Bartosz Gosławski

Multiscale Modelling

1st report

1. Introduction

Simple grain growth was implemented by using Cellular Automata- discrete model studied in computer science, mathematics, physics and microstructure modelling. A CA is a model of a system of cell objects with the following characteristics:

- Grid – space where cells live (represented 2D space in the project)
- State - finite number of state possibilities (eg 1 or 0)
- Neighbourhood - closest neighbours of a particular cell which can be defined in many ways (eg Von Neumann, Moore)

The simulation of simple grain growth and GUI was implemented in C# using Visual Studio IDE.

2. Functionalities

A. Simple grain growth

B. Microstructure import/export

C. Inclusions

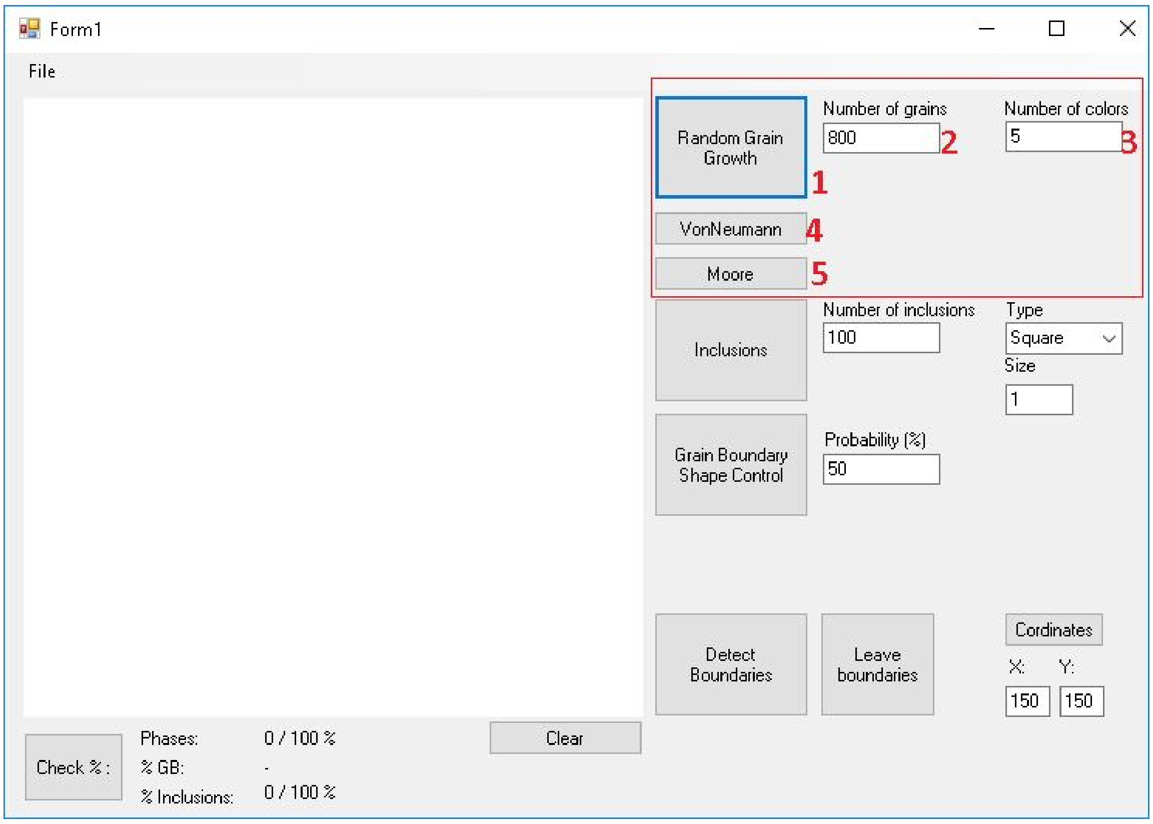
D. Moore 2

E. CA -> CA

F. Grains boundaries

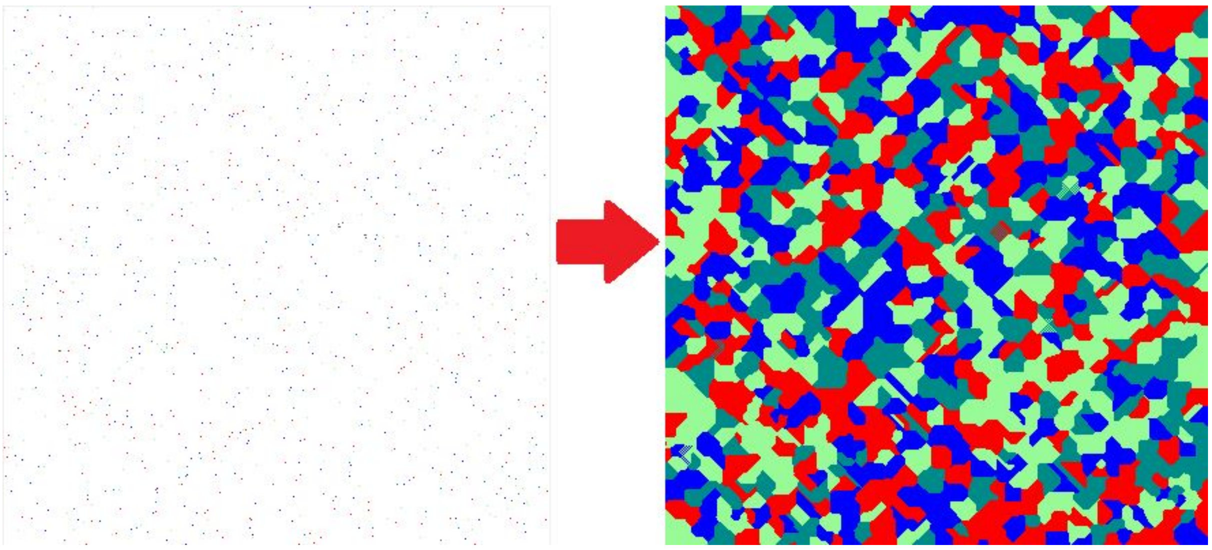
A.Simple grain growth

At the beginning (Picture 1) user need to set how many grains will be generated(2) and how many different phases will be in simulation(3). After that user have to run Random Grain Growth(1). After grains displayed there is possibility to generate whole structure based on two different neighbourhood types: VonNeuman(4) or Moore(5)



