

Graphical User Interface

Program's graphical user interface looks as follows:

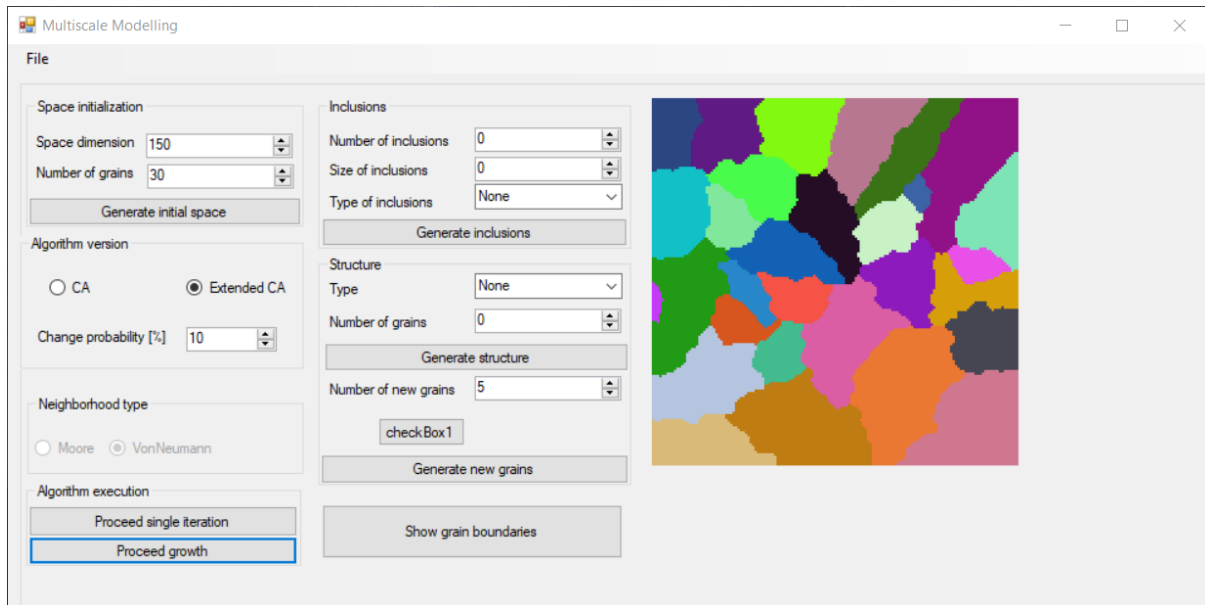


Figure 1: GUI

In order to proceed with any algorithm one needs to generate space of desired size with inputted number of grains:

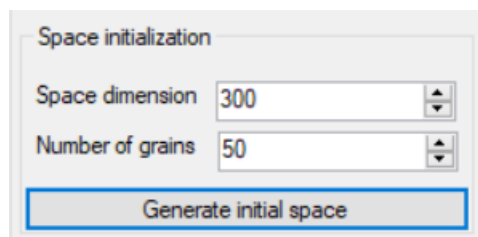


Figure 2: Space initialization group box

Class 1 - Simple grain growth

During first class the task was to implement basic version of Cellular Automata algorithm with utilizing classic Moore and VonNeumann neighborhood types. In order to ensure correct algorithm operation absorbing boundary condition was introduced. The state of cells on the edges of solution space were fixed with a specific value (0 in particular), thus they are not taken into account when updating grain structure.

In algorithm version group box **CA** radio button must be checked:

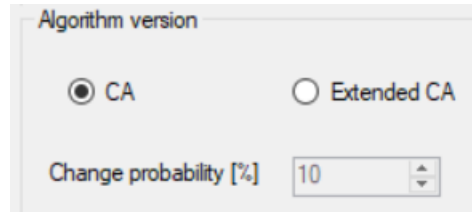


Figure 3: Simple CA

After the growth utilizing simple CA is complete a space such as the one presented below is produced.



Figure 4: Example of simple CA algorithm

Class 2 - Microstructure import/export

Second class goal was to implement mechanism of importing/exporting grain structure from/to file. In order to do so one shall access **File->Export data** or **File->Import data** depending on intentions.

