Piotr Adryan

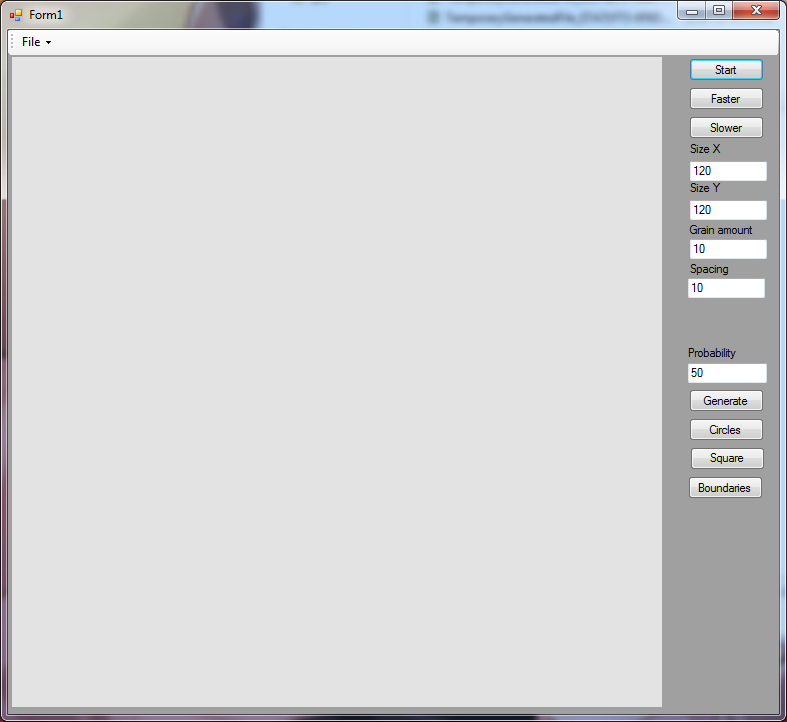
Informatyka Stosowana, IO

**Report 1**

The goal of the project: To create a simulation of Grain Growth using Moore neighborhood

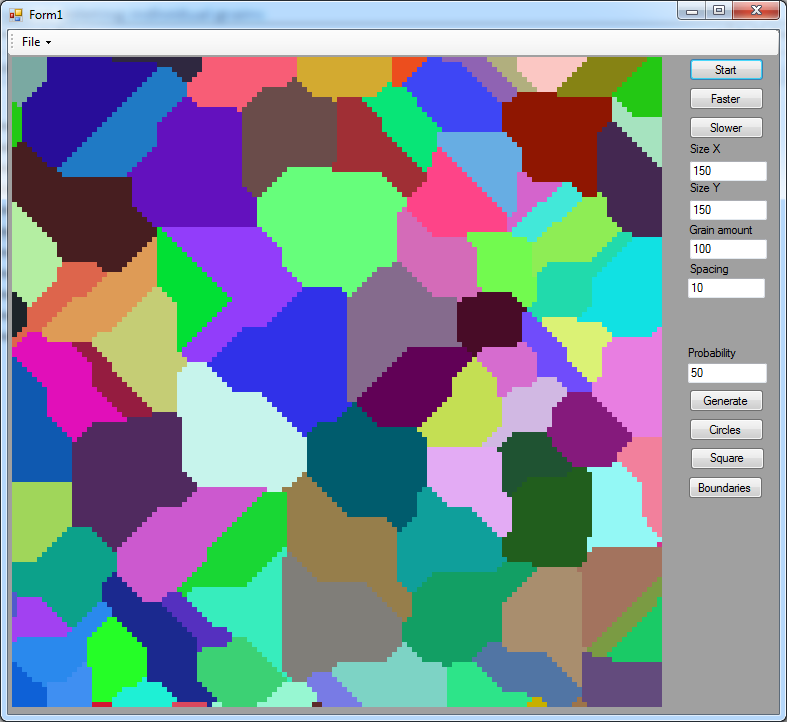
Part 1: The interface:

The options included in the interface:

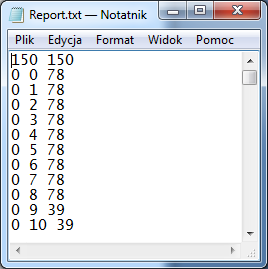
* The size of the simulation space
* the number of generated grains
* the speed of the simulation
* square and circle inclusions (only after the growth)
* drawing the boundaries of the grains after growth
* Importing and exporting the current state of the simulation to .txt and .bmp files
* Selecting and deleting individual grains

Task 1: Choosing neighborhood and boundary type and implementing transition rules

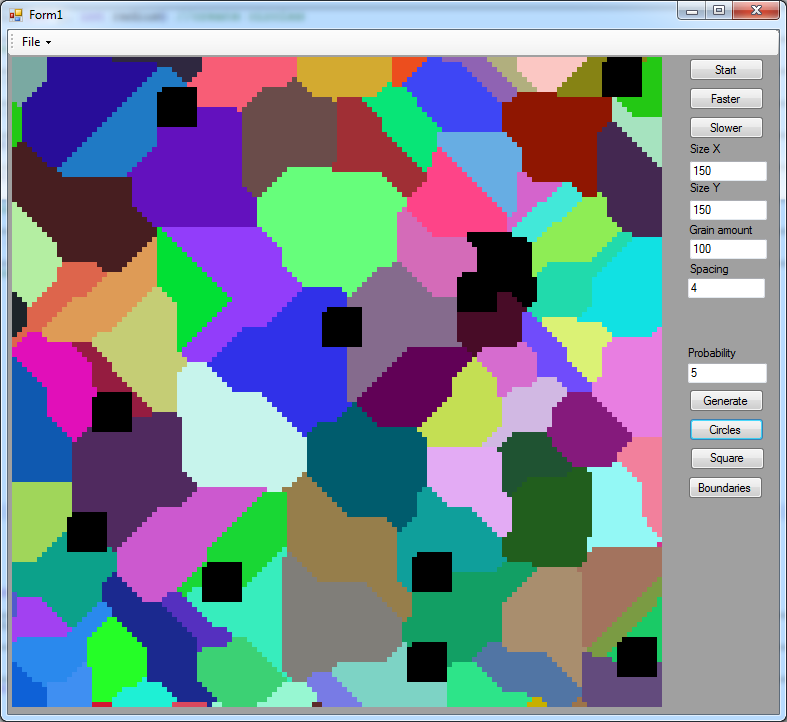
This task encompasses the initial implementation of the algorithm, where a decision was taken upon which of the several types of neighborhoods would be chosen, along with the type of boundary conditions. The program takes data from the interface to create a workspace, and generates a defined number of grains that are placed randomly on the board, after which they start growing.



Task 2: The import and export mechanics

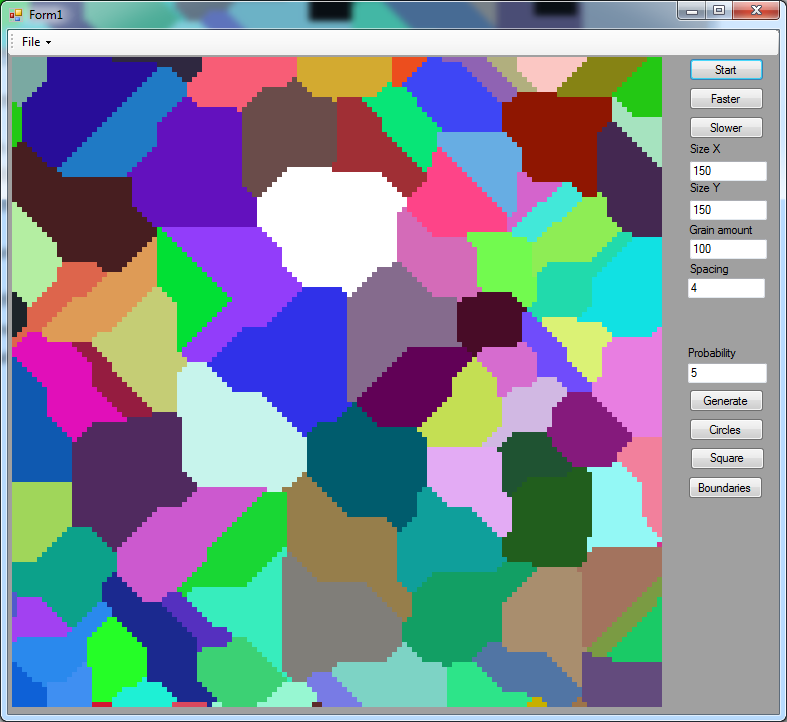
At any time during the simulation we can pause it, and export the data into a .txt or .bmp file. The .txt file contains the data by containing the size of the workspace in the first line, and then each line contains data about another space on the workspace in the format of x coordinate, y coordinate and the id of the grain currently within that space, with value 0 representing empty space. The .bmp is simply storing the visual data of the simulation, as it looks in the program. You can import the data from the files at a later time and the simulation will continue from there.