Picture 1. Main window with highlighted Simple Grain Growth options

In the example (Picture 2) was 1000 Number of grains and 4 different phases.

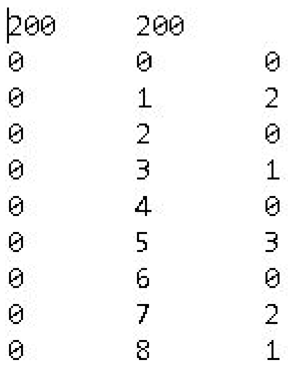


Picture 2. Successfully done Growth simulations

B. Microstructure import / export

Program have functionality to import or export 2D model in two different ways:
-as a bitmap file(Picture 4, 1) - there is no minimal picture size that program can handle with but the upper limit is 400x400 (160.000 cells)
-txt file(Picture 3, 2) with specific format:

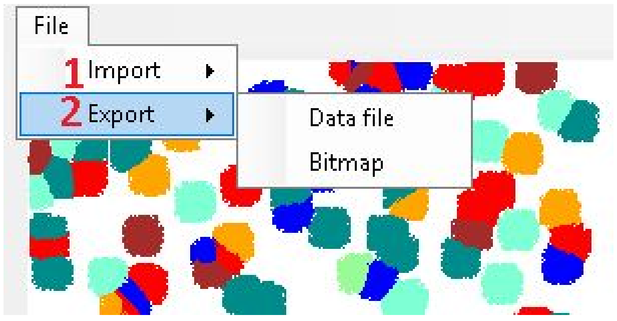
a) first row describes how many cells (or empty slots for them) are
b) then we look on whole txt file column by column rather than row by row. first columns describes coordinate x, second describes coordinate y, and third describes state of (x,y) cell.



```
200     200
0       0       0
0       1       2
0       2       0
0       3       1
0       4       0
0       5       3
0       6       0
0       7       2
0       8       1
```

