Multiscale modeling		
Name Surname: Michał Słowikowski	First report	Date: 21.11.2019
Index number: 286241		Grade:

1. Introduction

In multiscale modeling Cellular Automata is a very powerful tool for creation of virtual representation of material microstructure. In this project I have developed an application for generation of such structures using C# language with .NET Framework Windows Form user interface. I have chosen this technology because of the ability to rapid prototyping both frontend and backend, it has native garbage-collection, enormous standard library with many algorithms available “right out of the box” and easy multithreading which seed up generation of microstructure. As for software I have used Visual Studio 2019 and Visual Code 2019.

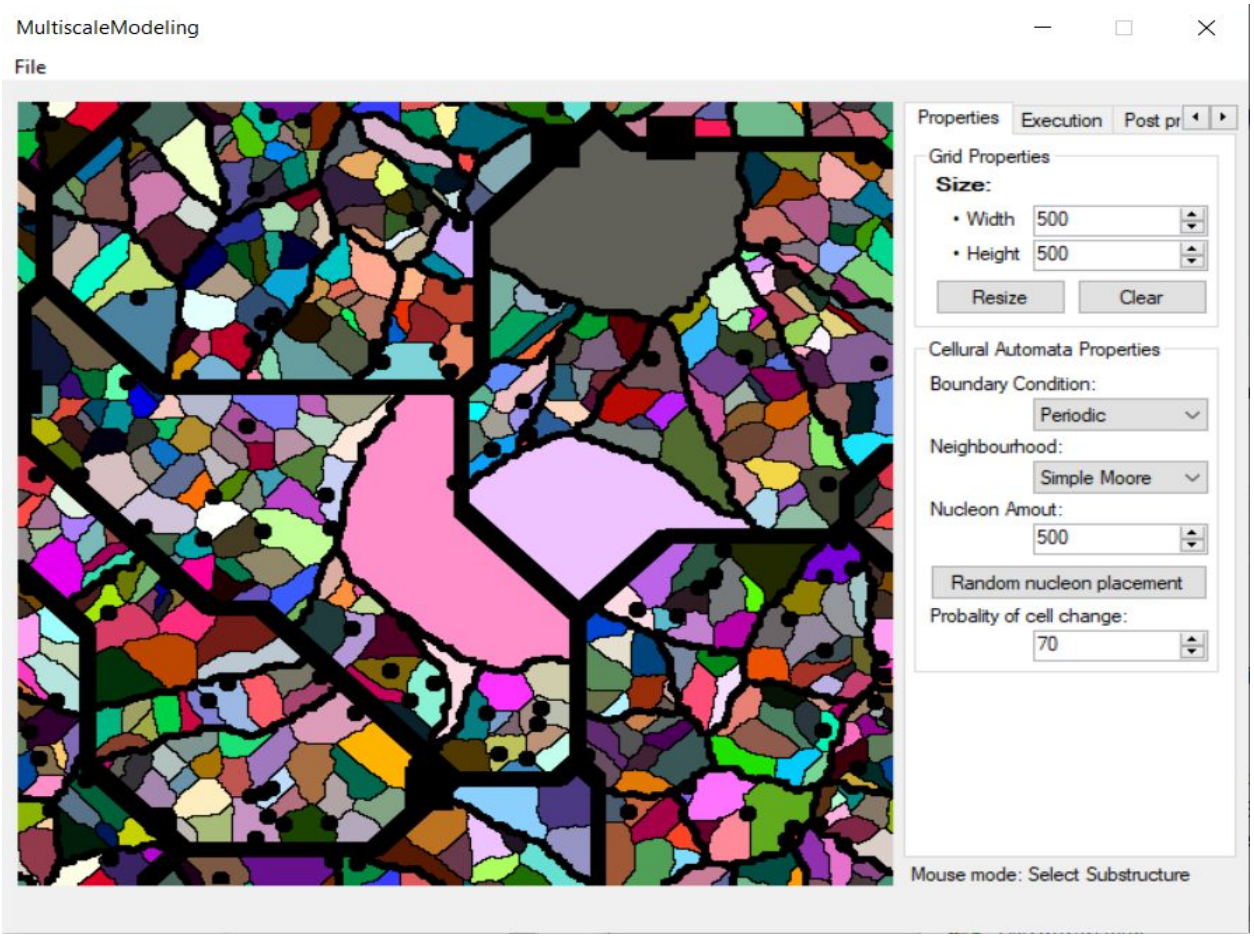
Visual Studio and Visual Code are both integrated development environments from Microsoft. Studio version is considered as one of the most powerful IDE out there. It helps with writing code by easy refactoring and great IntelliSense, it also provides robust debugger. Thanks to Form Designer, integrated into Visual Studio, it is very easy to develop functional GUI with different kinds of controls and simple way to handle a variety of types of events. Visual Studio comes with NuGet - simple package manager which automates the process of installing and updating additional libraries C#. Both Visual Studio and Visual Code support wide variety of supplementary tools and plugins. Visual Code is lightweight version of Visual Studio, and was used in situations when Visual Studio wasn’t available because of computer specification or to read long files.

