

AutoCell

Generated by Doxygen 1.8.14

Contents

1	Main Page	1
2	Presentation	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Class Documentation	11
6.1	Cell Class Reference	11
6.1.1	Detailed Description	12
6.1.2	Constructor & Destructor Documentation	12
6.1.2.1	Cell()	12
6.1.3	Member Function Documentation	12
6.1.3.1	addNeighbour()	12
6.1.3.2	forceState()	13
6.1.3.3	getNeighbour()	13
6.1.3.4	getNeighbours()	13
6.1.3.5	getRelativePosition()	14
6.1.3.6	getState()	14
6.1.3.7	setState()	14

6.1.3.8	validState()	15
6.1.4	Member Data Documentation	15
6.1.4.1	m_neighbours	15
6.1.4.2	m_nextState	15
6.1.4.3	m_state	15
6.2	CellHandler Class Reference	16
6.2.1	Detailed Description	17
6.2.2	Member Typedef Documentation	17
6.2.2.1	const_iterator	17
6.2.2.2	iterator	17
6.2.3	Member Enumeration Documentation	17
6.2.3.1	generationTypes	17
6.2.4	Constructor & Destructor Documentation	18
6.2.4.1	CellHandler() [1/2]	18
6.2.4.2	CellHandler() [2/2]	18
6.2.4.3	~CellHandler()	19
6.2.5	Member Function Documentation	19
6.2.5.1	begin() [1/2]	19
6.2.5.2	begin() [2/2]	20
6.2.5.3	end()	20
6.2.5.4	foundNeighbours()	20
6.2.5.5	generate()	20
6.2.5.6	getCell()	21
6.2.5.7	getDimensions()	21
6.2.5.8	getListNeighboursPositions()	21
6.2.5.9	getListNeighboursPositionsRecursive()	22
6.2.5.10	load()	23
6.2.5.11	nextStates()	23
6.2.5.12	positionIncrement()	24
6.2.5.13	print()	25

6.2.5.14	save()	25
6.2.6	Member Data Documentation	26
6.2.6.1	m_cells	26
6.2.6.2	m_dimensions	26
6.3	CreationDialog Class Reference	26
6.3.1	Detailed Description	27
6.3.2	Constructor & Destructor Documentation	27
6.3.2.1	CreationDialog()	27
6.3.3	Member Function Documentation	28
6.3.3.1	createGenButtons()	28
6.3.3.2	processSettings	28
6.3.3.3	settingsFilled	28
6.3.4	Member Data Documentation	28
6.3.4.1	m_densityBox	29
6.3.4.2	m_dimensionsEdit	29
6.3.4.3	m_doneBt	29
6.3.4.4	m_empGen	29
6.3.4.5	m_groupBox	29
6.3.4.6	m_randGen	30
6.3.4.7	m_stateMaxBox	30
6.3.4.8	m_symGen	30
6.4	CellHandler::iteratorT< CellHandler_T, Cell_T > Class Template Reference	30
6.4.1	Detailed Description	31
6.4.2	Constructor & Destructor Documentation	31
6.4.2.1	iteratorT()	31
6.4.3	Member Function Documentation	32
6.4.3.1	changedDimension()	32
6.4.3.2	operator!=()	32
6.4.3.3	operator*()	32
6.4.3.4	operator++()	33

6.4.3.5	operator->()	33
6.4.4	Friends And Related Function Documentation	33
6.4.4.1	CellHandler	33
6.4.5	Member Data Documentation	33
6.4.5.1	m_changedDimension	33
6.4.5.2	m_finished	34
6.4.5.3	m_handler	34
6.4.5.4	m_position	34
6.4.5.5	m_zero	34
6.5	MainWindow Class Reference	35
6.5.1	Detailed Description	36
6.5.2	Constructor & Destructor Documentation	37
6.5.2.1	MainWindow()	37
6.5.3	Member Function Documentation	37
6.5.3.1	closeTab	37
6.5.3.2	createActions()	37
6.5.3.3	createBoard()	38
6.5.3.4	createIcons()	38
6.5.3.5	createTab()	38
6.5.3.6	createTabs()	38
6.5.3.7	createToolBar()	39
6.5.3.8	forward	39
6.5.3.9	getBoard()	39
6.5.3.10	nextState()	39
6.5.3.11	openCreationWindow	40
6.5.3.12	openFile	40
6.5.3.13	saveToFile	40
6.5.3.14	setCellHandler	40
6.5.3.15	updateBoard()	41
6.5.4	Member Data Documentation	41

6.5.4.1	m_boardHSize	41
6.5.4.2	m_boardVSize	41
6.5.4.3	m_cellHandlers	41
6.5.4.4	m_cellSize	42
6.5.4.5	m_fastBackward	42
6.5.4.6	m_fastBackwardBt	42
6.5.4.7	m_fastBackwardIcon	42
6.5.4.8	m_fastForward	42
6.5.4.9	m_fastForwardBt	43
6.5.4.10	m_fastForwardIcon	43
6.5.4.11	m_jumpSpeed	43
6.5.4.12	m_newAutomaton	43
6.5.4.13	m_newAutomatonBt	43
6.5.4.14	m_newIcon	44
6.5.4.15	m_nextState	44
6.5.4.16	m_nextStateBt	44
6.5.4.17	m_openAutomaton	44
6.5.4.18	m_openAutomatonBt	44
6.5.4.19	m_openIcon	45
6.5.4.20	m_pauseIcon	45
6.5.4.21	m_playIcon	45
6.5.4.22	m_playPause	45
6.5.4.23	m_playPauseBt	45
6.5.4.24	m_previousState	46
6.5.4.25	m_previousStateBt	46
6.5.4.26	m_resetAutomaton	46
6.5.4.27	m_resetBt	46
6.5.4.28	m_resetIcon	46
6.5.4.29	m_saveAutomaton	47
6.5.4.30	m_saveAutomatonBt	47
6.5.4.31	m_saveIcon	47
6.5.4.32	m_speedLabel	47
6.5.4.33	m_tabs	47
6.5.4.34	m_toolBar	47

7 File Documentation	49
7.1 cell.cpp File Reference	49
7.2 cell.cpp	49
7.3 cell.h File Reference	50
7.4 cell.h	50
7.5 cellhandler.cpp File Reference	50
7.6 cellhandler.cpp	51
7.7 cellhandler.h File Reference	56
7.8 cellhandler.h	56
7.9 creationdialog.cpp File Reference	57
7.10 creationdialog.cpp	57
7.11 creationdialog.h File Reference	58
7.12 creationdialog.h	59
7.13 main.cpp File Reference	59
7.13.1 Function Documentation	59
7.13.1.1 main()	60
7.14 main.cpp	60
7.15 mainwindow.cpp File Reference	60
7.16 mainwindow.cpp	60
7.17 mainwindow.h File Reference	64
7.18 mainwindow.h	64
7.19 presentation.md File Reference	65
7.20 presentation.md	65
7.21 README.md File Reference	65
7.22 README.md	65
Index	67

Chapter 1

Main Page

To generate the Documentation, go in Documentation directory and run `make`.

It will generate html doc (in `output/html/index.html`) and latex doc (pdf output directly in Documentation directory (`docPdf.pdf`)).

Chapter 2

Presentation

What is AutoCell

The purpose of this project is to create a Cellular Automate Simulator.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Cell	11
CellHandler	16
CellHandler::iteratorT< CellHandler_T, Cell_T >	30
QDialog	
CreationDialog	26
QMainWindow	
MainWindow	35

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Cell	Contains the state, the next state and the neighbours	11
CellHandler	Cell container and cell generator	16
CreationDialog	Automaton creation dialog box	26
CellHandler::iteratorT< CellHandler_T, Cell_T >	Implementation of iterator design pattern with a template to generate iterator and const_iterator at the same time	30
MainWindow	Simulation window	35

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

cell.cpp	49
cell.h	50
cellhandler.cpp	50
cellhandler.h	56
creationdialog.cpp	57
creationdialog.h	58
main.cpp	59
mainwindow.cpp	60
mainwindow.h	64

Chapter 6

Class Documentation

6.1 Cell Class Reference

Contains the state, the next state and the neighbours.

```
#include <cell.h>
```

Public Member Functions

- [Cell](#) (unsigned int state=0)
Constructs a cell with the state given. State 0 is dead state.
- void [setState](#) (unsigned int state)
Set temporary state.
- void [validState](#) ()
Validate temporary state.
- void [forceState](#) (unsigned int state)
Force the state change.
- unsigned int [getState](#) () const
Access current cell state.
- bool [addNeighbour](#) (const [Cell](#) *neighbour, const QVector< short > relativePosition)
Add a new neighbour to the [Cell](#).
- QMap< QVector< short >, const [Cell](#) * > [getNeighbours](#) () const
Access neighbours list.
- const [Cell](#) * [getNeighbour](#) (QVector< short > relativePosition) const
Get the neighbour asked. If not existent, return nullptr.

Static Public Member Functions

- static QVector< short > [getRelativePosition](#) (const QVector< unsigned int > cellPosition, const QVector< unsigned int > neighbourPosition)
Get the relative position, as neighbourPosition minus cellPosition.

Private Attributes

- unsigned int `m_state`
Current state.
- unsigned int `m_nextState`
Temporary state, before validation.
- QMap< QVector< short >, const Cell * > `m_neighbours`
Cell's neighbours. Key is the relative position of the neighbour.

6.1.1 Detailed Description

Contains the state, the next state and the neighbours.

Definition at line 10 of file `cell.h`.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Cell()

```
Cell::Cell (
    unsigned int state = 0 )
```

Constructs a cell with the state given. State 0 is dead state.

Parameters

<code>state</code>	Cell state, dead state by default
--------------------	-----------------------------------

Definition at line 7 of file `cell.cpp`.

6.1.3 Member Function Documentation

6.1.3.1 addNeighbour()

```
bool Cell::addNeighbour (
    const Cell * neighbour,
    const QVector< short > relativePosition )
```

Add a new neighbour to the Cell.