Picture 3. Screenshot from txt file

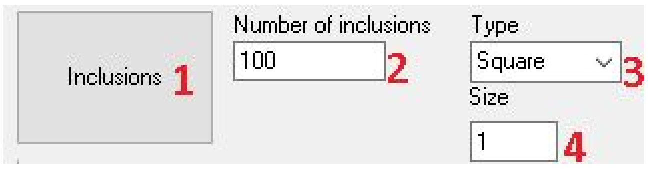
Options to bring data from txt / bmp file or save them are in drop down menu hidden in File menu on the top of program view.



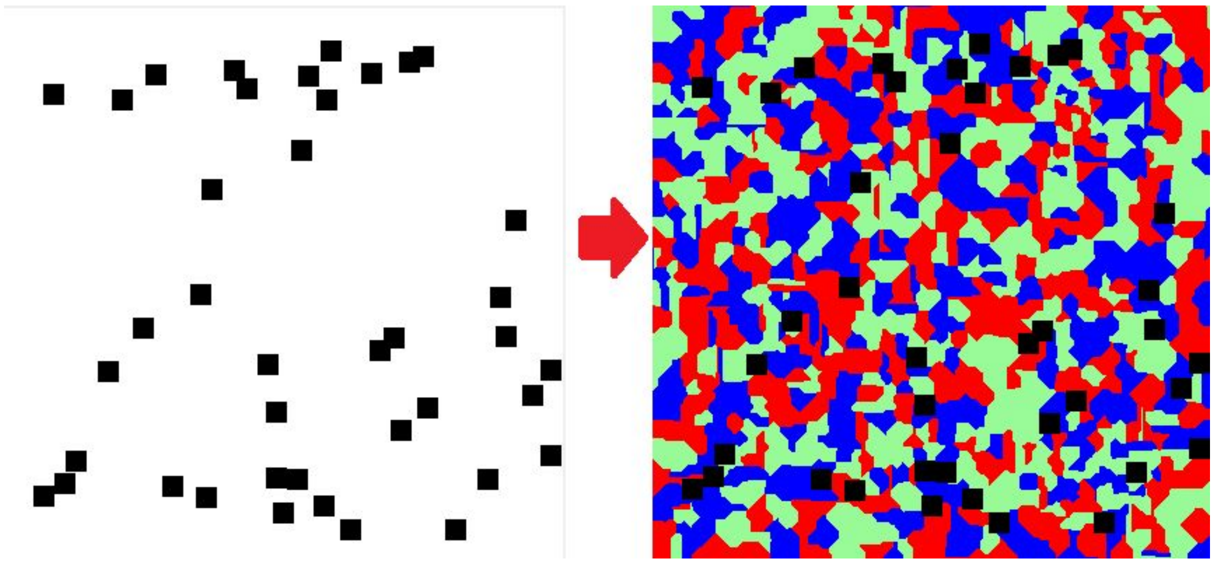
Picture 4. Import / export option

C. Inclusions

This functionality offers to add inclusions (Picture 5, 1) based on the number of inclusions (Picture 5, 2) type and their size(Picture 5, 3, 4). generating inclusions can be done before Grain Growth and also after the process. Finally the results can be seen at Picture 6.


Inclusions 1
Number of inclusions: 100 2
Type: Square 3
Size: 1 4

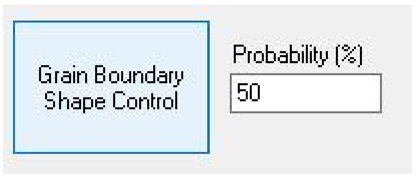
Picture 5. Properties for inclusions.



Picture 6. Inclusions done before simulation. Inclusions(40), Type(square), Size(7).