Additionally I have used Newtonsoft.Json, highly efficient JSON framework for .NET. It was used for producing human-readable save which would contain all information about generated structure.

To keep track of file changes, always having at least one stable build of my program, being able to work on a project on different computers and secure code in case of bugs and necessity to restore older version I have used version control system GIT and host repository on GitHub.

2. Graphical User Interface Overview

Thanks to Windows Forms I could developed simple, but functional GUI.



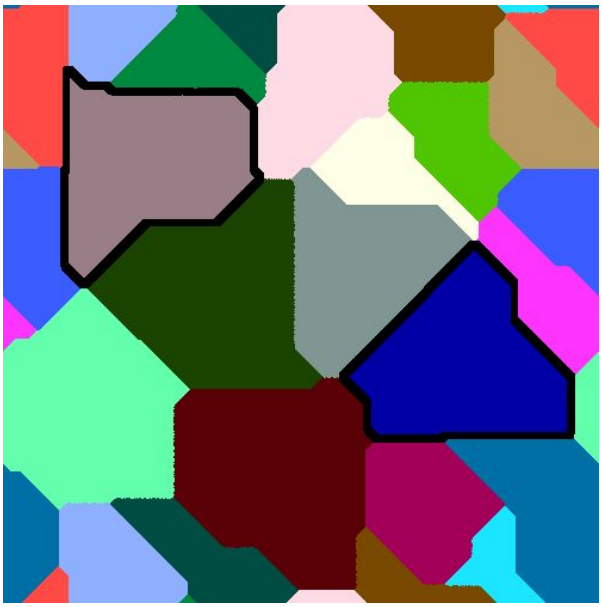
Pic.1 Main application window.

Main window [1] can be divided into two main sections. Panel which is used to set properties of generated structure is located on the right. It consists of three tabs which split options into properties set before start cellular automata simulation, execution and processing data after structure is generated. In the left part of the window user can see current result. As this viewport is interactive current mode is displayed underneath properties panel. At the top of the application window menu bar which is used for saving and loading microstructures is located.

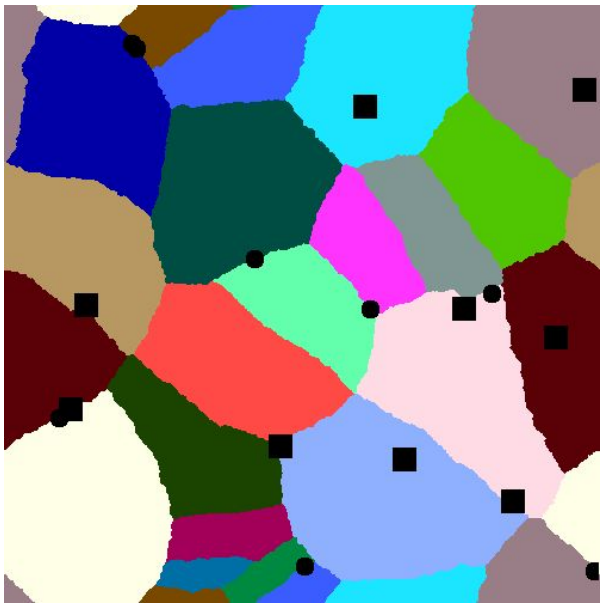
3. Features

At the beginning user can choose grid size ranging from 1x1 to 6000x6000 cells. Then in Cellular Automata Properties user can determine neighbour rules: Simple Moore [2] and Moore

with 4 rules [3], amount of nucleons, which can be placed randomly or by clicking on viewport, and finally the probability of cell change (used in Moore with 4 rules growth).