Parameters

<code>relativePosition</code>	Relative position of the new neighbour
<code>neighbour</code>	New neighbour

Returns

False if the neighbour already exists

Definition at line 60 of file [cell.cpp](#).

References [m_neighbours](#).

6.1.3.2 forceState()

```
void Cell::forceState (
    unsigned int state )
```

Force the state change.

Is equivalent to setState followed by validState

Parameters

<i>state</i>	New state
--------------	-----------

Definition at line 41 of file [cell.cpp](#).

References [m_nextState](#), and [m_state](#).

6.1.3.3 getNeighbour()

```
const Cell * Cell::getNeighbour (
    QVector< short > relativePosition ) const
```

Get the neighbour asked. If not existent, return nullptr.

Definition at line 80 of file [cell.cpp](#).

References [m_neighbours](#).

6.1.3.4 getNeighbours()

```
QMap< QVector< short >, const Cell * > Cell::getNeighbours ( ) const
```

Access neighbours list.

The map key is the relative position of the neighbour (like -1,0 for the cell just above)

Definition at line 73 of file [cell.cpp](#).

References [m_neighbours](#).

6.1.3.5 getRelativePosition()

```
QVector< short > Cell::getRelativePosition (
    const QVector< unsigned int > cellPosition,
    const QVector< unsigned int > neighbourPosition ) [static]
```

Get the relative position, as *neighbourPosition* minus *cellPosition*.

Exceptions

<i>QString</i>	Different size of position vectors
----------------	------------------------------------

Parameters

<i>cellPosition</i>	Cell Position
<i>neighbourPosition</i>	Neighbour absolute position

Definition at line 91 of file [cell.cpp](#).

Referenced by [CellHandler::foundNeighbours\(\)](#).

6.1.3.6 getState()

```
unsigned int Cell::getState ( ) const
```

Access current cell state.

Definition at line 48 of file [cell.cpp](#).

References [m_state](#).

6.1.3.7 setState()

```
void Cell::setState (
    unsigned int state )
```

Set temporary state.

To change current cell state, use [setState\(unsigned int state\)](#) then [validState\(\)](#).

Parameters

<i>state</i>	New state
--------------	-----------

Definition at line 20 of file [cell.cpp](#).

References [m_nextState](#).

6.1.3.8 validState()

```
void Cell::validState ( )
```

Validate temporary state.

To change current cell state, use [setState\(unsigned int state\)](#) then [validState\(\)](#).

Definition at line 30 of file [cell.cpp](#).

References [m_nextState](#), and [m_state](#).

6.1.4 Member Data Documentation

6.1.4.1 m_neighbours

```
QMap<QVector<short>, const Cell*> Cell::m_neighbours [private]
```

[Cell](#)'s neighbours. Key is the relative position of the neighbour.

Definition at line 30 of file [cell.h](#).

Referenced by [addNeighbour\(\)](#), [getNeighbour\(\)](#), and [getNeighbours\(\)](#).

6.1.4.2 m_nextState

```
unsigned int Cell::m_nextState [private]
```

Temporary state, before validation.

Definition at line 28 of file [cell.h](#).

Referenced by [forceState\(\)](#), [setState\(\)](#), and [validState\(\)](#).

6.1.4.3 m_state

```
unsigned int Cell::m_state [private]
```

Current state.

Definition at line 27 of file [cell.h](#).

Referenced by [forceState\(\)](#), [getState\(\)](#), and [validState\(\)](#).

The documentation for this class was generated from the following files:

- [cell.h](#)
- [cell.cpp](#)

6.2 CellHandler Class Reference

[Cell](#) container and cell generator.

```
#include <cellhandler.h>
```

Classes

- class [iteratorT](#)

Implementation of iterator design pattern with a template to generate iterator and const_iterator at the same time.

Public Types

- enum [generationTypes](#) { [empty](#), [random](#), [symetric](#) }
- *Type of random generation.*
- typedef [iteratorT](#)< const [CellHandler](#), const [Cell](#) > [const_iterator](#)
- typedef [iteratorT](#)< [CellHandler](#), [Cell](#) > [iterator](#)

Public Member Functions

- [CellHandler](#) (const QString filename)
Construct all the cells from the json file given.
- [CellHandler](#) (const QVector< unsigned int > dimensions, [generationTypes](#) type=[empty](#), unsigned int state↔
Max=1, unsigned int density=20)
Construct a [CellHandler](#) of the given dimension.
- virtual [~CellHandler](#) ()
Destroys all cells in the [CellHandler](#).
- [Cell](#) * [getCell](#) (const QVector< unsigned int > position) const
Access the cell to the given position.
- QVector< unsigned int > [getDimensions](#) () const
Accessor of m_dimensions.
- void [nextStates](#) () const
Valid the state of all cells.
- bool [save](#) (QString filename) const
Save the [CellHandler](#) current configuration in the file given.
- void [generate](#) ([generationTypes](#) type, unsigned int stateMax=1, unsigned short density=50)
Replace [Cell](#) values by random values (symetric or not)
- void [print](#) (std::ostream &stream) const
Print in the given stream the [CellHandler](#).
- [const_iterator](#) [begin](#) () const
Give the iterator which corresponds to the current [CellHandler](#).
- [iterator](#) [begin](#) ()
Give the iterator which corresponds to the current [CellHandler](#).
- bool [end](#) () const
End condition of the iterator.

Private Member Functions

- bool [load](#) (const QJsonObject &json)
Load the config file in the [CellHandler](#).
- void [foundNeighbours](#) ()
Set the neighbours of each cells.
- void [positionIncrement](#) (QVector< unsigned int > &pos, unsigned int value=1) const
Increment the QVector given by the value choosen.
- QVector< QVector< unsigned int > > * [getListNeighboursPositionsRecursive](#) (const QVector< unsigned int > position, unsigned int dimension, QVector< unsigned int > lastAdd) const
Recursive function which browse the position possibilities tree.
- QVector< QVector< unsigned int > > & [getListNeighboursPositions](#) (const QVector< unsigned int > position) const
Prepare the call of the recursive version of itself.

Private Attributes

- QVector< unsigned int > [m_dimensions](#)
Vector of x dimensions.
- QMap< QVector< unsigned int >, [Cell](#) *> [m_cells](#)
Map of cells, with a x dimensions vector as key.

6.2.1 Detailed Description

[Cell](#) container and cell generator.

Generate cells from a json file.

Definition at line 20 of file [cellhandler.h](#).

6.2.2 Member Typedef Documentation

6.2.2.1 const_iterator

```
typedef iteratorT<const CellHandler, const Cell> CellHandler::const\_iterator
```

Definition at line 64 of file [cellhandler.h](#).

6.2.2.2 iterator

```
typedef iteratorT<CellHandler, Cell> CellHandler::iterator
```

Definition at line 65 of file [cellhandler.h](#).

6.2.3 Member Enumeration Documentation

6.2.3.1 generationTypes

```
enum CellHandler::generationTypes
```

Type of random generation.

Enumerator

empty	Only empty cells.
random	Random cells.
symetric	Random cells but with vertical symetry (on the 1st dimension component)

Definition at line 69 of file [cellhandler.h](#).

6.2.4 Constructor & Destructor Documentation**6.2.4.1 CellHandler()** [1/2]

```
CellHandler::CellHandler (
    const QString filename )
```

Construct all the cells from the json file given.

The size of "cells" array must be the product of all dimensions (60 in the following example). Typical Json file:

```
{
  "dimensions": "3x4x5",
  "cells": [0,1,4,4,2,5,3,4,2,4,
            4,2,5,0,0,0,0,0,0,0,
            2,4,1,1,1,1,1,2,1,1,
            0,0,0,0,0,0,2,2,2,2,
            3,4,5,1,2,0,9,0,0,0,
            1,2,0,0,0,0,1,2,3,2]
}
```

Parameters

<i>filename</i>	Json file which contains the description of all the cells
-----------------	---

Exceptions

<i>QString</i>	Unreadable file
<i>QString</i>	Empty file
<i>QString</i>	Not valid file

Definition at line 25 of file [cellhandler.cpp](#).

References [foundNeighbours\(\)](#), and [load\(\)](#).

6.2.4.2 CellHandler() [2/2]

```
CellHandler::CellHandler (
    const QVector< unsigned int > dimensions,
```

```

generationTypes type = empty,
unsigned int stateMax = 1,
unsigned int density = 20 )

```

Construct a [CellHandler](#) of the given dimension.

If generationTypes is given, the [CellHandler](#) won't be empty.

Parameters

<i>dimensions</i>	Dimensions of the CellHandler
<i>type</i>	Generation type, empty by default
<i>stateMax</i>	Generate states between 0 and stateMax
<i>density</i>	Average (%) of non-zeros

Definition at line 65 of file [cellhandler.cpp](#).

References [empty](#), [foundNeighbours\(\)](#), [generate\(\)](#), [m_cells](#), [m_dimensions](#), and [positionIncrement\(\)](#).

6.2.4.3 ~CellHandler()

```
CellHandler::~~CellHandler ( ) [virtual]
```

Destroys all cells in the [CellHandler](#).

Definition at line 97 of file [cellhandler.cpp](#).

References [m_cells](#).

6.2.5 Member Function Documentation

6.2.5.1 begin() [1/2]

```
CellHandler::const_iterator CellHandler::begin ( ) const
```

Give the iterator which corresponds to the current [CellHandler](#).

Definition at line 262 of file [cellhandler.cpp](#).

Referenced by [print\(\)](#), and [save\(\)](#).

6.2.5.2 begin() [2/2]

```
CellHandler::iterator CellHandler::begin ( )
```

Give the iterator which corresponds to the current [CellHandler](#).

Definition at line [255](#) of file [cellhandler.cpp](#).

6.2.5.3 end()

```
bool CellHandler::end ( ) const
```

End condition of the iterator.