D. Moore 2

This functionality (Picture 7) is based on modification of CA grain growth algorithm - based on influence of grain curvature. Main thought here was to implement algorithm with four different principles on which grain growth was based on. If condition for fist rule in not fulfilled, algorithm checks next one (up to 4). Last rule have special input *Probability* which determines how big is chance to generate new cell based on the neighbourhood after all 3 rules are not executed.

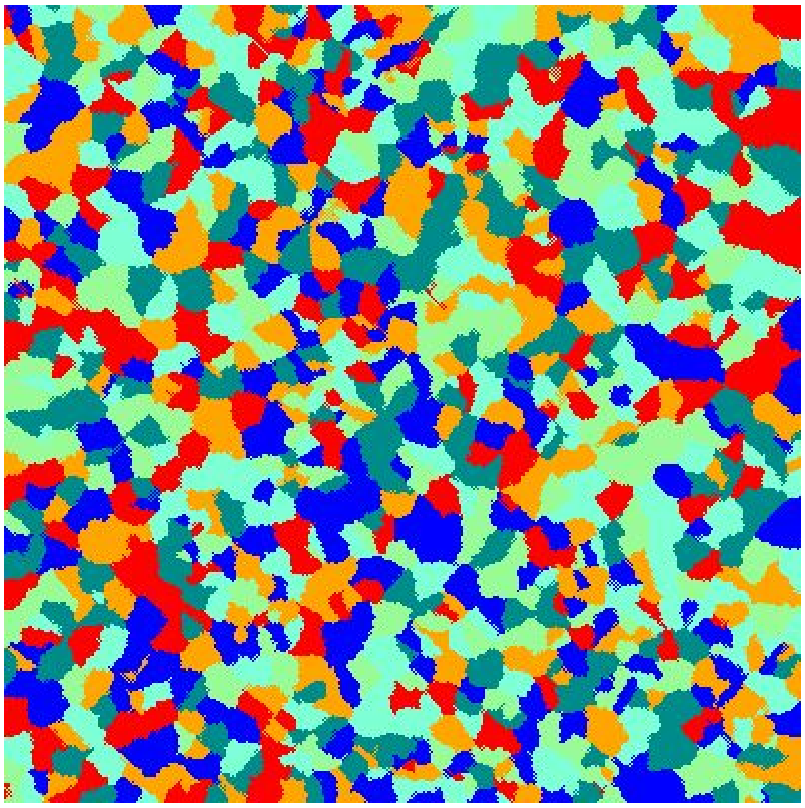


Probability (%): 50

Picture 7. Properties for Moore 2 algorithm simulation.

Ate the picture 8 can be seen successfully done Moore2 simulation with following parameters:

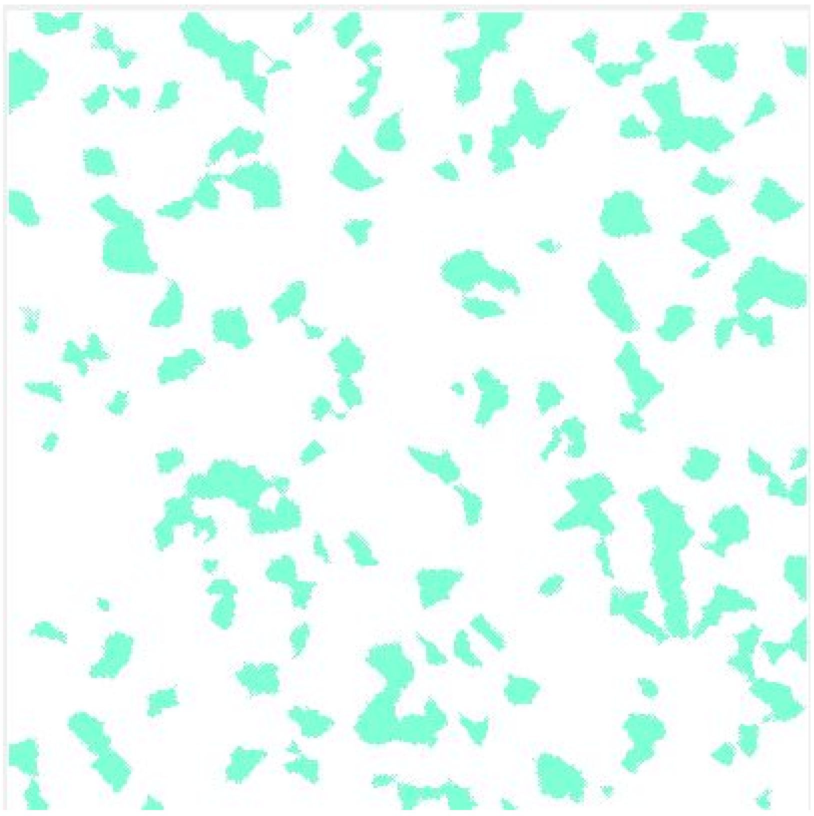
- -Random Grain Growth: Grains(1200),
- -Phases(6),
- -Probability(50%).