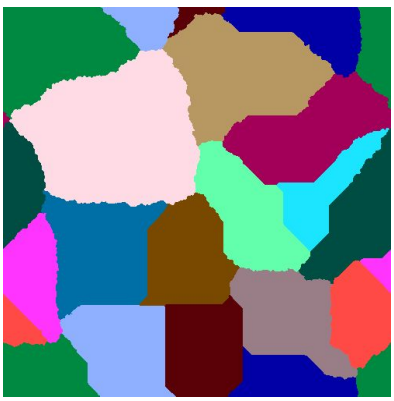


Pic. 2 Structure generated with simple Moore and two grains boundaries selected.

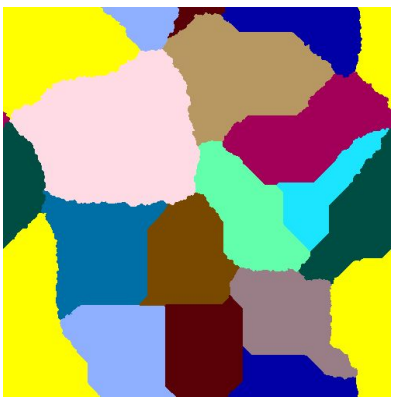


Pic. 3 Structure generated by four Moore rules with 50% probability and added inclusions. Squares was placed at the beginning, circles at the end of simulation.

On the next tab user can determine if inclusions should be added [3] (they can be placed before and after simulation), type of inclusions, their size and amount. From here, the user can also trigger start of cellular automata simulation, view progress on progress bar and stop it when he wants it.



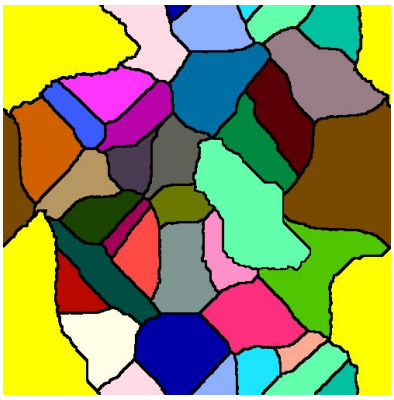
Pic. 4 First half of simulation probability was equal to 100%, second half probability was equal 10%.



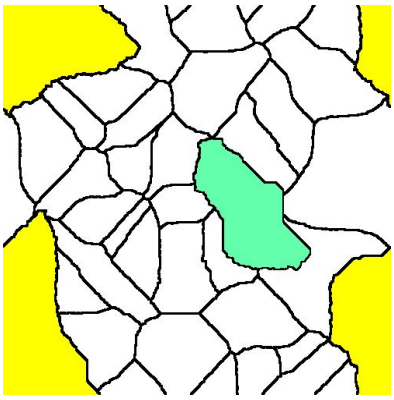
Pic 5. Next corner grains was selected as Dual-Phase (color yellow) and green grain in the middle was selected as substructure.

Pic 6. After selecting grains simulation was “soft-cleared”.

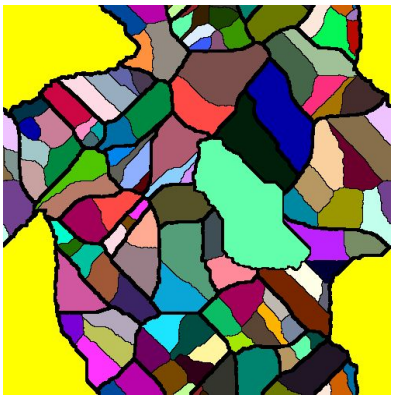
Fainal tab contains options to select and lock grains for next simulation in two ways: as dual-phase [Pic. 5] (change selected grain id to new, special id which cannot be produced by normal nucleons) and as substructure [Pic. 6] (keep id). Here user can also generate all grain boundaries[Pic. 8] or boundaries of selected grains [Pic. 9], for all grains or only selected ones, in different sizes. Inclusions, locked grains and boundaries can be left for the next CA simulation [Pic. 10].



