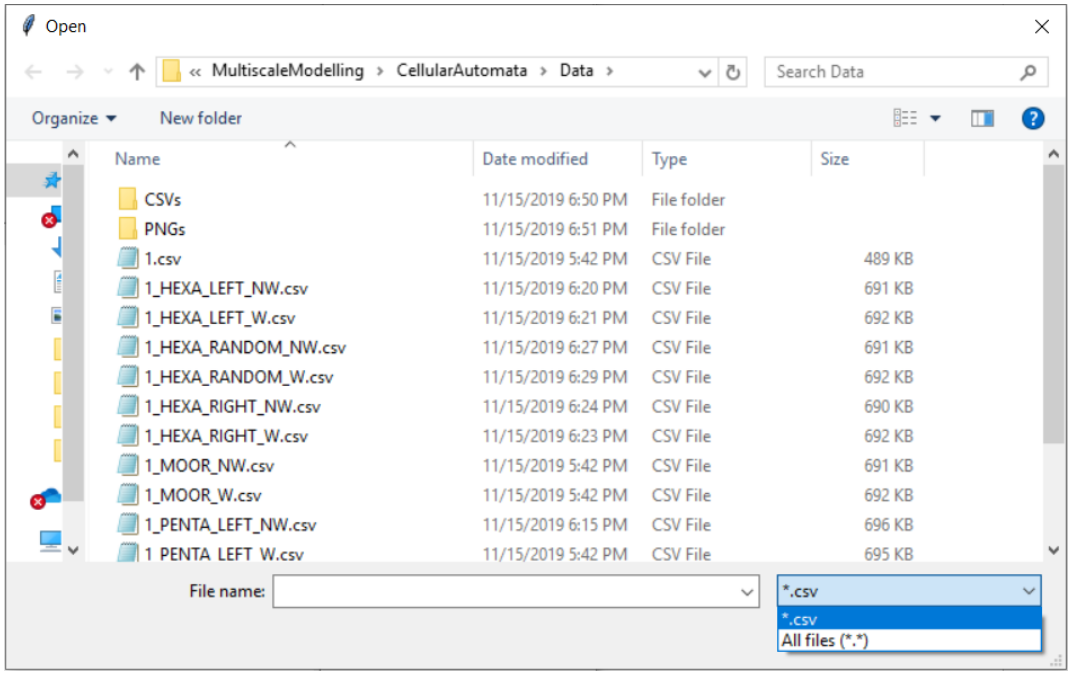
3. GUI

Graphical User Interface (shown below) was prepared in single-window, which parameters can be chosen in, simulation can be started, results can be exported in 3 types of files, also loaded to continue simulation that has not been ended.



Frames Description:

- File Operations – it is a place where data can be saved[1], exported[2] and imported(opened[3])
    1. Save – allows to save current state of simulation in order to open it later on. The file format is - .csv. Default name is: CellularAutomata_YYYY-MM-DD_hhmmss (e.g. CellularAutomata_2019-11-15_200358).
    2. Export" button – creates two files. One of them is .csv file that contains 3 columns: [row, column, grainID or 0 (when empty)]. The second one is .png file that displays imaged version of data, where colour is based on grainID. There can be up to 300 different colours. Default name is: Export_YYYY-MM-DD_hhmmss (e.g.Export_2019-11-15_200358.csv).
    3. Open" button – allows to import data file to continue computation. After clicking the button the dialog box is opened and filter is applied (.csv files shown only).



- Data – Canvas where data is visualised. Initialized image has 500x500 pixels. There is a possibility to calculate through smaller not rectangular shapes, however bigger are not permitted for now.
- Start-up – this panel shall be used when new generation of seeds is needed, then these parameters are to be set.
    1. "Rows" entry – number of rows to be generated
    2. Columns:" entry – number of columns to be generated