Picture 8. Moore 2 simulation

E. CA -> CA

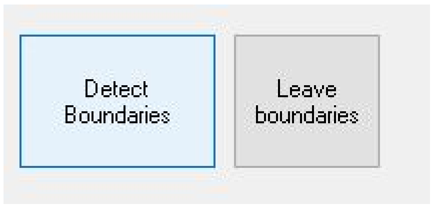
In this module main task was to give the possibility for the user to select one type of grains and separate them from the rest. After implementation this process is extremely easy and it comes to click left mouse button on selected grain color (Picture 9).

If selection is done and only one phase is on the screen - we can continue to do other options like Random Grain Growth or VonNeumann fulfillment.

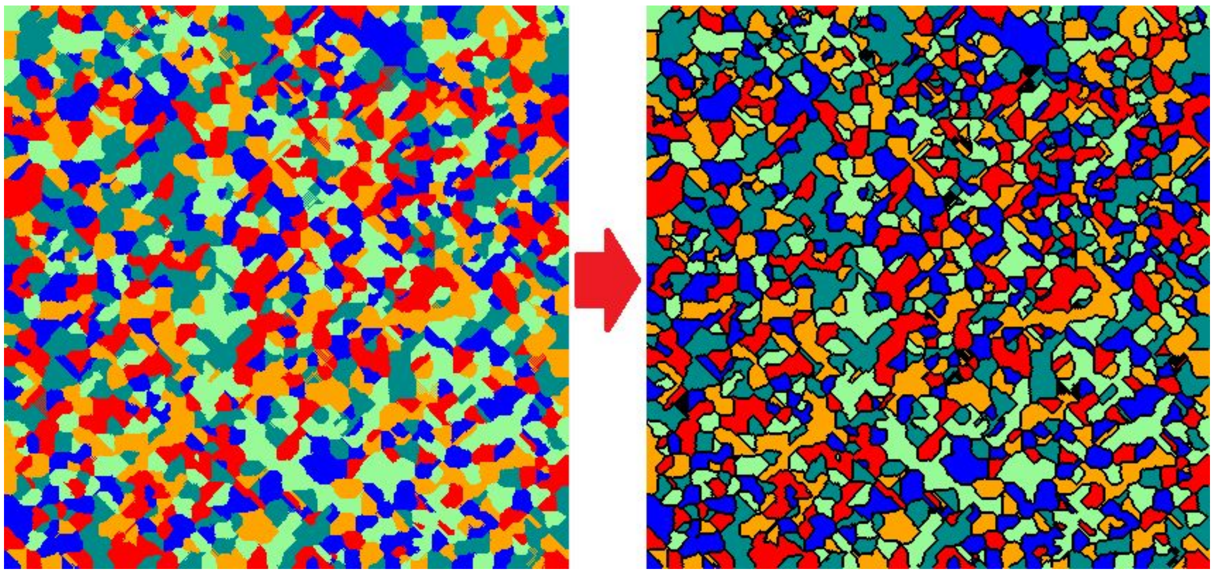

Picture 9. Model from Picture 8 with selected green grains.