Task 3: Inclusions

When all the grains grow as much as they could and the simulation ends, you can insert inclusions into the workspace, in shape of either a circle, or a square, with the size of inclusion being configurable from the interface. The inclusions will always be generated on the boundary between the grains and their number is randomized.

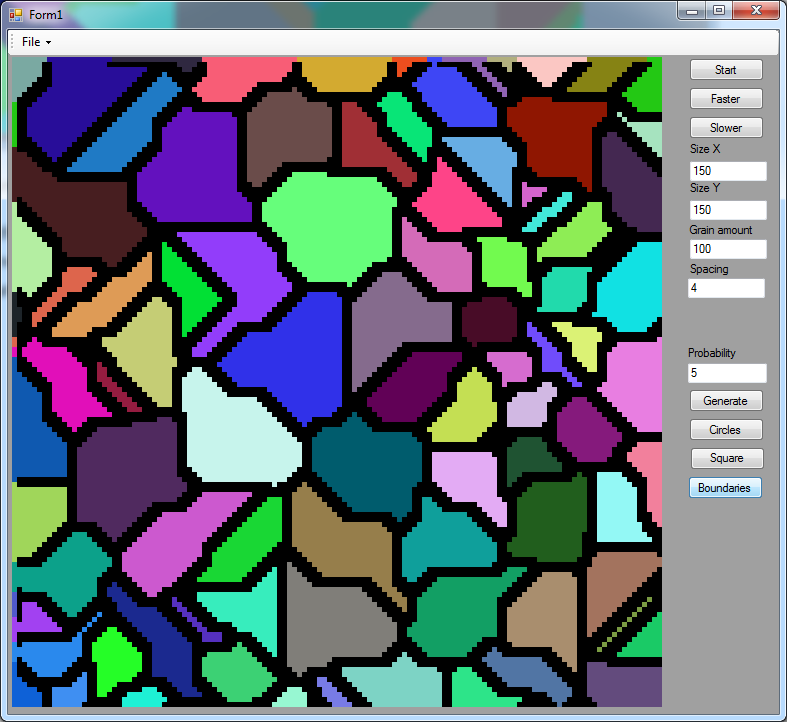
Task 4: Substructure of the Cellular Automata

After the growth of the grains you can click with your mouse on any chosen grain, causing it to be deleted. This will create additional space for other grains to grow in there. The algorithm checks the id of the cell at the point user clicked on and sets all other cells containing a matching id to 0.



Task 5: Grain boundaries

When the growth simulation is finished you can choose to draw the boundaries between the grains. The algorithm creates a list of all cells which have at least 2 different neighbors and later paints all the points in that list, creating the boundaries between grains.



Conclusions:

The algorithm can be useful for the purposes of simulating the behavior of real metallic structures, and is much faster than conducting testing in laboratory on a real sample. Additionally, the tests can be recreated, with changed conditions, which is impossible with real samples.