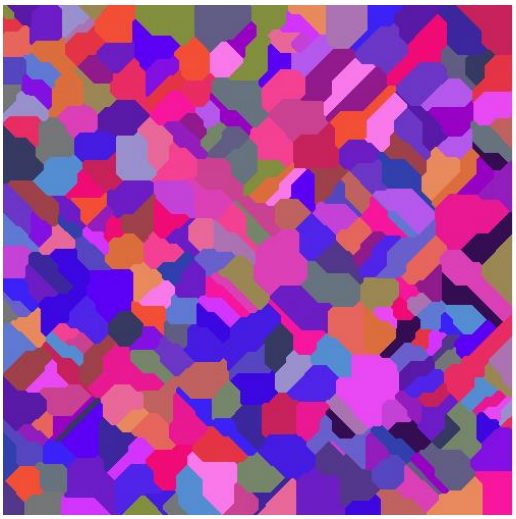
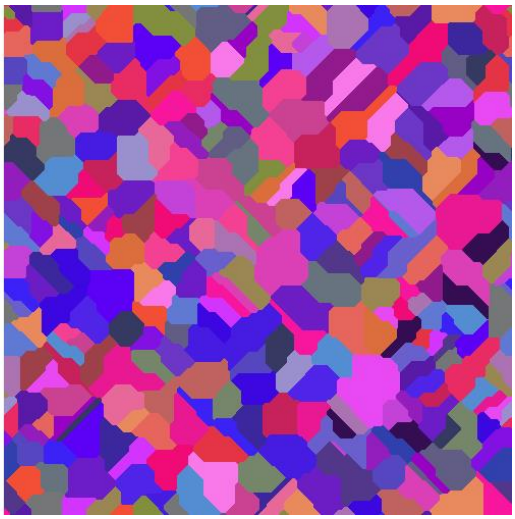
3. “Seed” button – Generation of seeds that is given on the left side of the button, resetting of data is applied before generation
4. “RST” button – reset of the data

- Generation – this panel is used to control the simulation
  1) “MOORE” label on the greyed background – current chosen mask that is used for simulating
  2) “Once” button – simulate one step
  3) “Start/Pause” button – Starting and Pausing the simulation steps
  4) “Random” checkbox – is considered when using masks: PENTA_LEFT, PENTA_RIGHT, HEXA_LEFT or HEXA_RIGHT. When is ticked left and right types are chosen randomly
  5) “Wrap?” checkbox – not ticked: zeros around the matrix. When ticked matrix is wrapped
  6) Listbox – list of masks available for usage

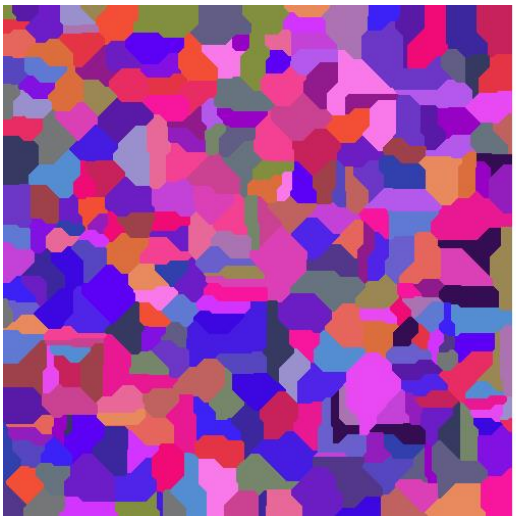
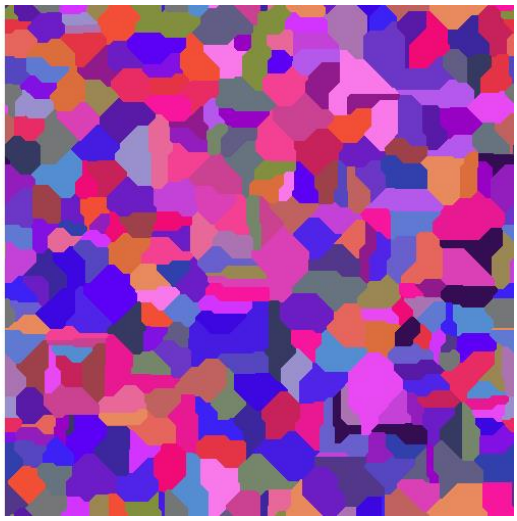
4. Results

Pairs of results of simulating grain growth using same template (500x500 matrix, no. of grains = 300, same initial grains positions). On the left sides are images of generation without checkbox “Wrap?” ticked, on the right with this option choosen.