Pic. 8 Regrowing structure from Pic 6. and selecting all grain boundaries.



Pic. 9 “Soft-clearing” Pic. 8.



Pic. 9 Regrowing structure and selecting all grain boundaries again.

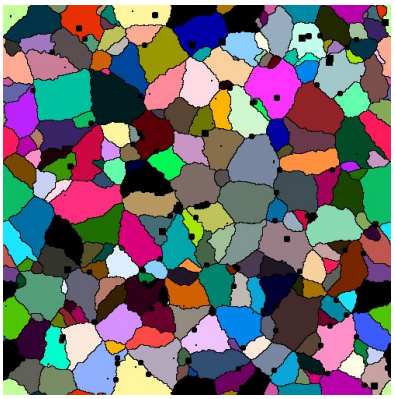
Generated structure with additional metadata can be saved into JSON [Pic. 10] and Bitmap formats (using menu bar or proper shortcuts) and later loaded into application for further processing.


```
{
    "Size": {
        "width": 500,
        "height": 500
    },
    "caProperties": {
        "nucleonAmout": 100,
        "Neighbourhood": "Moore 4 rules",
        "boundaryCondition": "Periodic"
    },
    "Grid": [
        {
            "x": 0,
            "y": 0,
            "id": -2
        },
```

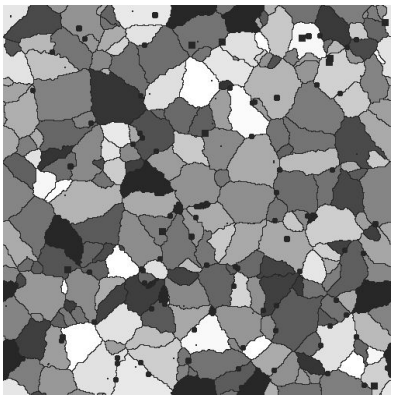

Pic. 10 Example JSON output.

4. Comparison with Low-carbon Steel

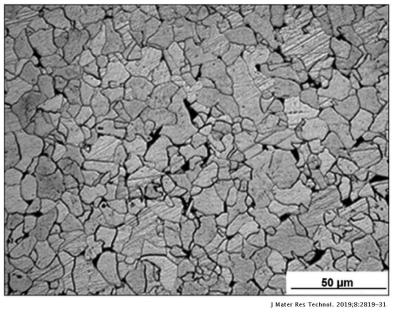
At the beginning of generating structure I have added a couple of small inclusions. Simulation started with 10% possability and 70 grains, after about ½ progress posabilities was changed to 70% and about additional 120 grains was added. After simulation I have added more inclusions and generate boundaries for all grains. Result can be seen below [Pic. 11]



Pic. 11 Generated structure.



Pic. 12 Generated structure with desaturation.



Pic. 13 Actual microstructure. Source:http://www.jmrt.com.br/en-optimization-tmcp-strategy-for-microstructure-article-S2238785419301723#imagen-2

Structure generated by application share some similarities with actual microstructure. Although not perfect I think it can be used for further simulations. Grains are a little bit to big, but they have overall “jaggy” boundaries. Ocasional inclusions can be found in the middle of a grain like in the picture [Pic. 13].

One way to get more accurate structure would be use original picture. Because of the feature of application to open bitmap file we could convert, paint over, original microstructure. This way user could get far better results. It can be also accomplished in application, but user cannot create inclusions in specific places, only randomly.

In current application biggest difficulty is to generate imperfections with complex or even oblong shapes. Also tool for multiselection would prove beneficial for user experience. Another problem is grid size restriction. It would be better to keep only part of it in RAM and load the rest of it from harddrive only if it is necessary. Somewhat of a problem is that software provides multiple options when clicking on the viewport dependent on the context during what phase user clicked it. It can be a little confusing, even with label stating what current mode is turned on and with no way to undo this can lead to very frustrating situations.

Despite minor problems with generated microstructure and application overall, developed software have all necessary components to producing different kinds of microstructures.