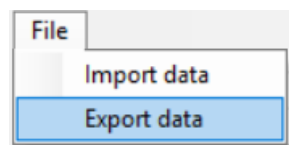


Figure 5: Import/export menu

0.1 Exporting state

State is saved simultaneously in a bitmap and in a text file. User is able to choose the location where the files are supposed to be stored. Name for each file is generated

automatically and is unique, because of the fact that it includes current date.

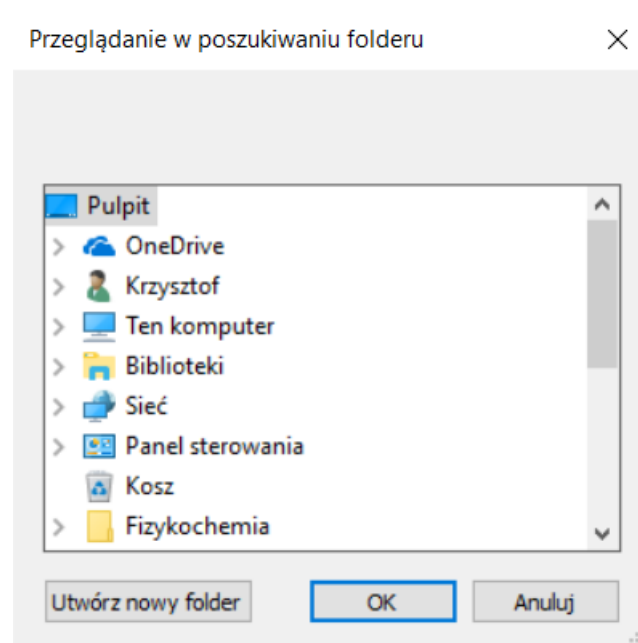


Figure 6: Export dialog

0.1.1 Exporting state to bitmap

0.1.2 Exporting state to .txt file

0.2 Importing state

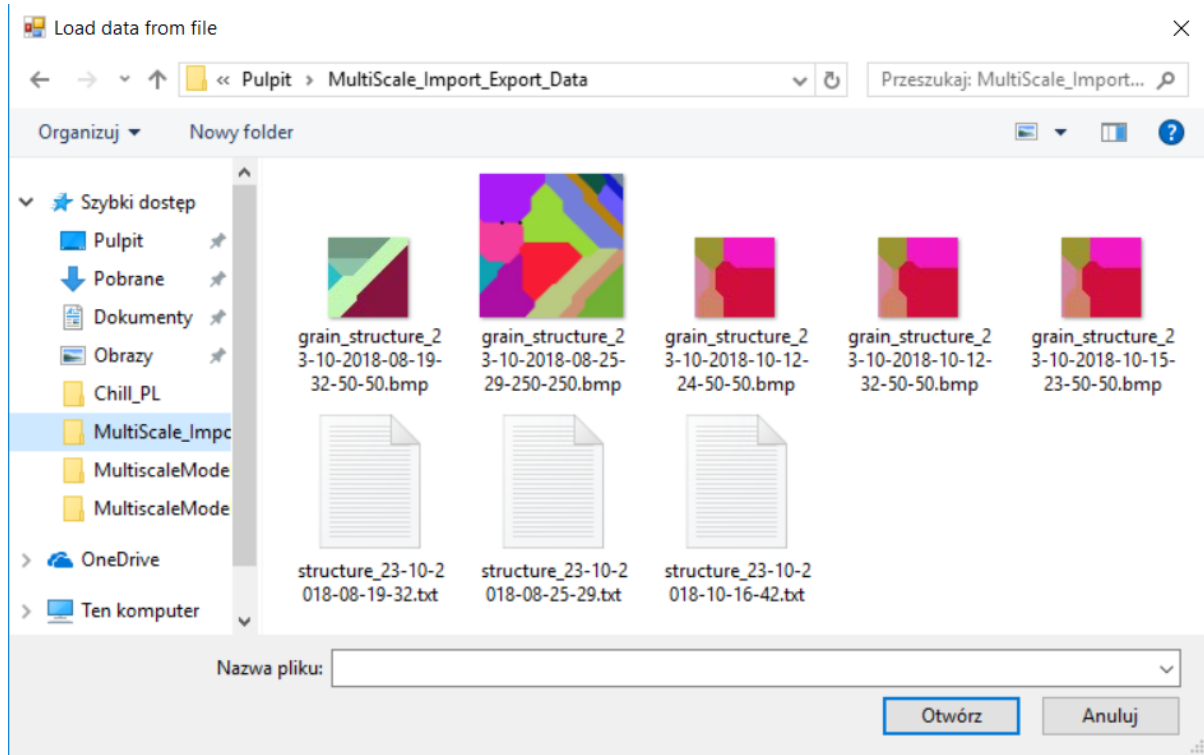


Figure 7: Import dialog

0.2.1 Importing state from .txt file

The format of txt file is as follows:

1. Dimension of space
2. Header
3. Values corresponding to header's columns

Example of such file:

1	50					
2	posx	posy	ID	Phase	Color	
3	0	0	5	0	Color [A=255, R=116, G=153, B=130]	
4	0	1	5	0	Color [A=255, R=116, G=153, B=130]	
5	0	2	5	0	Color [A=255, R=116, G=153, B=130]	
6	0	3	5	0	Color [A=255, R=116, G=153, B=130]	
7	0	4	5	0	Color [A=255, R=116, G=153, B=130]	
8	0	5	5	0	Color [A=255, R=116, G=153, B=130]	
9	0	6	5	0	Color [A=255, R=116, G=153, B=130]	
10	0	7	5	0	Color [A=255, R=116, G=153, B=130]	
11	0	8	5	0	Color [A=255, R=116, G=153, B=130]	

Figure 8: structure_23-10-2018-08-19-32.txt

0.2.2 Importing state from bitmap



Figure 9: grain_structure_23-10-2018-08-25-29-250-250.bmp

Class 3 - Inclusions

Aim of next class was to introduce a feature which would allow user to add inclusions to grain structure either before the growth (in any place/coordinates) or after the growth (only on grain borders). Inclusions are defined with following parameters:

- number - how many inclusions to create
