AGH University of Science and Technology

Laboratory Report

Multiscale Modelling

Project 2: Monte Carlo grain growth algorithm

Agnieszka Gorczycka (279727)

1. 1st Laboratory

The goal was to implement Monte Carlo grain growth algorithm and create GUI *(Figure 1)*. The language of my choice was Java, and the IDE was, once again, IntelliJ IDEA – what I do at work is Front-End development, which is why I chose the language and IDE I did use a little during studies, additionally using JavaFX, which allows to style the whole GUI with CSS. Size of the map, the number of grains, number of MCS and minimum and maximum energy – those are all the fields that are adjustable from the graphical user interface. The most important to do in this step was implementing Monte Carlo Algorithm *(Figure 2)* shown with the use of all the default settings from GUI.

The GUI was slightly rebuild this time, since in my previous project I managed to get a lot of logic behind the functions wrong. Naïve grain growth algorithm (for this project I used Von Neumann) must have been changed in some parts, the same thing had to be done to initializing the growth below the selected grain, since the last time it worked quite badly. All the other options were implemented during the process of creating this project.

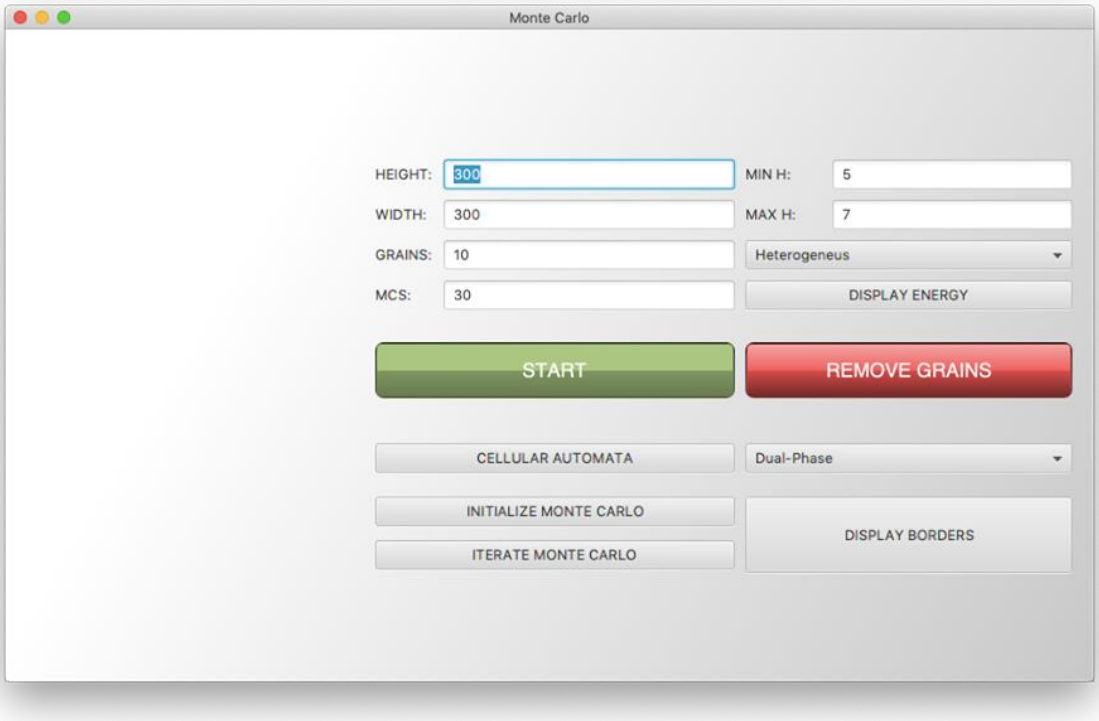