See [iterator::operator!=\(bool finished\)](#) for further information.

Definition at line [271](#) of file [cellhandler.cpp](#).

Referenced by [print\(\)](#), [save\(\)](#), and [MainWindow::updateBoard\(\)](#).

6.2.5.4 foundNeighbours()

```
void CellHandler::foundNeighbours ( ) [private]
```

Set the neighbours of each cells.

Careful, this is in $O(n \cdot 3^d)$, with n the number of cells and d the number of dimensions

Definition at line [364](#) of file [cellhandler.cpp](#).

References [getListNeighboursPositions\(\)](#), [Cell::getRelativePosition\(\)](#), [m_cells](#), [m_dimensions](#), and [positionIncrement\(\)](#).

Referenced by [CellHandler\(\)](#).

6.2.5.5 generate()

```
void CellHandler::generate (
    CellHandler::generationTypes type,
    unsigned int stateMax = 1,
    unsigned short density = 50 )
```

Replace [Cell](#) values by random values (symetric or not)

Parameters

<i>type</i>	Type of random generation
<i>stateMax</i>	Generate states between 0 and stateMax
<i>density</i>	Average (%) of non-zeros

Definition at line 176 of file [cellhandler.cpp](#).

References [m_cells](#), [m_dimensions](#), [positionIncrement\(\)](#), [random](#), and [symetric](#).

Referenced by [CellHandler\(\)](#).

6.2.5.6 [getCell\(\)](#)

```
Cell * CellHandler::getCell (
    const QVector< unsigned int > position ) const
```

Access the cell to the given position.

Definition at line 107 of file [cellhandler.cpp](#).

References [m_cells](#).

6.2.5.7 [getDimensions\(\)](#)

```
QVector< unsigned int > CellHandler::getDimensions ( ) const
```

Accessor of [m_dimensions](#).

Definition at line 114 of file [cellhandler.cpp](#).

References [m_dimensions](#).

Referenced by [MainWindow::updateBoard\(\)](#).

6.2.5.8 [getListNeighboursPositions\(\)](#)

```
QVector< QVector< unsigned int > > & CellHandler::getListNeighboursPositions (
    const QVector< unsigned int > position ) const [private]
```

Prepare the call of the recursive version of itself.

Parameters

<i>position</i>	Position of the central cell (x1,x2,x3,...,xn)
-----------------	--

Returns

List of positions

Definition at line 423 of file [cellhandler.cpp](#).

References [getListNeighboursPositionsRecursive\(\)](#).

Referenced by [foundNeighbours\(\)](#).

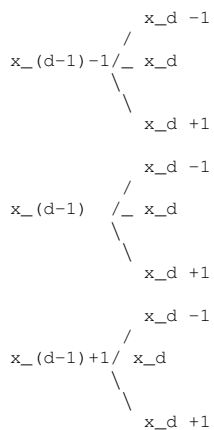
6.2.5.9 getListNeighboursPositionsRecursive()

```
QVector< QVector< unsigned int > > * CellHandler::getListNeighboursPositionsRecursive (
    const QVector< unsigned int > position,
    unsigned int dimension,
    QVector< unsigned int > lastAdd ) const [private]
```

Recursive function which browse the position possibilities tree.

Careful, the complexity is in $O(3^{\text{dimension}})$

Piece of the tree:



The path in the tree to reach the leaf give the position

Parameters

<i>position</i>	Position of the cell
<i>dimension</i>	Current working dimension (number of the digit). Dimension = 2 \Leftrightarrow working on x2 coordinates on (x1, x2, x3, ..., xn) vector
<i>lastAdd</i>	Last position added. Like the father node of the new tree

Returns

List of position

Definition at line 464 of file [cellhandler.cpp](#).

References [m_dimensions](#).

Referenced by [getListNeighboursPositions\(\)](#).

6.2.5.10 load()

```
bool CellHandler::load (
    const QJsonObject & json ) [private]
```

Load the config file in the [CellHandler](#).

Exemple of a way to print cell states :

```
QVector<unsigned int> position;
for (unsigned short i = 0; i < m_dimensions.size(); i++)
{
    position.push_back(0);
}
for (unsigned int j = 0; j < m_cells.size(); j++)
{
    std::cout << m_cells.value(position)->getState() << " ";
    position.replace(0, position.at(0)+1);
    for (unsigned short i = 0; i < m_dimensions.size(); i++)
    {
        if (position.at(i) >= m_dimensions.at(i))
        {
            position.replace(i, 0);
            std::cout << std::endl;
            if (i + 1 != m_dimensions.size())
                position.replace(i+1, position.at(i+1)+1);
        }
    }
}
```

Parameters

<i>json</i>	Json Object which contains the grid configuration
-------------	---

Returns

False if the Json Object is not correct

Definition at line 306 of file [cellhandler.cpp](#).

References [m_cells](#), [m_dimensions](#), and [positionIncrement\(\)](#).

Referenced by [CellHandler\(\)](#).

6.2.5.11 nextStates()

```
void CellHandler::nextStates ( ) const
```

Valid the state of all cells.

Definition at line 121 of file [cellhandler.cpp](#).

References [m_cells](#).

6.2.5.12 positionIncrement()

```
void CellHandler::positionIncrement (
    QVector< unsigned int > & pos,
    unsigned int value = 1 ) const [private]
```

Increment the QVector given by the value choosen.

Careful, when the position reach the maximum, it goes to zero without leaving the function

Parameters

<i>pos</i>	Position to increment
<i>value</i>	Value to add, 1 by default

Definition at line 394 of file [cellhandler.cpp](#).

References [m_dimensions](#).

Referenced by [CellHandler\(\)](#), [foundNeighbours\(\)](#), [generate\(\)](#), and [load\(\)](#).

6.2.5.13 print()

```
void CellHandler::print (
    std::ostream & stream ) const
```

Print in the given stream the [CellHandler](#).

Parameters

<i>stream</i>	Stream to print into
---------------	----------------------

Definition at line 241 of file [cellhandler.cpp](#).

References [begin\(\)](#), and [end\(\)](#).

6.2.5.14 save()

```
bool CellHandler::save (
    QString filename ) const
```

Save the [CellHandler](#) current configuration in the file given.

Parameters

<i>filename</i>	Path to the file
-----------------	------------------

Returns

False if there was a problem

Exceptions

<i>QString</i>	Impossible to open the file
----------------	-----------------------------

Definition at line 136 of file [cellhandler.cpp](#).

References [begin\(\)](#), [end\(\)](#), and [m_dimensions](#).

6.2.6 Member Data Documentation

6.2.6.1 m_cells

```
QMap<QVector<unsigned int>, Cell* > CellHandler::m_cells [private]
```

Map of cells, with a x dimensions vector as key.

Definition at line 100 of file [cellhandler.h](#).

Referenced by [CellHandler\(\)](#), [foundNeighbours\(\)](#), [generate\(\)](#), [getCell\(\)](#), [load\(\)](#), [nextStates\(\)](#), and [~CellHandler\(\)](#).

6.2.6.2 m_dimensions

```
QVector<unsigned int> CellHandler::m_dimensions [private]
```

Vector of x dimensions.

Definition at line 99 of file [cellhandler.h](#).

Referenced by [CellHandler\(\)](#), [foundNeighbours\(\)](#), [generate\(\)](#), [getDimensions\(\)](#), [getListNeighboursPositionsRecursive\(\)](#), [load\(\)](#), [positionIncrement\(\)](#), and [save\(\)](#).

The documentation for this class was generated from the following files:

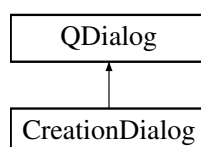
- [cellhandler.h](#)
- [cellhandler.cpp](#)

6.3 CreationDialog Class Reference

Automaton creation dialog box.

```
#include <creationdialog.h>
```

Inheritance diagram for CreationDialog:



Public Slots

- void [processSettings](#) ()

Signals

- void [settingsFilled](#) (const QVector< unsigned int > dimensions, [CellHandler::generationTypes](#) type=Cell↔Handler::generationTypes::empty, unsigned int stateMax=1, unsigned int density=20)

Public Member Functions

- [CreationDialog](#) (QWidget *parent=0)

Private Member Functions

- QGroupBox * [createGenButtons](#) ()
Creates radio buttons to select cell generation type.

Private Attributes

- QLineEdit * [m_dimensionsEdit](#)
- QSpinBox * [m_densityBox](#)
- QSpinBox * [m_stateMaxBox](#)
- QPushButton * [m_doneBt](#)
- QGroupBox * [m_groupBox](#)
- QRadioButton * [m_empGen](#)
- QRadioButton * [m_randGen](#)
- QRadioButton * [m_symGen](#)

6.3.1 Detailed Description

Automaton creation dialog box.

Allow the user to input settings to create an automaton

Definition at line 13 of file [creationdialog.h](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 CreationDialog()

```
CreationDialog::CreationDialog (  
    QWidget * parent = 0 )
```

Definition at line 5 of file [creationdialog.cpp](#).

References [createGenButtons\(\)](#), [m_densityBox](#), [m_dimensionsEdit](#), [m_doneBt](#), [m_stateMaxBox](#), and [processSettings\(\)](#).

6.3.3 Member Function Documentation

6.3.3.1 createGenButtons()

```
CreationDialog::createGenButtons ( ) [private]
```

Creates radio buttons to select cell generation type.

Validates user settings and sends them to [MainWindow](#).

Definition at line 51 of file [creationdialog.cpp](#).

References [m_empGen](#), [m_groupBox](#), [m_randGen](#), and [m_symGen](#).

Referenced by [CreationDialog\(\)](#).

6.3.3.2 processSettings

```
void CreationDialog::processSettings ( ) [slot]
```

Definition at line 72 of file [creationdialog.cpp](#).

References [m_densityBox](#), [m_dimensionsEdit](#), [m_randGen](#), [m_stateMaxBox](#), [m_symGen](#), and [settingsFilled\(\)](#).

