Author: Kamil Łuc

Multiscale Modelling

Report 2

Repository: <https://github.com/kamilluc/multiscale-modelling>

1. Technology (same as in previous part)

**Java 11 LTS** - is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Author choose Java because it is sufficient for this task and wanted to expand his knowledge about new version (11 vs 8 from 4 years ago).

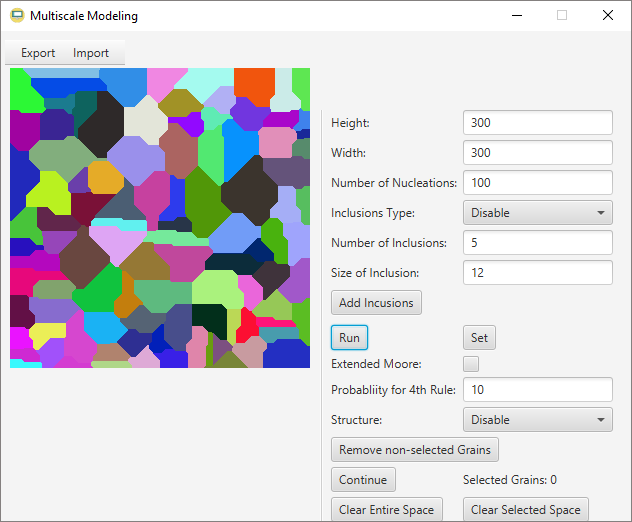
|  |  |
| --- | --- |
| Strengths | Weaknesses |
| Strongly-typed language, compiled, good performance, old and tested, very popular on market, variety of libraries, big company (Oracle) behind it, multiplatform, one of the best if not the best environment for programming desktop applications. | Slower than C/C++, quite hard to learn (above basic level) compared to Python/JavaScript etc., confusing licence. |

**JavaFX** - is a Java library used to build Rich Internet Applications. The applications written using this library can run consistently across multiple platforms. The applications developed using JavaFX can run on various devices such as Desktop Computers, Mobile Phones, TVs, Tablets, etc. It’s basically a standard for Javas GUI application (that’s why author choose this technology), it replaced SWING many years ago. Also it allows to use XML like syntax for structure View elements like buttons ale CSS like syntax for styling them.

|  |  |
| --- | --- |
| Strengths | Weaknesses |
| Follows MVC pattern, good looking apps could be created quite easily thanks to HTML and CSS like syntax. Easier and better than predecessor SWING. | Since JDK 9 quite complex initial configuration (JavaFX is no longer part of JDK). |

Tools like **InteliJ Idea** (Integrated Development Environment) where code was written and **GIT** to manage repository was used but they are not part of application itself so the they are not described here.

1. Graphical User Interface



9 

1

2

4

6

7

1 0

8



3

5

Figure 1 GUI of application

1. Main Window of developed application.
2. Menu for importing and exporting files, in \*.txt and \*.bmp format.
3. Image of current microstructure, user can click on any part to select corresponding grain, to unselect it user have to press shift and while doing so also click on grain.
4. Main settings for simple grain growth, user can specify size of microstructure, number of nucleons.
5. Section dedicated to inclusions, user can specify their number, type (square or circular and position, on boundaries of random) and size after selecting these all what is left is clicking Add Inclusion button to add them and show in current microstructure.
6. Main buttons of app, Set is for confirmation of chosen settings and generate starting nucleation, Run is for begging simulation.
7. Panel with checkbox to enable extended more method (instead of simple grain growth and specify probability (1-100 %) for fourth rule of this method.
8. Drop down menu with structure types, there are 2 substructure and dual-phase. User have to specify with method He or She wants and then remove non-selected grains and after that generate new ones, by clicking Continue button.
9. Buttons to clear structure from grains and leave boundaries of previous structure, depends with button was chosen it will either leave boundaries of just selected grains or all of them.
10. Counter of selected grains, show information about their current number.
11. Results, application features
12. **Class 8 - Monte Carlo grain growth algorithm**

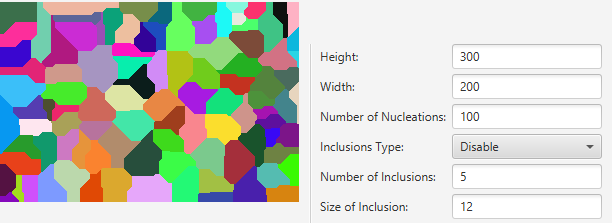


Figure 2 Simple grain growth

Implementation of Simple grain growth algorithm, with von Neumann neighbourhood. In GUI size of structure could be specified like in Figure 2, height and width is set to 300 and 200 respectively, initial number of nucleons is set to 100, after pressings Set and Run button, above (Figure 2) structure is generated.