Figure 1: GUI

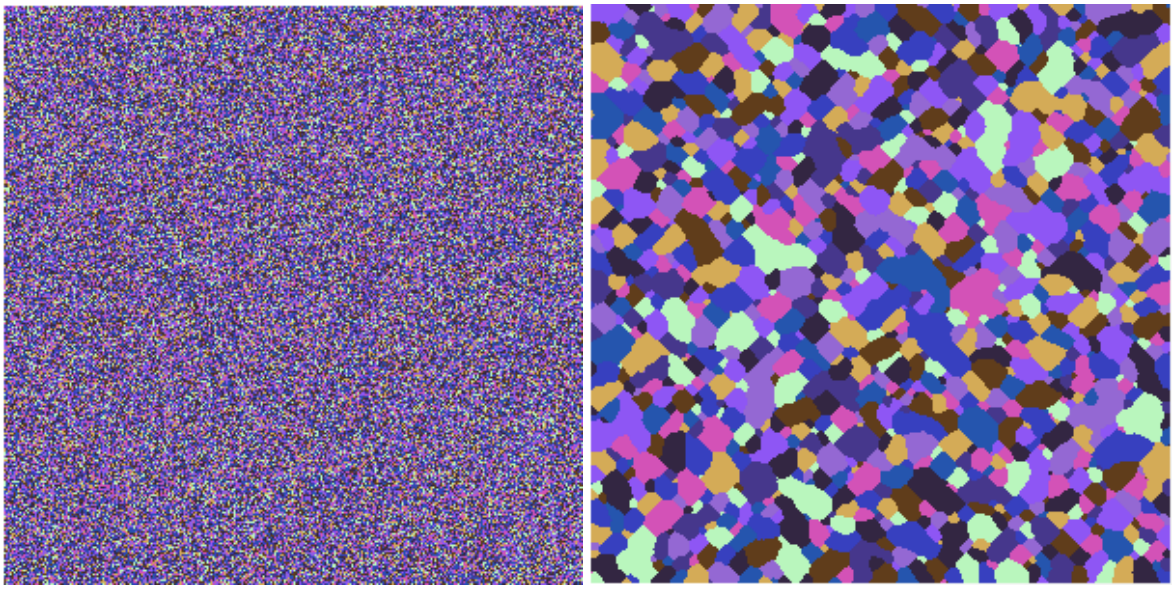

Figure 2: Monte Carlo Grain Growth

The goal of the first laboratory was to generate the Monte Carlo Grain Growth. What we can choose is the number of grain colours and MCS, all of which result in the slight differences in possible microstructures.

2. 2nd Laboratory

The goal of this laboratory was to implement the functionality of grain growth of Cellular Automata or Monte Carlo below the grains selected during previous grain growth. The selected grains are shown of *Figure 3*, grain growth of Cellular Automata is presented on *Figure 4* and *Figure 5* demonstrates Monte Carlo grain Growth below the selected grains. The Monte Carlo below the selected grains may not be very clear, since I unwisely chose grey colour for selected grains.

Figure 3: Selected grains

Figure 4: Cellular automata below selected grains

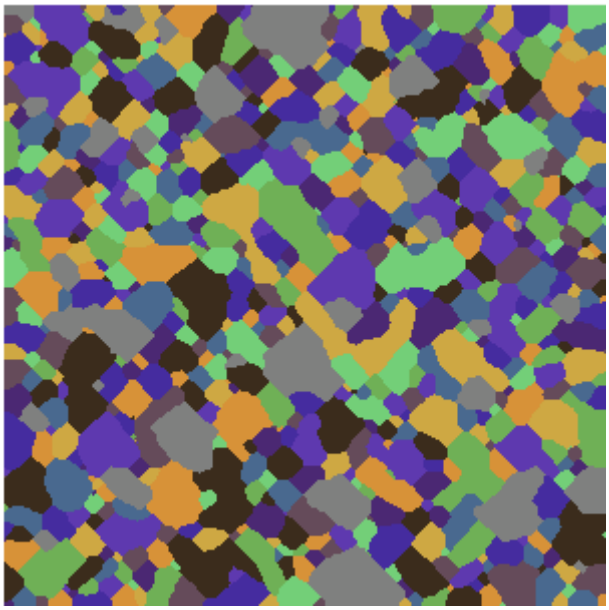

Figure 5: Monte Carlo below the selected grains

3. 3rd Laboratory

The objective of this class was to implement energy distribution along with choosing the type - homogeneous (*Figure 6*) or heterogeneous (*Figure 7*). The energy was run on the last state, presented by *Figure 5*. The user has the ability to pick the minimum and maximum energy values. *Figure 8* presents comparison with the real-world microstructure.