Referenced by [CreationDialog\(\)](#).

6.3.3.3 settingsFilled

```
void CreationDialog::settingsFilled (
    const QVector< unsigned int > dimensions,
    CellHandler::generationTypes type = CellHandler::generationTypes::empty,
    unsigned int stateMax = 1,
    unsigned int density = 20 ) [signal]
```

Referenced by [processSettings\(\)](#).

6.3.4 Member Data Documentation

6.3.4.1 m_densityBox

```
QSpinBox* CreationDialog::m_densityBox [private]
```

Definition at line 30 of file [creationdialog.h](#).

Referenced by [CreationDialog\(\)](#), and [processSettings\(\)](#).

6.3.4.2 m_dimensionsEdit

```
QLineEdit* CreationDialog::m_dimensionsEdit [private]
```

Definition at line 29 of file [creationdialog.h](#).

Referenced by [CreationDialog\(\)](#), and [processSettings\(\)](#).

6.3.4.3 m_doneBt

```
QPushButton* CreationDialog::m_doneBt [private]
```

Definition at line 32 of file [creationdialog.h](#).

Referenced by [CreationDialog\(\)](#).

6.3.4.4 m_empGen

```
QRadioButton* CreationDialog::m_empGen [private]
```

Definition at line 35 of file [creationdialog.h](#).

Referenced by [createGenButtons\(\)](#).

6.3.4.5 m_groupBox

```
QGroupBox* CreationDialog::m_groupBox [private]
```

Definition at line 34 of file [creationdialog.h](#).

Referenced by [createGenButtons\(\)](#).

6.3.4.6 m_randGen

`QRadioButton* CreationDialog::m_randGen [private]`

Definition at line 36 of file [creationdialog.h](#).

Referenced by [createGenButtons\(\)](#), and [processSettings\(\)](#).

6.3.4.7 m_stateMaxBox

`QSpinBox* CreationDialog::m_stateMaxBox [private]`

Definition at line 31 of file [creationdialog.h](#).

Referenced by [CreationDialog\(\)](#), and [processSettings\(\)](#).

6.3.4.8 m_symGen

`QRadioButton* CreationDialog::m_symGen [private]`

Definition at line 37 of file [creationdialog.h](#).

Referenced by [createGenButtons\(\)](#), and [processSettings\(\)](#).

The documentation for this class was generated from the following files:

- [creationdialog.h](#)
- [creationdialog.cpp](#)

6.4 CellHandler::iteratorT< CellHandler_T, Cell_T > Class Template Reference

Implementation of iterator design pattern with a template to generate iterator and const_iterator at the same time.

Public Member Functions

- [iteratorT](#) (CellHandler_T *handler)
Construct an initial iterator to browse the [CellHandler](#).
- [iteratorT](#) & [operator++](#) ()
Increment the current position and handle dimension changes.
- Cell_T * [operator->](#) () const
Get the current cell.
- Cell_T * [operator*](#) () const
Get the current cell.
- bool [operator!=](#) (bool finished) const
- unsigned int [changedDimension](#) () const
Return the number of dimensions we change.

Private Attributes

- CellHandler_T * [m_handler](#)
CellHandler to go through.
- QVector< unsigned int > [m_position](#)
Current position of the iterator.
- bool [m_finished](#) = false
If we reach the last position.
- QVector< unsigned int > [m_zero](#)
Nul vector of the good dimension (depend of m_handler)
- unsigned int [m_changedDimension](#)
Save the number of dimension change.

Friends

- class [CellHandler](#)

6.4.1 Detailed Description

```
template<typename CellHandler_T, typename Cell_T>
class CellHandler::iteratorT< CellHandler_T, Cell_T >
```

Implementation of iterator design pattern with a template to generate iterator and const_iterator at the same time.

Example of use:

```
CellHandler handler("file.atc");
for (CellHandler::const_iterator it = handler.begin(); it != handler.end(); ++it)
{
    for (unsigned int i = 0; i < it.changedDimension(); i++)
        std::cout << std::endl;
    std::cout << it->getState() << " ";
}
```

This code will print each cell states and go to a new line when there is a change of dimension. So if there are 3 dimensions, there will be a empty line between 2D groups.

Definition at line 41 of file [cellhandler.h](#).

6.4.2 Constructor & Destructor Documentation

6.4.2.1 iteratorT()

```
template<typename CellHandler_T , typename Cell_T >
CellHandler::iteratorT< CellHandler_T, Cell_T >::iteratorT (
    CellHandler_T * handler )
```

Construct an initial iterator to browse the [CellHandler](#).

Parameters

<i>handler</i>	CellHandler to browse
----------------	---------------------------------------

Definition at line 504 of file [cellhandler.cpp](#).

References [CellHandler::iteratorT< CellHandler_T, Cell_T >::m_position](#), and [CellHandler::iteratorT< CellHandler_T, Cell_T >::m_z](#).

6.4.3 Member Function Documentation

6.4.3.1 `changedDimension()`

```
template<typename CellHandler_T , typename Cell_T >
unsigned int CellHandler::iteratorT< CellHandler\_T, Cell\_T >::changedDimension ( ) const
```

Return the number of dimensions we change.

For example, if we were at the (3,4,4) cell, and we incremented the position, we are now at (4,0,0), and `changedDimension` return 2 (because of the 2 zeros).

Definition at line 566 of file [cellhandler.cpp](#).

6.4.3.2 `operator!=()`

```
template<typename CellHandler_T , typename Cell_T >
bool CellHandler::iteratorT< CellHandler\_T, Cell\_T >::operator!= (
    bool finished ) const [inline]
```

Definition at line 51 of file [cellhandler.h](#).

References [CellHandler::iteratorT< CellHandler_T, Cell_T >::m_finished](#).

6.4.3.3 `operator*()`

```
template<typename CellHandler_T , typename Cell_T >
Cell_T * CellHandler::iteratorT< CellHandler\_T, Cell\_T >::operator\* ( ) const
```

Get the current cell.

Definition at line 555 of file [cellhandler.cpp](#).

6.4.3.4 operator++()

```
template<typename CellHandler_T , typename Cell_T >  
CellHandler::iteratorT< CellHandler_T, Cell_T > & CellHandler::iteratorT< CellHandler_T,  
Cell_T >::operator++ ( )
```

Increment the current position and handle dimension changes.

Definition at line 518 of file [cellhandler.cpp](#).

6.4.3.5 operator->()

```
template<typename CellHandler_T , typename Cell_T >  
Cell_T * CellHandler::iteratorT< CellHandler_T, Cell_T >::operator-> ( ) const
```

Get the current cell.

Definition at line 546 of file [cellhandler.cpp](#).

6.4.4 Friends And Related Function Documentation

6.4.4.1 CellHandler

```
template<typename CellHandler_T , typename Cell_T >  
friend class CellHandler [friend]
```

Definition at line 43 of file [cellhandler.h](#).

6.4.5 Member Data Documentation

6.4.5.1 m_changedDimension

```
template<typename CellHandler_T , typename Cell_T >  
unsigned int CellHandler::iteratorT< CellHandler_T, Cell_T >::m_changedDimension [private]
```

Save the number of dimension change.

Definition at line 61 of file [cellhandler.h](#).

6.4.5.2 m_finished

```
template<typename CellHandler_T , typename Cell_T >
bool CellHandler::iteratorT< CellHandler_T, Cell_T >::m_finished = false [private]
```

If we reach the last position.

Definition at line 59 of file [cellhandler.h](#).

Referenced by [CellHandler::iteratorT< CellHandler_T, Cell_T >::operator!==\(\(\)\)](#).

6.4.5.3 m_handler

```
template<typename CellHandler_T , typename Cell_T >
CellHandler_T* CellHandler::iteratorT< CellHandler_T, Cell_T >::m_handler [private]
```

[CellHandler](#) to go through.

Definition at line 57 of file [cellhandler.h](#).

6.4.5.4 m_position

```
template<typename CellHandler_T , typename Cell_T >
QVector<unsigned int> CellHandler::iteratorT< CellHandler_T, Cell_T >::m_position [private]
```

Current position of the iterator.

Definition at line 58 of file [cellhandler.h](#).

Referenced by [CellHandler::iteratorT< CellHandler_T, Cell_T >::iteratorT\(\)](#).

6.4.5.5 m_zero

```
template<typename CellHandler_T , typename Cell_T >
QVector<unsigned int> CellHandler::iteratorT< CellHandler_T, Cell_T >::m_zero [private]
```

Nul vector of the good dimension (depend of m_handler)

Definition at line 60 of file [cellhandler.h](#).

Referenced by [CellHandler::iteratorT< CellHandler_T, Cell_T >::iteratorT\(\)](#).

The documentation for this class was generated from the following files:

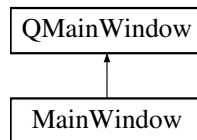
- [cellhandler.h](#)
- [cellhandler.cpp](#)

6.5 MainWindow Class Reference

Simulation window.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Slots

- void [openFile](#) ()
Opens a file browser for the user to select automaton files and creates an automaton.
- void [saveToFile](#) ()
Allows user to select a location and saves automaton's state and settings.
- void [openCreationWindow](#) ()
Opens the automaton creation window.
- void [setCellHandler](#) (const QVector< unsigned int > dimensions, [CellHandler::generationTypes](#) type=CellHandler::generationTypes::empty, unsigned int stateMax=1, unsigned int density=20)
Creates a new cellHandler with the provided arguments and updates the board with the created cellHandler.
- void [forward](#) ()
Skips the number of steps chosen by the user and sets the automaton on the last one.
- void [closeTab](#) (int n)
Closes the tab at index n. Before closing, prompts the user to save the automaton.