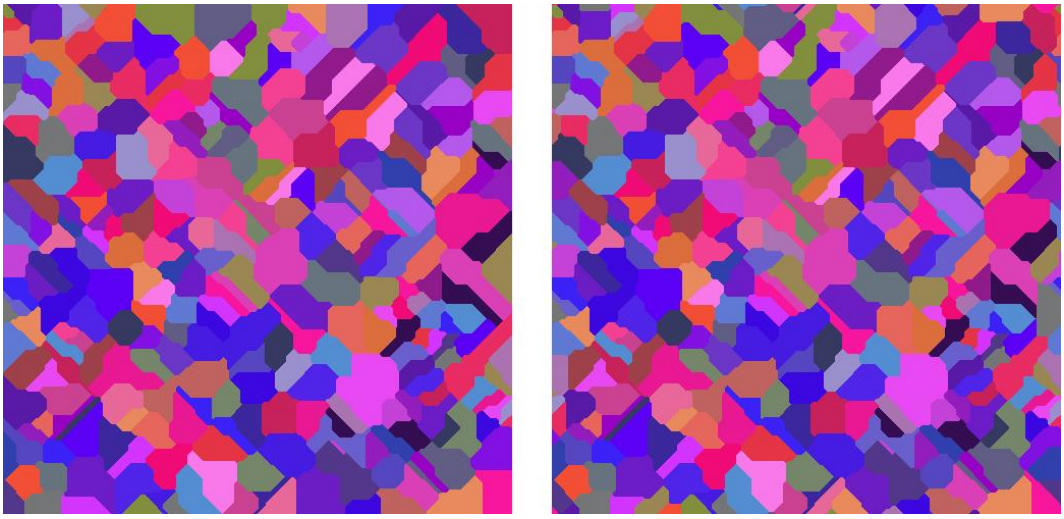
- MOORE

- VON NEUMANN

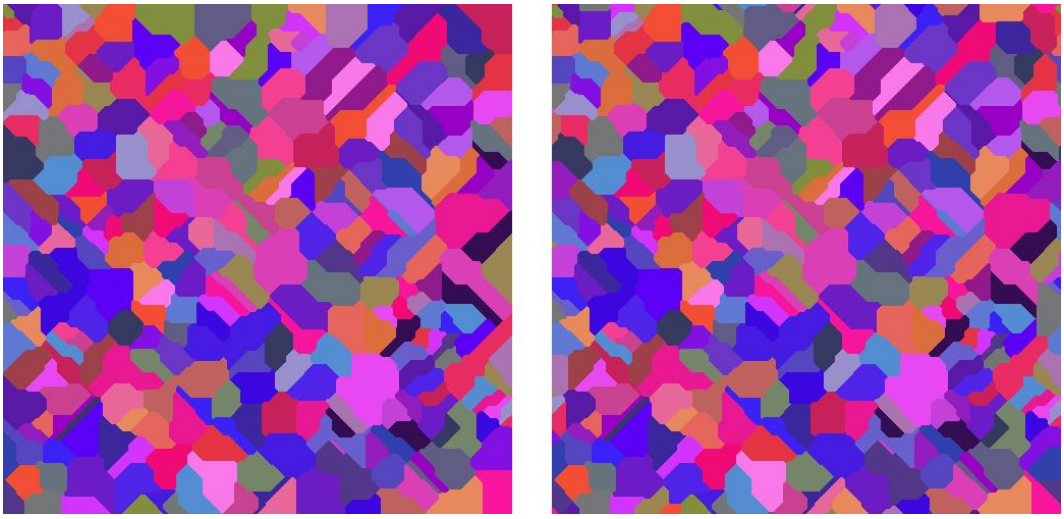



- PENTA_LEFT

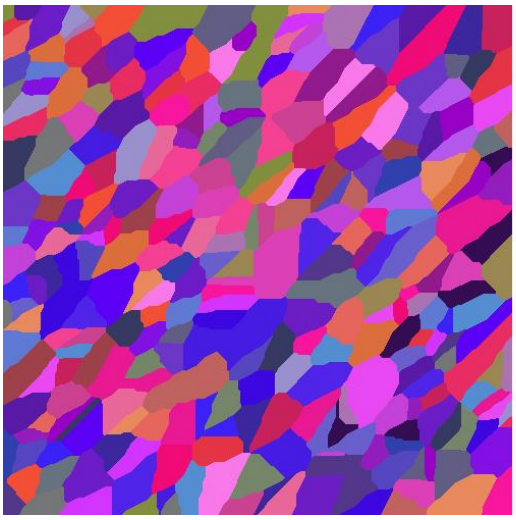
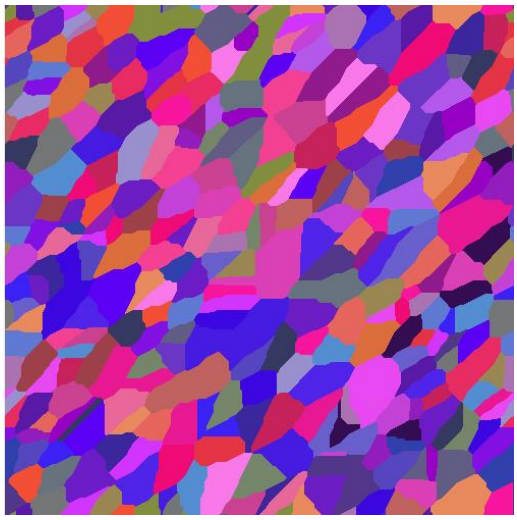