- size - what is the diameter (circular)/ side (square) of a single inclusion
- type - what type of inclusions to create (circular/square)

A callback function is connected to structure display space, so that user can click and add inclusions of his/her choosing before the growth starts and a dedicated button is responsible for generating inclusions specified with parameters in corresponding group box is also available.

Any action will take place only when type of inclusion is changed to a valid one (from

default None) and automatic generation will take place only when number of inclusions is positive, as well as, their size.

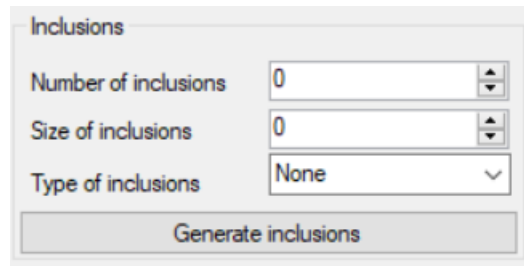
A software interface window titled "Inclusions". It contains three input fields: "Number of inclusions" with a value of 0, "Size of inclusions" with a value of 0, and "Type of inclusions" with a dropdown menu showing "None". Below these fields is a button labeled "Generate inclusions".

Figure 10: Inclusions group box

Example of structure with circular inclusions:



Figure 11: Example of circular inclusions

Example of structure with square inclusions:



Figure 12: Example of square inclusions

Class 4 - Extended CA

Fourth class purpose was to implement extended version of Cellular Automata algorithm. Instead of one rule used in basic four rules were implemented, featuring new types of neighborhoods, as well as, new conditions for changing state of a particular cell.

In order to take advantage of extended version of the algorithm it must be checked on algorithm version group box:

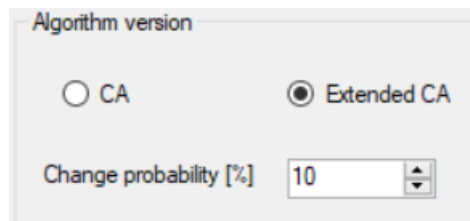


Figure 13: Extended CA

Then and only then probability parameter for 4th rule is unlocked for modification.