Figure 6:Homogeneous energy

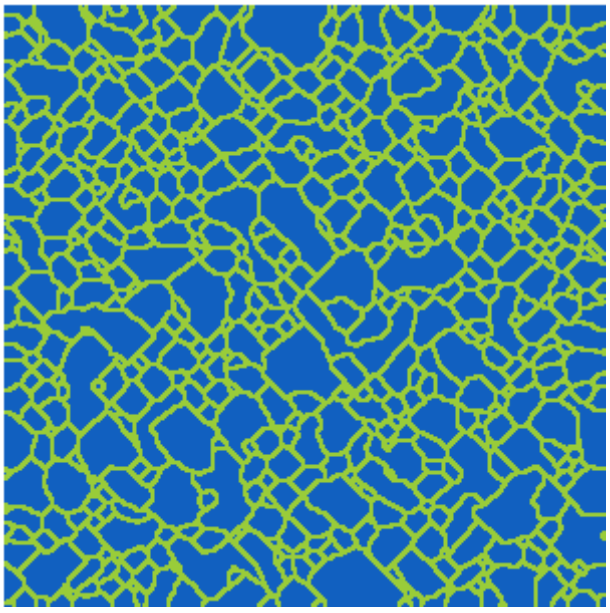

Figure 7: Heterogeneous energy

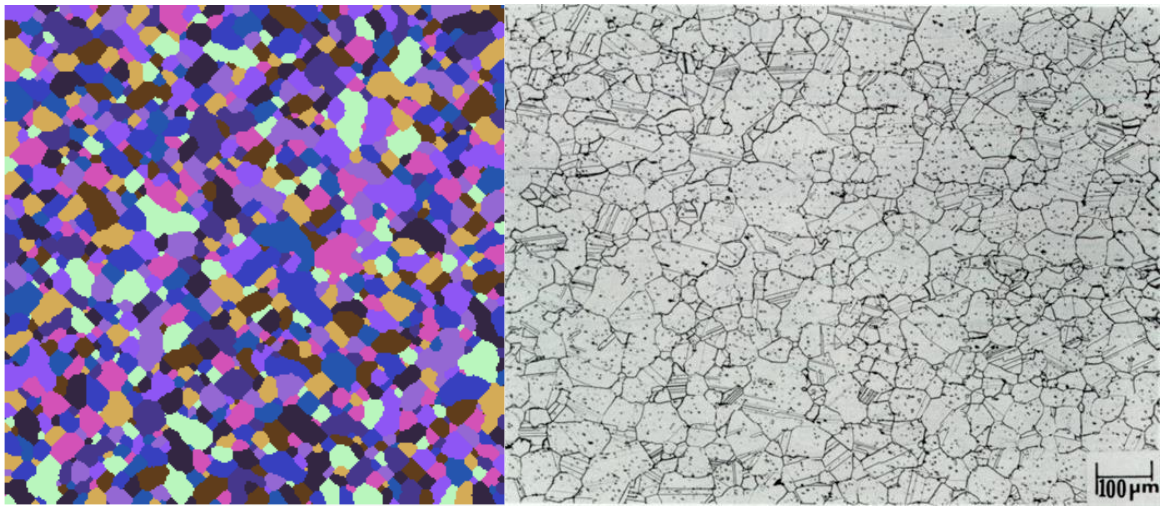

Figure 8: Monte Carlo grain growth compared to real-world microstructure ((ASTM 5 grain size) Annealed at 2250°F (1230°C) – courtesy of http://www.haynesintl.com/alloys/alloy-portfolio_/High-temperature-Alloys/HAYNES-230-ALLOY/typical-microstructure.aspx)

The comparison demonstrates similarity between the real microstructure and custom written algorithm, generating one. Naturally, increasing the accuracy of algorithm would have positive impact on the actual mapping.

4. 4th and 5th Laboratory

Sadly, I did not manage to do this part of the classes.

5. Conclusions

The second project represents the software designed to show Monte Carlo Grain Growth simulation. Monte Carlo algorithm can find usage in many different fields, the most popular could be microstructure modelling in my opinion.

The program created by me allows the user to choose between dual phase and substructure microstructure, choose the size of the display, see the energy distribution and choose its type, show boundaries and present grain growth below the selected grain. Personally, as I mentioned above, neither am I a Computer Science graduate, nor do I work with Java, or C#, or C++ professionally, so what I am happy about is that this project let me understand Java and Java FX better.