Public Member Functions

- [MainWindow](#) (QWidget *parent=nullptr)

Private Member Functions

- void [createIcons](#) ()
Creates Icons for the [MainWindow](#).
- void [createActions](#) ()
Creates and connects QActions and associated buttons for the [MainWindow](#).
- void [createToolBar](#) ()
Creates the toolBar for the [MainWindow](#).
- void [createBoard](#) ()
- QWidget * [createTab](#) ()
Creates a new Tab with an empty board.
- void [createTabs](#) ()
Creates a QTabWidget for the main window and displays it.
- void [updateBoard](#) (int index)
Updates cells on the board on the tab at the give index with the cellHandler's cells states.
- void [nextState](#) (int n)
Shows the nth next state of the automaton on the board.
- QTableWidgetItem * [getBoard](#) (int n)

Private Attributes

- QTabWidget * [m_tabs](#)
- QVector< [CellHandler](#) * > [m_cellHandlers](#)
- QIcon [m_fastBackwardIcon](#)

Icons.

- QIcon [m_fastForwardIcon](#)
- QIcon [m_playIcon](#)
- QIcon [m_pauseIcon](#)
- QIcon [m_newIcon](#)
- QIcon [m_saveIcon](#)
- QIcon [m_openIcon](#)
- QIcon [m_resetIcon](#)
- QAction * [m_playPause](#)

Actions.

- QAction * [m_nextState](#)
- QAction * [m_previousState](#)
- QAction * [m_fastForward](#)
- QAction * [m_fastBackward](#)
- QAction * [m_openAutomaton](#)
- QAction * [m_saveAutomaton](#)
- QAction * [m_newAutomaton](#)
- QAction * [m_resetAutomaton](#)
- QToolButton * [m_playPauseBt](#)

Buttons.

- QToolButton * [m_nextStateBt](#)
- QToolButton * [m_previousStateBt](#)
- QToolButton * [m_fastForwardBt](#)
- QToolButton * [m_fastBackwardBt](#)
- QToolButton * [m_openAutomatonBt](#)
- QToolButton * [m_saveAutomatonBt](#)
- QToolButton * [m_newAutomatonBt](#)
- QToolButton * [m_resetBt](#)
- QSpinBox * [m_jumpSpeed](#)
- QLabel * [m_speedLabel](#)

Simulation speed input.

- QToolBar * [m_toolBar](#)
- unsigned int [m_boardHSize](#) = 25

Toolbar containing the buttons.

- unsigned int [m_boardVSize](#) = 25
- unsigned int [m_cellSize](#) = 30

6.5.1 Detailed Description

Simulation window.

Displays the automaton's current state as a board and contains user interaction components.

Definition at line 16 of file [mainwindow.h](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr ) [explicit]
```

Definition at line 3 of file [mainwindow.cpp](#).

References [createActions\(\)](#), [createIcons\(\)](#), [createToolBar\(\)](#), and [m_tabs](#).

6.5.3 Member Function Documentation

6.5.3.1 closeTab

```
void MainWindow::closeTab (
    int n ) [slot]
```

Closes the tab at index n. Before closing, prompts the user to save the automaton.

Definition at line 324 of file [mainwindow.cpp](#).

References [m_tabs](#), and [saveToFile\(\)](#).

Referenced by [createTabs\(\)](#).

6.5.3.2 createActions()

```
void MainWindow::createActions ( ) [private]
```

Creates and connects QActions and associated buttons for the [MainWindow](#).

Definition at line 51 of file [mainwindow.cpp](#).

References [forward\(\)](#), [m_fastBackward](#), [m_fastBackwardBt](#), [m_fastBackwardIcon](#), [m_fastForward](#), [m_fastForwardBt](#), [m_fastForwardIcon](#), [m_newAutomaton](#), [m_newAutomatonBt](#), [m_newIcon](#), [m_openAutomaton](#), [m_openAutomatonBt](#), [m_openIcon](#), [m_playIcon](#), [m_playPause](#), [m_playPauseBt](#), [m_resetAutomaton](#), [m_resetBt](#), [m_resetIcon](#), [m_saveAutomaton](#), [m_saveAutomatonBt](#), [m_saveIcon](#), [openCreationWindow\(\)](#), [openFile\(\)](#), and [saveToFile\(\)](#).

Referenced by [MainWindow\(\)](#).

6.5.3.3 createBoard()

```
void MainWindow::createBoard ( ) [private]
```

6.5.3.4 createIcons()

```
void MainWindow::createIcons ( ) [private]
```

Creates Icons for the [MainWindow](#).

Definition at line 21 of file [mainwindow.cpp](#).

References [m_fastBackwardIcon](#), [m_fastForwardIcon](#), [m_newIcon](#), [m_openIcon](#), [m_pauseIcon](#), [m_playIcon](#), [m_resetIcon](#), and [m_saveIcon](#).

Referenced by [MainWindow\(\)](#).

6.5.3.5 createTab()

```
QWidget * MainWindow::createTab ( ) [private]
```

Creates a new Tab with an empty board.

Definition at line 131 of file [mainwindow.cpp](#).

References [m_cellHandlers](#), and [m_cellSize](#).

Referenced by [openFile\(\)](#), and [setCellHandler\(\)](#).

6.5.3.6 createTabs()

```
void MainWindow::createTabs ( ) [private]
```

Creates a QTabWidget for the main window and displays it.

Definition at line 312 of file [mainwindow.cpp](#).

References [closeTab\(\)](#), and [m_tabs](#).

Referenced by [openFile\(\)](#), and [setCellHandler\(\)](#).

6.5.3.7 createToolBar()

```
void MainWindow::createToolBar ( ) [private]
```

Creates the toolBar for the [MainWindow](#).

Definition at line 97 of file [mainwindow.cpp](#).

References [m_fastBackwardBt](#), [m_fastForwardBt](#), [m_jumpSpeed](#), [m_newAutomatonBt](#), [m_openAutomatonBt](#), [m_playPauseBt](#), [m_resetBt](#), [m_saveAutomatonBt](#), [m_speedLabel](#), and [m_toolBar](#).

Referenced by [MainWindow\(\)](#).

6.5.3.8 forward

```
void MainWindow::forward ( ) [slot]
```

Skips the number of steps chosen by the user and sets the automaton on the last one.

Definition at line 300 of file [mainwindow.cpp](#).

References [m_jumpSpeed](#), and [nextState\(\)](#).

Referenced by [createActions\(\)](#).

6.5.3.9 getBoard()

```
QTableWidget * MainWindow::getBoard (
    int n ) [private]
```

Definition at line 304 of file [mainwindow.cpp](#).

References [m_tabs](#).

Referenced by [updateBoard\(\)](#).

6.5.3.10 nextState()

```
void MainWindow::nextState (
    int n ) [private]
```

Shows the nth next state of the automaton on the board.

Definition at line 243 of file [mainwindow.cpp](#).

References [m_cellHandlers](#), [m_tabs](#), and [updateBoard\(\)](#).

Referenced by [forward\(\)](#).

6.5.3.11 openCreationWindow

```
void MainWindow::openCreationWindow ( ) [slot]
```

Opens the automaton creation window.

Definition at line 210 of file [mainwindow.cpp](#).

References [setCellHandler\(\)](#).

Referenced by [createActions\(\)](#).

6.5.3.12 openFile

```
void MainWindow::openFile ( ) [slot]
```

Opens a file browser for the user to select automaton files and creates an automaton.

Definition at line 176 of file [mainwindow.cpp](#).

References [createTab\(\)](#), [createTabs\(\)](#), [m_cellHandlers](#), [m_tabs](#), and [updateBoard\(\)](#).

Referenced by [createActions\(\)](#).

6.5.3.13 saveToFile

```
void MainWindow::saveToFile ( ) [slot]
```

Allows user to select a location and saves automaton's state and settings.

Definition at line 192 of file [mainwindow.cpp](#).

References [m_cellHandlers](#), and [m_tabs](#).

Referenced by [closeTab\(\)](#), and [createActions\(\)](#).

6.5.3.14 setCellHandler

```
void MainWindow::setCellHandler (
    const QVector< unsigned int > dimensions,
    CellHandler::generationTypes type = CellHandler::generationTypes::empty,
    unsigned int stateMax = 1,
    unsigned int density = 20 ) [slot]
```

Creates a new cellHandler with the provided arguments and updates the board with the created cellHandler.

Definition at line 223 of file [mainwindow.cpp](#).

References [createTab\(\)](#), [createTabs\(\)](#), [m_cellHandlers](#), [m_tabs](#), and [updateBoard\(\)](#).

Referenced by [openCreationWindow\(\)](#).

6.5.3.15 updateBoard()

```
void MainWindow::updateBoard (
    int index ) [private]
```

Updates cells on the board on the tab at the give index with the cellHandler's cells states.

Definition at line 259 of file [mainwindow.cpp](#).

References [CellHandler::end\(\)](#), [getBoard\(\)](#), [CellHandler::getDimensions\(\)](#), and [m_cellHandlers](#).

Referenced by [nextState\(\)](#), [openFile\(\)](#), and [setCellHandler\(\)](#).

6.5.4 Member Data Documentation

6.5.4.1 m_boardHSize

```
unsigned int MainWindow::m_boardHSize = 25 [private]
```

Toolbar containing the buttons.

Board size settings

Definition at line 62 of file [mainwindow.h](#).

6.5.4.2 m_boardVSize

```
unsigned int MainWindow::m_boardVSize = 25 [private]
```

Definition at line 63 of file [mainwindow.h](#).

6.5.4.3 m_cellHandlers

```
QVector<CellHandler *> MainWindow::m_cellHandlers [private]
```

Definition at line 21 of file [mainwindow.h](#).

Referenced by [createTab\(\)](#), [nextState\(\)](#), [openFile\(\)](#), [saveToFile\(\)](#), [setCellHandler\(\)](#), and [updateBoard\(\)](#).

6.5.4.4 m_cellSize

```
unsigned int MainWindow::m_cellSize = 30 [private]
```

Definition at line 64 of file [mainwindow.h](#).

Referenced by [createTab\(\)](#).

6.5.4.5 m_fastBackward

