Multiscale modelling   
Second report   
Konrad Tabiś

1. **Technology**

To implement this project I choose Java programming language, because it is class-based, object-oriented. Additionally Java allows developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. A prior knowledge of this programming language was also a very important factor in this choice.

The JavaFX Scene Builder application was used to create the GUI, which speeds up the work with interface. JavaFX Scene Builder is a visual layout tool that lets users quickly design JavaFX application user interfaces, without coding. Users can drag and drop UI components to a work area, modify their properties, apply style sheets, and the FXML code for the layout that they are creating is automatically generated in the background. The result is an FXML file that can then be combined with a Java project by binding the UI to the application’s logic.

To implement this project IDE created by JetBrains - IntelliJ IDEA was used. The powerful static code analysis and ergonomic design make development not only productive but also an enjoyable experience.

1. **Application**

In this chapter will be presented the most important functionalities implemented during second part of classes.

After choosing ‘Monte Carlo Grain Growth’ in combobox named ‘Simulation Type’ application provides the opportunity to create Monte Carlo structure and grain growth. The number of unique nucleons, iterations and grain boundary energy can be set via interface, what was presented on Figure 1.

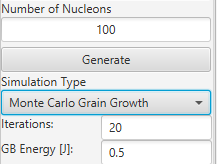


Figure 1 Monte Carlo options

After Monte Carlo grain growth user may pick some grains to block them by clicking mouse cursor on grains. After cleaning unpicked grains interface allows to add new nucleons and execution next grain growth.

Another functionality is energy distribution, which can be done in two ways: homogeneous and heterogeneous. In the first case, the user enters only energy value(Figure 2).

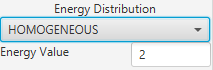


Figure 2 Homogeneous energy distribution

In the second case, two values must be entered: energy on edges and energy inside(Figure 3).

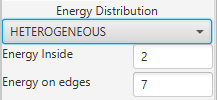


Figure 3 Heterogenous energy distribution

1. **Example of application operation**

First example of program operation, which is presented on figure 4, shows Monte Carlo grain growth. In first step, 20 iterations of algorithm were made and after that another 80. Next grain boundaries was added and picked grains was locked. After locking in the empty space new nucleons was added.

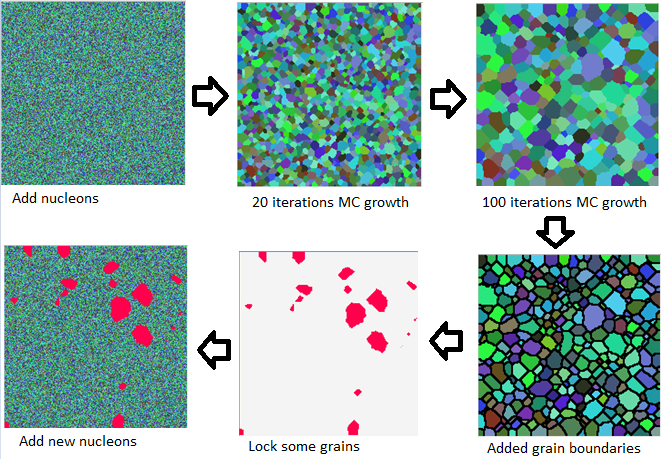


Figure 4 First example of program operation

Second program scenario presents energy distribution. The first step was to set heterogeneous type, with bigger energy on edges. Result of this process is shown on the figure 5, the same as the example of homogeneous energy distribution.

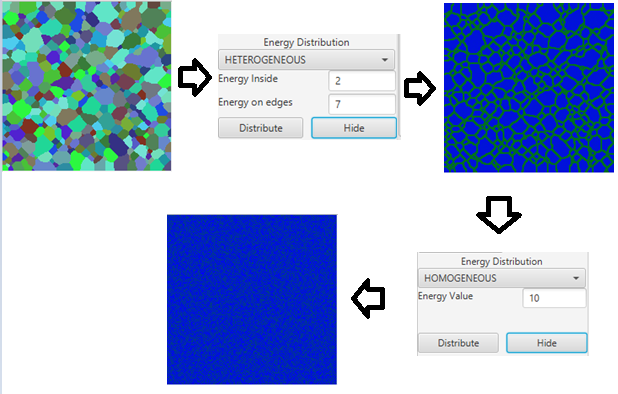


Figure 5 Second example of program operation

1. **Application versus real microstructure**

Basing on the real model of Mg AZ31(figure 6), the model of the microstructure was created by implemented application. This model is presented on figure 7.

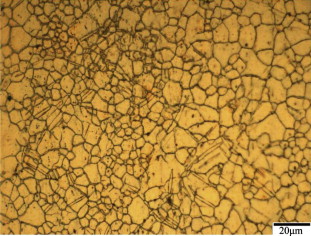


Figure 6 Picture of the microstructure of Mg AZ31 [1]

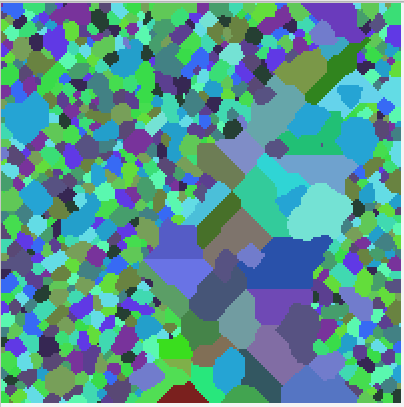


Figure 7 Simulated structure

The simulated structure does not fully reflect the real microstructure, but some similarities can be seen. Not only the strip of the bigger grains, which can be seen on real microstructure, has been achieved but also the shape of the grains on the simulated structure is similar to picture with real material.

1. **Sources**

[1] „ Effects of tool rotational and welding speed on microstructure and mechanical properties of bobbin-tool friction-stir welded Mg AZ31” W.Y. Li, T. Fu, J. Higert, 2014