After the growth utilizing extended CA is complete a space such as the one presented below is produced.

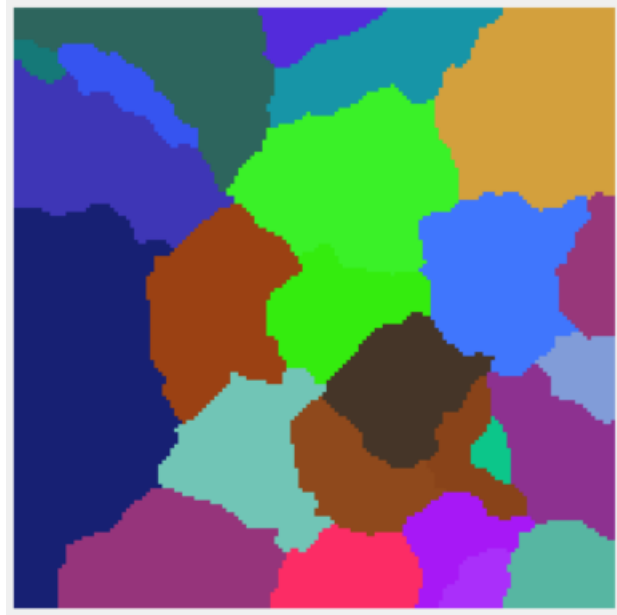


Figure 14: Example of extended CA algorithm

Class 5 - Substructures/Dual phase

In course of class number 5 we were supposed to give user possibility to create a substructure or dual phase after initial growth. User can choose how many grains from initial growth are meant to be considered while creating substructure of dual phase. After those grains are "transformed" according to user preference new grains can be added and once more growth is possible.

0.3 Substructure

User defined number of grains is preserved on the space with their original color and IDs.



Figure 15: Example of a substructur

0.4 Dual phase

User defined number of grains is preserved on the space with a new color common for all of them and their phase parameter is incremented.



Figure 16: Example of dual phase

Class 6 - Grain boundaries

After the growth with chosen algorithm version is complete user can generate grain boundaries and display them.

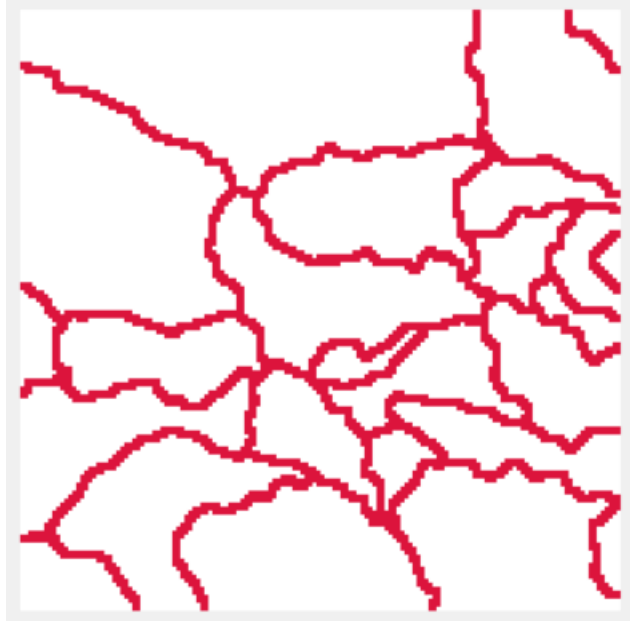


Figure 17: Example grain boundaries

Attachments

References

- [1] Fred G. Martin *Robotics Explorations: A Hands-On Introduction to Engineering*. New Jersey: Prentice Hall.
- [2] Flueck, Alexander J. 2005. *ECE 100*[online]. Chicago: Illinois Institute of Technology, Electrical and Computer Engineering Department, 2005 [cited 30 August 2005]. Available from World Wide Web: (<http://www.ece.iit.edu/~flueck/ece100>).