```
QAction* MainWindow::m_fastBackward [private]
```

Definition at line 38 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.6 m_fastBackwardBt

```
QPushButton* MainWindow::m_fastBackwardBt [private]
```

Definition at line 49 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.7 m_fastBackwardIcon

```
QIcon MainWindow::m_fastBackwardIcon [private]
```

Icons.

Definition at line 24 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.8 m_fastForward

```
QAction* MainWindow::m_fastForward [private]
```

Definition at line 37 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.9 m_fastForwardBt

`QToolButton* MainWindow::m_fastForwardBt [private]`

Definition at line 48 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.10 m_fastForwardIcon

`QIcon MainWindow::m_fastForwardIcon [private]`

Definition at line 25 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.11 m_jumpSpeed

`QSpinBox* MainWindow::m_jumpSpeed [private]`

Definition at line 56 of file [mainwindow.h](#).

Referenced by [createToolBar\(\)](#), and [forward\(\)](#).

6.5.4.12 m_newAutomaton

`QAction* MainWindow::m_newAutomaton [private]`

Definition at line 41 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.13 m_newAutomatonBt

`QToolButton* MainWindow::m_newAutomatonBt [private]`

Definition at line 52 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.14 `m_newIcon`

`QIcon MainWindow::m_newIcon [private]`

Definition at line 28 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.15 `m_nextState`

`QAction* MainWindow::m_nextState [private]`

Definition at line 35 of file [mainwindow.h](#).

6.5.4.16 `m_nextStateBt`

`QToolButton* MainWindow::m_nextStateBt [private]`

Definition at line 46 of file [mainwindow.h](#).

6.5.4.17 `m_openAutomaton`

`QAction* MainWindow::m_openAutomaton [private]`

Definition at line 39 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.18 `m_openAutomatonBt`

`QToolButton* MainWindow::m_openAutomatonBt [private]`

Definition at line 50 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.19 m_openIcon

`QIcon MainWindow::m_openIcon [private]`

Definition at line 30 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.20 m_pauseIcon

`QIcon MainWindow::m_pauseIcon [private]`

Definition at line 27 of file [mainwindow.h](#).

Referenced by [createIcons\(\)](#).

6.5.4.21 m_playIcon

`QIcon MainWindow::m_playIcon [private]`

Definition at line 26 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.22 m_playPause

`QAction* MainWindow::m_playPause [private]`

Actions.

Definition at line 34 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.23 m_playPauseBt

`QToolButton* MainWindow::m_playPauseBt [private]`

Buttons.

Definition at line 45 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.24 m_previousState

`QAction* MainWindow::m_previousState [private]`

Definition at line 36 of file [mainwindow.h](#).

6.5.4.25 m_previousStateBt

`QPushButton* MainWindow::m_previousStateBt [private]`

Definition at line 47 of file [mainwindow.h](#).

6.5.4.26 m_resetAutomaton

`QAction* MainWindow::m_resetAutomaton [private]`

Definition at line 42 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.27 m_resetBt

`QPushButton* MainWindow::m_resetBt [private]`

Definition at line 53 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.28 m_resetIcon

`QIcon MainWindow::m_resetIcon [private]`

Definition at line 31 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.29 m_saveAutomaton

`QAction* MainWindow::m_saveAutomaton [private]`

Definition at line 40 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#).

6.5.4.30 m_saveAutomatonBt

`QPushButton* MainWindow::m_saveAutomatonBt [private]`

Definition at line 51 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createToolBar\(\)](#).

6.5.4.31 m_saveIcon

`QIcon MainWindow::m_saveIcon [private]`

Definition at line 29 of file [mainwindow.h](#).

Referenced by [createActions\(\)](#), and [createIcons\(\)](#).

6.5.4.32 m_speedLabel

`QLabel* MainWindow::m_speedLabel [private]`

Simulation speed input.

Definition at line 57 of file [mainwindow.h](#).

Referenced by [createToolBar\(\)](#).

6.5.4.33 m_tabs

`QTabWidget* MainWindow::m_tabs [private]`

Definition at line 20 of file [mainwindow.h](#).

Referenced by [closeTab\(\)](#), [createTabs\(\)](#), [getBoard\(\)](#), [MainWindow\(\)](#), [nextState\(\)](#), [openFile\(\)](#), [saveToFile\(\)](#), and [setCellHandler\(\)](#).

6.5.4.34 m_toolBar

`QToolBar* MainWindow::m_toolBar [private]`

Definition at line 59 of file [mainwindow.h](#).

Referenced by [createToolBar\(\)](#).

The documentation for this class was generated from the following files:

- [mainwindow.h](#)
- [mainwindow.cpp](#)

Chapter 7

File Documentation

7.1 cell.cpp File Reference

```
#include "cell.h"
```

7.2 cell.cpp

```
00001 #include "cell.h"
00002
00007 Cell::Cell(unsigned int state):
00008     m_state(state), m_nextState(state)
00009 {
00010
00011 }
00012
00020 void Cell::setState(unsigned int state)
00021 {
00022     m_nextState = state;
00023 }
00024
00030 void Cell::validState()
00031 {
00032     m_state = m_nextState;
00033 }
00034
00041 void Cell::forceState(unsigned int state)
00042 {
00043     m_state = m_nextState = state;
00044 }
00045
00048 unsigned int Cell::getState() const
00049 {
00050     return m_state;
00051 }
00052
00060 bool Cell::addNeighbour(const Cell* neighbour, const QVector<short> relativePosition)
00061 {
00062     if (m_neighbours.count(relativePosition))
00063         return false;
00064
00065     m_neighbours.insert(relativePosition, neighbour);
00066     return true;
00067 }
00068
00073 QMap<QVector<short>, const Cell *> Cell::getNeighbours() const
00074 {
00075     return m_neighbours;
00076 }
00077
00080 const Cell *Cell::getNeighbour(QVector<short> relativePosition) const
00081 {
00082     return m_neighbours.value(relativePosition, nullptr);
```

```

00083 }
00084
00091 QVector<short> Cell::getRelativePosition(const QVector<unsigned int> cellPosition,
    const QVector<unsigned int> neighbourPosition)
00092 {
00093     if (cellPosition.size() != neighbourPosition.size())
00094     {
00095         throw QString(QObject::tr("Different size of position vectors"));
00096     }
00097     QVector<short> relativePosition;
00098     for (short i = 0; i < cellPosition.size(); i++)
00099         relativePosition.push_back(neighbourPosition.at(i) - cellPosition.at(i));
00100
00101     return relativePosition;
00102 }

```

7.3 cell.h File Reference

```

#include <QVector>
#include <QDebug>

```

Classes

- class [Cell](#)
Contains the state, the next state and the neighbours.

7.4 cell.h

```

00001 #ifndef CELL_H
00002 #define CELL_H
00003
00004 #include <QVector>
00005 #include <QDebug>
00006
00010 class Cell
00011 {
00012 public:
00013     Cell(unsigned int state = 0);
00014
00015     void setState(unsigned int state);
00016     void validState();
00017     void forceState(unsigned int state);
00018     unsigned int getState() const;
00019
00020     bool addNeighbour(const Cell* neighbour, const QVector<short> relativePosition);
00021     QMap<QVector<short>, const Cell*> getNeighbours() const;
00022     const Cell* getNeighbour(QVector<short> relativePosition) const;
00023
00024     static QVector<short> getRelativePosition(const QVector<unsigned int> cellPosition,
    const QVector<unsigned int> neighbourPosition);
00025
00026 private:
00027     unsigned int m_state;
00028     unsigned int m_nextState;
00029
00030     QMap<QVector<short>, const Cell*> m_neighbours;
00031 };
00032
00033 #endif // CELL_H

```

7.5 cellhandler.cpp File Reference

```

#include <iostream>
#include "cellhandler.h"

```

7.6 cellhandler.cpp

```

00001 #include <iostream>
00002 #include "cellhandler.h"
00003
00025 CellHandler::CellHandler(const QString filename)
00026 {
00027     QFile loadFile(filename);
00028     if (!loadFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
00029         qWarning("Couldn't open given file.");
00030         throw QString(QObject::tr("Couldn't open given file"));
00031     }
00032
00033     QJsonParseError parseErr;
00034     QJsonDocument loadDoc(QJsonDocument::fromJson(loadFile.readAll(), &parseErr));
00035
00036
00037
00038     if (loadDoc.isNull() || loadDoc.isEmpty()) {
00039         qWarning() << "Could not read data : ";
00040         qWarning() << parseErr.errorString();
00041         throw QString(parseErr.errorString());
00042     }
00043
00044     // Loading of the json file
00045     if (!loadDoc.isObject())
00046     {
00047         qWarning("File not valid");
00048         throw QString(QObject::tr("File not valid"));
00049     }
00050
00051     foundNeighbours();
00052
00053 }
00054
00055
00065 CellHandler::CellHandler(const QVector<unsigned int> dimensions,
00066     generationTypes type, unsigned int stateMax, unsigned int density)
00067 {
00068     m_dimensions = dimensions;
00069     QVector<unsigned int> position;
00070     unsigned int size = 1;
00071
00072     // Set position vector to 0
00073     for (unsigned short i = 0; i < m_dimensions.size(); i++)
00074     {
00075         position.push_back(0);
00076         size *= m_dimensions.at(i);
00077     }
00078
00079     // Creation of cells
00080     for (unsigned int j = 0; j < size; j++)
00081     {
00082         m_cells.insert(position, new Cell(0));
00083         positionIncrement(position);
00084     }
00085
00086     foundNeighbours();
00087
00088     if (type != empty)
00089         generate(type, stateMax, density);
00090
00091 }
00092
00093
00094
00097 CellHandler::~CellHandler()
00098 {
00099     for (QMap<QVector<unsigned int>, Cell* >::iterator it = m_cells.begin(); it !=
00100         m_cells.end(); ++it)
00101     {
00102         delete it.value();
00103     }
00104
00107 Cell *CellHandler::getCell(const QVector<unsigned int> position) const
00108 {
00109     return m_cells.value(position);
00110 }
00111
00114 QVector<unsigned int> CellHandler::getDimensions() const
00115 {
00116     return m_dimensions;
00117 }
00118

```