F. Grain Boundaries

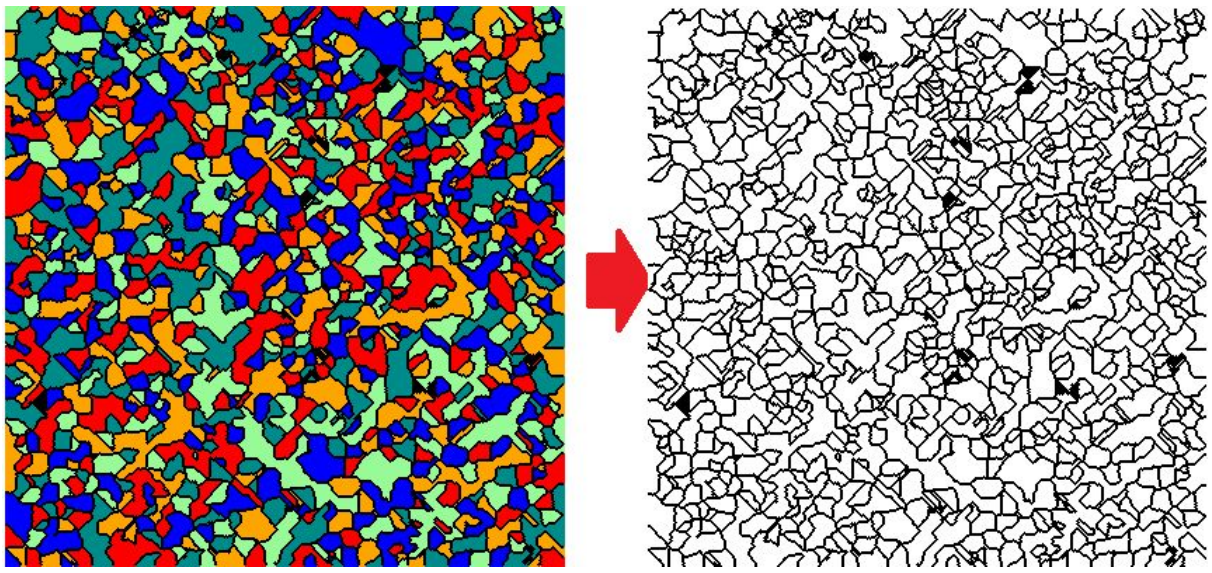
Last functionality given to the user allows to detect boundaries(Picture 10) between different grains(Picture 11). Algorithm detect which two Cells have different state and mark this boundary on black color. After that user can click *Leave boundaries* to clear all Cells from space and leave only boundaries (Picture 12).



Picture 10. Grain Boundaries buttons



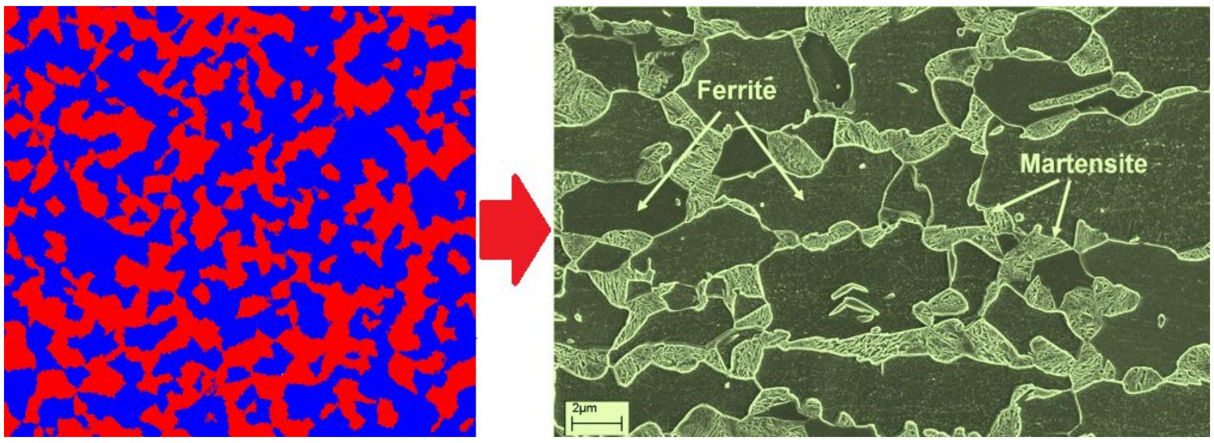
Picture 11. 2D space after Detect Boundaries