1. **Class 9 – Modification of MC grain growth algorithm - substructures CA, MC**

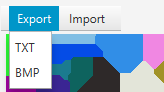


Figure 3 Export Feature

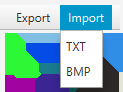


Figure 4 Import Feature

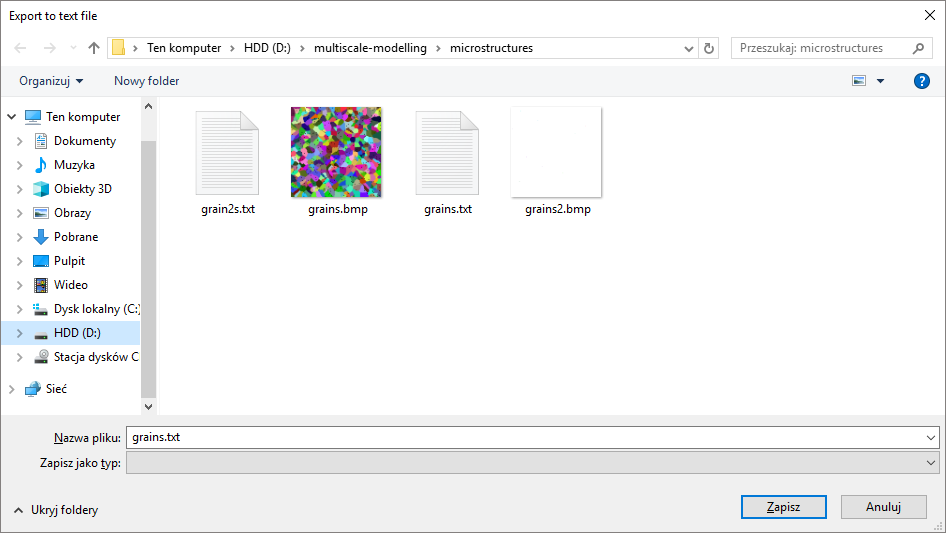


Figure 5 File Chooser

1. **Class 10, 11 and 12 - MC static recrystallization algorithm - energy distribution, nucleation, growth**

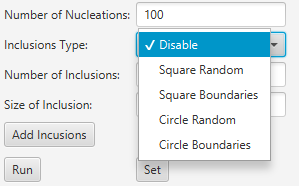


Figure 6 Inclusions type

Inclusions could be added at the beginning of simulation (only random ones), or at the end of it. User specifies (see Figure 1 part 5 and Figure 6) number of inclusions, its type and size (diagonal for circular inclusions and side for square ones) after that last thing wat is needed is pressing Add Inclusions button.

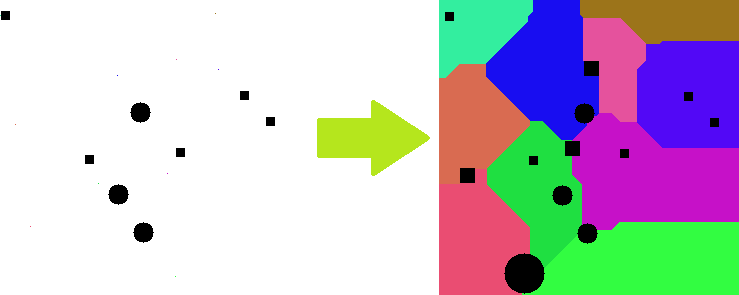


Figure 7 Different type of inclusions

In the figure 7 all possible type of inclusions were added, there are 3 small circles and 5 smaller squares added before simulation in random spots. Then simulation was completed and after that 1 big circular inclusion and 3 medium sized squares ones on boundaries were generated. Also example result could be seen in Figures 15,17,19,21.

1. Comparison with real microstructure

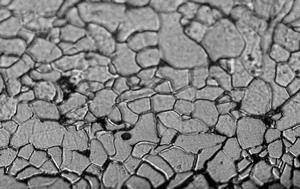


Figure 12 Microstructure of steel after hot rolling

<https://galvanizeit.org/education-and-resources/resources/technical-faq-dr-galv/mechanical-properties-of-galvanized-steel>

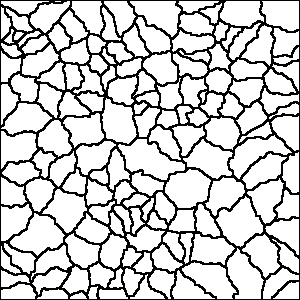


Figure 13 150 Nucleons, Extended Moore with 10% probability for 4th rule.

Structure of one phased element with many clear and smooth boundaries. Extended Moore method and an option of clearing generated space except boundaries is good example of features of developed program.

1. Summary

* In many cases microstructures very similar to “real micro structures” could be generated by this application, but not all of them (see Figure 22), there are cases where more tools are needed.
* Generated microstructures could be saved for example to BMP format and then used in CAE type programs to create better, more realistic models.
* Program could be expanded in many ways, for example by implementing more inclusions types or grains like “needle shaped” (see Figure 22) or creating another module for 3D simulations.
* Coding was quite easy but took some time, but despite correct implementation program could be refactored and optimized. For example, threads could be used for heavier computing wise functions or code could be spited to more smaller files.

Monte Carlo simulation is different method than Cellular Automate and it’s possible to generate more realistic results.

Monte Carlo method is very slowly with basic assumptions, but in application applied some improvements like: skip inside grain nodes, random only neighbors for new state. Performance of algorithm still isn’t the best, because application doesn’t use parallel calculations. It could be next step of application development.

Static recrystallization generating new nucleons, where energy is highest with more probability and process trying to minimalize energy in structure. Recrystallized grains haven’t energy so these can’t be seeds for new nucleation. We can observe this same process in really structures. Application possible to show energy in structure whenever it will be possible.

Monte Carlo method is working completely random. We can generate a lot of complex structures, but the most important parameter is number of iterations. The parameter is proportional to size of grains. After a large number of iterations simulation will generate only one big grain – it will be stable state, without energy.

Combination MC and CA methods allows to generate realistic structures, and it is a key to get the best results. In nature we can observe random structures like in MC, but some of them are more predictable.

For MC simulation Moore neighborhood will be batter than Von Neumann, because generate more stable results.