```

00121 void CellHandler::nextStates() const
00122 {
00123     for (QMap<QVector<unsigned int>, Cell* >::const_iterator it =
         m_cells.begin(); it != m_cells.end(); ++it)
00124     {
00125         it.value()->validState();
00126     }
00127 }
00128
00136 bool CellHandler::save(QString filename) const
00137 {
00138     QFile saveFile(filename);
00139     if (!saveFile.open(QIODevice::WriteOnly)) {
00140         qWarning("Couldn't create or open given file.");
00141         throw QString(QObject::tr("Couldn't create or open given file"));
00142     }
00143
00144     QJsonObject json;
00145     QString stringDimension;
00146     // Creation of the dimension string
00147     for (int i = 0; i < m_dimensions.size(); i++)
00148     {
00149         if (i != 0)
00150             stringDimension.push_back("x");
00151         stringDimension.push_back(QString::number(m_dimensions.at(i)));
00152     }
00153     json["dimensions"] = QJsonValue(stringDimension);
00154
00155     QJsonArray cells;
00156     for (CellHandler::const_iterator it = begin(); it !=
end(); ++it)
00157     {
00158         cells.append(QJsonValue((int)it->getState()));
00159     }
00160     json["cells"] = cells;
00161
00162     QJsonDocument saveDoc(json);
00163     saveFile.write(saveDoc.toJson());
00164
00165     saveFile.close();
00166     return true;
00167 }
00168
00169
00176 void CellHandler::generate(CellHandler::generationTypes
type, unsigned int stateMax, unsigned short density)
00177 {
00178     if (type == random)
00179     {
00180         QVector<unsigned int> position;
00181         for (unsigned short i = 0; i < m_dimensions.size(); i++)
00182         {
00183             position.push_back(0);
00184         }
00185         QRandomGenerator generator((float)qrand()*(float)time_t()/RAND_MAX);
00186         for (int j = 0; j < m_cells.size(); j++)
00187         {
00188             unsigned int state = 0;
00189             // 0 have (1-density)% of chance of being generate
00190             if (generator.generateDouble()*100.0 < density)
00191                 state = (float)(generator.generateDouble()*stateMax) +1;
00192             if (state > stateMax)
00193                 state = stateMax;
00194             m_cells.value(position)->forceState(state);
00195
00196             positionIncrement(position);
00197         }
00198     }
00199     else if (type == symetric)
00200     {
00201         QVector<unsigned int> position;
00202         for (short i = 0; i < m_dimensions.size(); i++)
00203         {
00204             position.push_back(0);
00205         }
00206
00207         QRandomGenerator generator((float)qrand()*(float)time_t()/RAND_MAX);
00208         QVector<unsigned int> savedStates;
00209         for (int j = 0; j < m_cells.size(); j++)
00210         {
00211             if (j % m_dimensions.at(0) == 0)
00212                 savedStates.clear();
00213             if (j % m_dimensions.at(0) < (m_dimensions.at(0)+1) / 2)
00214             {
00215                 unsigned int state = 0;
00216                 // 0 have (1-density)% of chance of being generate
00217                 if (generator.generateDouble()*100.0 < density)

```

```

00218         state = (float)(generator.generateDouble()*stateMax) +1;
00219         if (state > stateMax)
00220             state = stateMax;
00221         savedStates.push_back(state);
00222         m_cells.value(position)->forceState(state);
00223     }
00224     else
00225     {
00226         unsigned int i = savedStates.size() - (j % m_dimensions.at(0) - (
m_dimensions.at(0)-1)/2 + (m_dimensions.at(0) % 2 == 0 ? 0 : 1));
00227         m_cells.value(position)->forceState(savedStates.at(i));
00228     }
00229     positionIncrement(position);
00230
00231
00232     }
00233 }
00234 }
00235 }
00236
00241 void CellHandler::print(std::ostream &stream) const
00242 {
00243     for (const_iterator it = begin(); it != end(); ++it)
00244     {
00245         for (unsigned int d = 0; d < it.changedDimension(); d++)
00246             stream << std::endl;
00247         stream << it->getState() << " ";
00248     }
00249 }
00250
00251 }
00252
00255 CellHandler::iterator CellHandler::begin()
00256 {
00257     return iterator(this);
00258 }
00259
00262 CellHandler::const_iterator CellHandler::begin() const
00263 {
00264     return const_iterator(this);
00265 }
00266
00271 bool CellHandler::end() const
00272 {
00273     return true;
00274 }
00275
00306 bool CellHandler::load(const QJsonObject &json)
00307 {
00308     if (!json.contains("dimensions") || !json["dimensions"].isString())
00309         return false;
00310
00311     // RegExp to validate dimensions field format : "10x10"
00312     QRegExpValidator dimensionValidator(QRegExp("[0-9]*x?"));
00313     QString stringDimensions = json["dimensions"].toString();
00314     int pos= 0;
00315     if (dimensionValidator.validate(stringDimensions, pos) != QRegExpValidator::Acceptable)
00316         return false;
00317
00318     // Split of dimensions field : "10x10" => "10", "10"
00319     QRegExp rx("x");
00320     QStringList list = json["dimensions"].toString().split(rx, QString::SkipEmptyParts);
00321
00322     int product = 1;
00323     // Dimensions construction
00324     for (int i = 0; i < list.size(); i++)
00325     {
00326         product = product * list.at(i).toInt();
00327         m_dimensions.push_back(list.at(i).toInt());
00328     }
00329     if (!json.contains("cells") || !json["cells"].isArray())
00330         return false;
00331
00332     QJsonArray cells = json["cells"].toArray();
00333     if (cells.size() != product)
00334         return false;
00335
00336     QVector<unsigned int> position;
00337     // Set position vector to 0
00338     for (unsigned short i = 0; i < m_dimensions.size(); i++)
00339     {
00340         position.push_back(0);
00341     }
00342
00343     // Creation of cells
00344     for (int j = 0; j < cells.size(); j++)
00345     {

```

```

00346         if (!cells.at(j).isDouble())
00347             return false;
00348         if (cells.at(j).toDouble() < 0)
00349             return false;
00350         m_cells.insert(position, new Cell(cells.at(j).toDouble()));
00351
00352         positionIncrement(position);
00353     }
00354
00355     return true;
00356 }
00357 }
00358
00364 void CellHandler::foundNeighbours()
00365 {
00366     QVector<unsigned int> currentPosition;
00367     // Set position vector to 0
00368     for (unsigned short i = 0; i < m_dimensions.size(); i++)
00369     {
00370         currentPosition.push_back(0);
00371     }
00372     // Modification of all the cells
00373     for (int j = 0; j < m_cells.size(); j++)
00374     {
00375         // Get the list of the neighbours positions
00376         // This function is recursive
00377         QVector<QVector<unsigned int> > listPosition(getListNeighboursPositions(
currentPosition));
00378
00379         // Adding neighbours
00380         for (int i = 0; i < listPosition.size(); i++)
00381             m_cells.value(currentPosition)->addNeighbour(m_cells.value(listPosition.at(i)),
Cell::getRelativePosition(currentPosition, listPosition.at(i)));
00382         positionIncrement(currentPosition);
00383     }
00384 }
00385 }
00386
00394 void CellHandler::positionIncrement(QVector<unsigned int> &pos, unsigned int
value) const
00395 {
00396     pos.replace(0, pos.at(0) + value); // adding the value to the first digit
00397
00398     // Carry management
00399     for (unsigned short i = 0; i < m_dimensions.size(); i++)
00400     {
00401         if (pos.at(i) >= m_dimensions.at(i) && pos.at(i) <
m_dimensions.at(i)*2)
00402         {
00403             pos.replace(i, 0);
00404             if (i + 1 != m_dimensions.size())
00405                 pos.replace(i+1, pos.at(i+1)+1);
00406         }
00407         else if (pos.at(i) >= m_dimensions.at(i))
00408         {
00409             pos.replace(i, pos.at(i) - m_dimensions.at(i));
00410             if (i + 1 != m_dimensions.size())
00411                 pos.replace(i+1, pos.at(i+1)+1);
00412             i--;
00413         }
00414     }
00415 }
00416 }
00417
00423 QVector<QVector<unsigned int> > & CellHandler::getListNeighboursPositions
(const QVector<unsigned int> position) const
00424 {
00425     QVector<QVector<unsigned int> > *list = getListNeighboursPositionsRecursive
(position, position.size(), position);
00426     // We remove the position of the cell
00427     list->removeAll(position);
00428     return *list;
00429 }
00430
00464 QVector<QVector<unsigned int> > *
CellHandler::getListNeighboursPositionsRecursive(const
QVector<unsigned int> position, unsigned int dimension, QVector<unsigned int> lastAdd) const
00465 {
00466     if (dimension == 0) // Stop condition
00467     {
00468         QVector<QVector<unsigned int> > *list = new QVector<QVector<unsigned int> >;
00469         return list;
00470     }
00471     QVector<QVector<unsigned int> > *listPositions = new QVector<QVector<unsigned int> >;
00472
00473     QVector<unsigned int> modifiedPosition(lastAdd);
00474

```