- PENTA_RIGHT

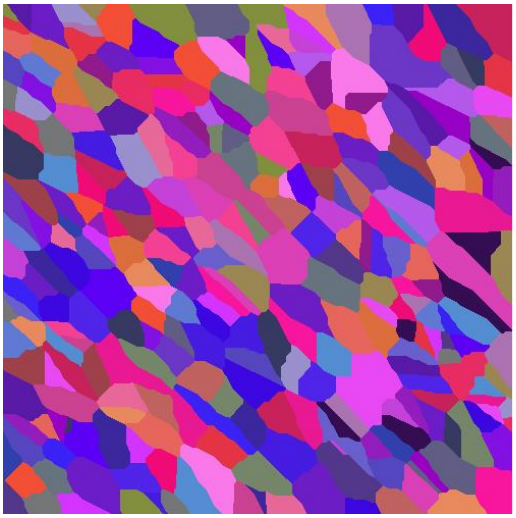
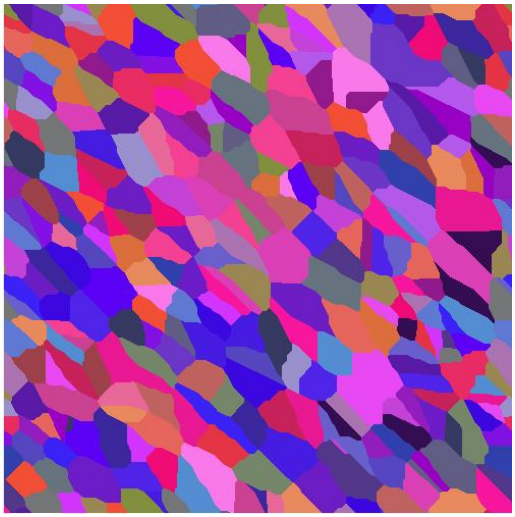
- PENTA_RANDOM



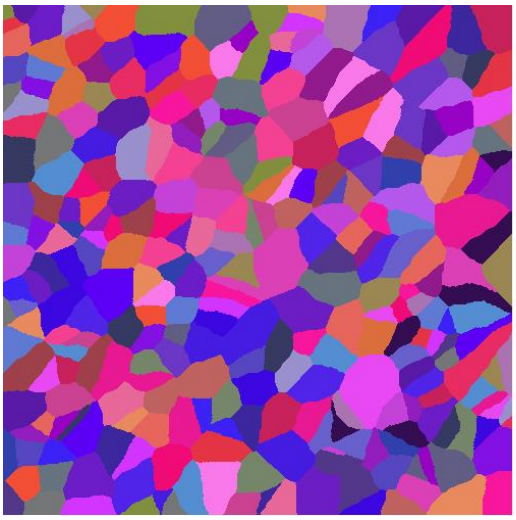
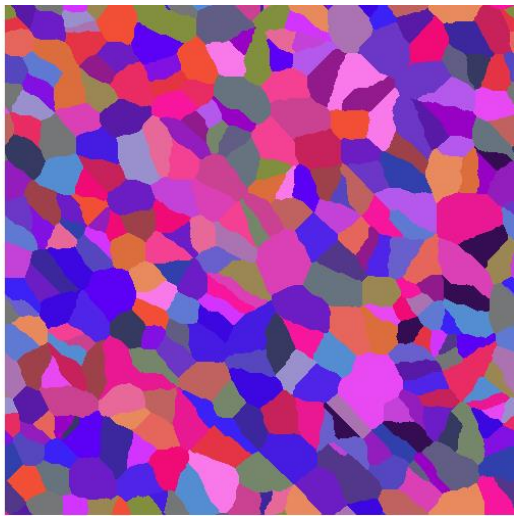
- HEXA_LEFT

- HEXA _RIGHT

- HEXA _RANDOM

5. Summary

The application is capable of simulating grain growth. The Graphical User Interface is readable and user-friendly. Requirements are fulfilled. However, as time was crucial, there are a few possible improvements, such as: refactoring of the code, exception handling, GUI, optimization – multithreading / multiprocessing and probably some more, which will be found during evolution of the program.