Picture 12. 2D space after Leave Boundaries

3. Comparison

Picture 13 presents a comparison between microstructure generated using the program and the microstructure of a dual phase steel. As it could be seen there are many similarities but there are also some shortcomings. Microstructure created in the program have grains in less clusters but in bigger size. Real dual phase microstructure have more clusters but their are mostly separated from each other.



Picture 13. Comparison of a generated microstructure and real dual phase microstructure source: *http://ispatguru.com/dual-phase-steels/*

4. Conclusions

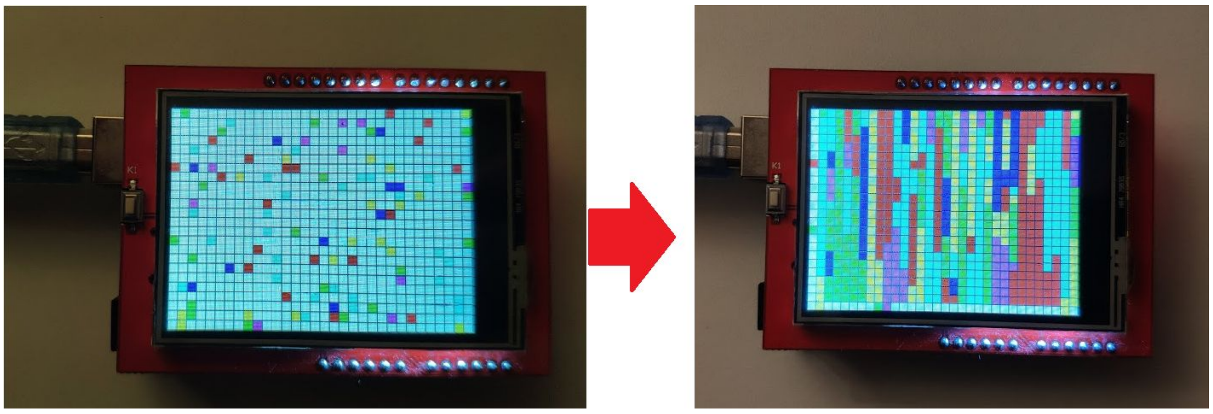
In the project report was presented how application is working, what are the functionalities based on given algorithms schemas, and how the GUI is looking. The main thought of Cellular Automata is fulfilled and all of the guidelines are done:
-Simple grain growth,
-Microstructure import/export
-Inclusions,
-Moore2,
-CA->CA,
-Grain boundaries.

As long as I tested the application I did not detect any errors. Additionally the results are quite satisfying

5. Addition

For now I am working on Arduino Uno project which includes randomly generating points and making simulation based on neighbourhood. In the project I use 2,4" screen attached to TFT LCD SHIELD (Picture 14).
(przyp. https://allegro.pl/oferta/wyswietlacz-ekran-dotykowy-lcd-tft-2-4-do-arduino-7737077037).

As you can see in the picture below, the work is in progress, and the main problem that I have to face is finding a solution for the small amount of memory in Arduino microcontroller::
Flash 32k bytes (of which .5k is used for the bootloader)
SRAM   2k bytes
EEPROM 1k byte.
In desktop version of the project I use thousands time more memory, so writing algorithms is not only contained to rewrite it from C# to arduino(C/C++), but write it again in totally different way.



Picture 14. Example of Random cell generation and Simple neighbourhood algorithm