```

00475 // "x_d - 1" tree
00476 if (modifiedPosition.at(dimension-1) != 0) // Avoid "negative" position
00477     modifiedPosition.replace(dimension-1, position.at(dimension-1) - 1);
00478 listPositions->append(*getListNeighboursPositionsRecursive(position,
dimension -1, modifiedPosition));
00479 if (!listPositions->count(modifiedPosition))
00480     listPositions->push_back(modifiedPosition);
00481
00482 // "x_d" tree
00483 modifiedPosition.replace(dimension-1, position.at(dimension-1));
00484 listPositions->append(*getListNeighboursPositionsRecursive(position,
dimension -1, modifiedPosition));
00485 if (!listPositions->count(modifiedPosition))
00486     listPositions->push_back(modifiedPosition);
00487
00488 // "x_d + 1" tree
00489 if (modifiedPosition.at(dimension - 1) + 1 < m_dimensions.at(dimension-1)) // Avoid position
out of the cell space
00490     modifiedPosition.replace(dimension-1, position.at(dimension-1) +1);
00491 listPositions->append(*getListNeighboursPositionsRecursive(position,
dimension -1, modifiedPosition));
00492 if (!listPositions->count(modifiedPosition))
00493     listPositions->push_back(modifiedPosition);
00494
00495 return listPositions;
00496 }
00497 }
00498
00503 template<typename CellHandler_T, typename Cell_T>
00504 CellHandler::iteratorT<CellHandler_T,Cell_T>::iteratorT
(CellHandler_T *handler):
00505     m_handler(handler), m_changedDimension(0)
00506 {
00507     // Initialisation of m_position
00508     for (unsigned short i = 0; i < handler->m_dimensions.size(); i++)
00509     {
00510         m_position.push_back(0);
00511     }
00512     m_zero = m_position;
00513 }
00514
00517 template<typename CellHandler_T, typename Cell_T>
00518 CellHandler::iteratorT<CellHandler_T,Cell_T> &
CellHandler::iteratorT<CellHandler_T,Cell_T>::operator++
()
00519 {
00520     m_position.replace(0, m_position.at(0) + 1); // adding the value to the first digit
00521
00522     m_changedDimension = 0;
00523     // Carry management
00524     for (unsigned short i = 0; i < m_handler->m_dimensions.size(); i++)
00525     {
00526         if (m_position.at(i) >= m_handler->m_dimensions.at(i))
00527         {
00528             m_position.replace(i, 0);
00529             m_changedDimension++;
00530             if (i + 1 != m_handler->m_dimensions.size())
00531                 m_position.replace(i+1, m_position.at(i+1)+1);
00532         }
00533     }
00534
00535     // If we return to zero, we have finished
00536     if (m_position == m_zero)
00537         m_finished = true;
00538
00539     return *this;
00540 }
00541 }
00542
00545 template<typename CellHandler_T, typename Cell_T>
00546 Cell_T* CellHandler::iteratorT<CellHandler_T,Cell_T>::operator->
() const
00547 {
00548     return m_handler->m_cells.value(m_position);
00549 }
00550
00551
00554 template<typename CellHandler_T, typename Cell_T>
00555 Cell_T *CellHandler::iteratorT<CellHandler_T,Cell_T>::operator*
() const
00556 {
00557     return m_handler->m_cells.value(m_position);
00558 }
00559
00565 template<typename CellHandler_T, typename Cell_T>
00566 unsigned int CellHandler::iteratorT<CellHandler_T,Cell_T>::changedDimension
() const

```

```

00567 {
00568     return m_changedDimension;
00569 }
00570

```

7.7 cellhandler.h File Reference

```

#include <QString>
#include <QFile>
#include <QJsonDocument>
#include <QtWidgets>
#include <QMap>
#include <QRegExpValidator>
#include <QDebug>
#include "cell.h"

```

Classes

- class [CellHandler](#)
Cell container and cell generator.
- class [CellHandler::iteratorT< CellHandler_T, Cell_T >](#)
Implementation of iterator design pattern with a template to generate iterator and const_iterator at the same time.

7.8 cellhandler.h

```

00001 #ifndef CELLHANDLER_H
00002 #define CELLHANDLER_H
00003
00004 #include <QString>
00005 #include <QFile>
00006 #include <QJsonDocument>
00007 #include <QtWidgets>
00008 #include <QMap>
00009 #include <QRegExpValidator>
00010 #include <QDebug>
00011
00012 #include "cell.h"
00013
00014
00015
00020 class CellHandler
00021 {
00022
00040     template <typename CellHandler_T, typename Cell_T>
00041     class iteratorT
00042     {
00043     friend class CellHandler;
00044     public:
00045         iteratorT(CellHandler_T* handler);
00046
00047         iteratorT& operator++();
00048         Cell_T* operator->() const;
00049         Cell_T* operator*() const;
00050
00051         bool operator!=(bool finished) const { return (m_finished != finished); }
00052         unsigned int changedDimension() const;
00053
00054
00055     private:
00056         CellHandler_T *m_handler;
00057         QVector<unsigned int> m_position;
00058         bool m_finished = false;
00059         QVector<unsigned int> m_zero;
00060

```



```

00061         unsigned int m_changedDimension;
00062     };
00063 public:
00064     typedef iteratorT<const CellHandler, const Cell>
const_iterator;
00065     typedef iteratorT<CellHandler, Cell> iterator;
00066
00069     enum generationTypes {
00070         empty,
00071         random,
00072         symetric
00073     };
00074
00075     CellHandler(const QString filename);
00076     CellHandler(const QVector<unsigned int> dimensions,
generationTypes type = empty, unsigned int stateMax = 1, unsigned int density = 20);
00077     virtual ~CellHandler();
00078
00079     Cell* getCell(const QVector<unsigned int> position) const;
00080     QVector<unsigned int> getDimensions() const;
00081     void nextStates() const;
00082
00083     bool save(QString filename) const;
00084
00085     void generate(generationTypes type, unsigned int stateMax = 1, unsigned short
density = 50);
00086     void print(std::ostream &stream) const;
00087
00088     const_iterator begin() const;
00089     iterator begin();
00090     bool end() const;
00091
00092 private:
00093     bool load(const QJsonObject &json);
00094     void foundNeighbours();
00095     void positionIncrement(QVector<unsigned int> &pos, unsigned int value = 1) const;
00096     QVector<QVector<unsigned int> > *getListNeighboursPositionsRecursive
(const QVector<unsigned int> position, unsigned int dimension, QVector<unsigned int> lastAdd) const;
00097     QVector<QVector<unsigned int> > &getListNeighboursPositions(const
QVector<unsigned int> position) const;
00098
00099     QVector<unsigned int> m_dimensions;
00100     QMap<QVector<unsigned int>, Cell* > m_cells;
00101 };
00102
00103 template class CellHandler::iteratorT<CellHandler, Cell>;
00104 template class CellHandler::iteratorT<const CellHandler, const Cell>
;
00105
00106 #endif // CELLHANDLER_H

```

7.9 creationdialog.cpp File Reference

```

#include "creationdialog.h"
#include <iostream>

```

7.10 creationdialog.cpp

```

00001 #include "creationdialog.h"
00002 #include <iostream>
00003
00004
00005 CreationDialog::CreationDialog(QWidget *parent)
00006 {
00007     QLabel *m_dimLabel= new QLabel(tr("Write your dimensions and their size, separated by a comma.\n"
00008         "For instance, '25,25 ' will create a 2-dimensional 25x25 Automaton. "));
00009     QLabel *m_densityLabel = new QLabel(tr("Density :"));
00010     QLabel *m_stateMaxLabel = new QLabel(tr("Max state :"));
00011     m_densityBox = new QSpinBox();
00012     m_densityBox->setValue(20);
00013     m_stateMaxBox = new QSpinBox();
00014     m_stateMaxBox->setValue(1);
00015
00016     QHBoxLayout *densityLayout = new QHBoxLayout();

```

```

00017     densityLayout->addWidget(m_densityLabel);
00018     densityLayout->addWidget(m_densityBox);
00019
00020     QHBoxLayout *stateMaxLayout = new QHBoxLayout();
00021     stateMaxLayout->addWidget(m_stateMaxLabel);
00022     stateMaxLayout->addWidget(m_stateMaxBox);
00023
00024     m_dimensionsEdit = new QLineEdit;
00025     QRegExp rgx("[0-9]+,)*");
00026     QRegExpValidator *v = new QRegExpValidator(rgx, this);
00027     m_dimensionsEdit->setValidator(v);
00028     m_doneBt = new QPushButton(tr("Done !"));
00029
00030     QVBoxLayout *layout = new QVBoxLayout;
00031
00032     QGroupBox *grpBox = createGenButtons();
00033
00034     layout->addWidget(m_dimLabel);
00035     layout->addWidget(m_dimensionsEdit);
00036     layout->addLayout(densityLayout);
00037     layout->addLayout(stateMaxLayout);
00038     layout->addWidget(grpBox);
00039     layout->addWidget(m_doneBt);
00040     setLayout(layout);
00041
00042     connect(m_doneBt, SIGNAL(clicked(bool)), this, SLOT(processSettings()));
00043
00044 }
00045
00051 QGroupBox *CreationDialog::createGenButtons(){
00052     m_groupBox = new QGroupBox(tr("Cell generation settings"));
00053     m_empGen = new QRadioButton(tr("&Empty Board"));
00054     m_randGen = new QRadioButton(tr("&Random"));
00055     m_symGen = new QRadioButton(tr("&Symmetrical"));
00056
00057     QVBoxLayout *layout = new QVBoxLayout;
00058     layout->addWidget(m_empGen);
00059     layout->addWidget(m_randGen);
00060     layout->addWidget(m_symGen);
00061
00062     m_groupBox->setLayout(layout);
00063
00064     return m_groupBox;
00065 }
00066
00072 void CreationDialog::processSettings(){
00073     QString dimensions = m_dimensionsEdit->text();
00074     if(dimensions.length() == 0){
00075         QMessageBox messageBox;
00076         messageBox.critical(0,"Error","You must specify valid dimensions !");
00077         messageBox.setFixedSize(500,200);
00078     }
00079     else{
00080         CellHandler::generationTypes genType;
00081         if(m_symGen->isChecked()) genType = CellHandler::generationTypes::symetric;
00082         else if(m_randGen->isChecked()) genType = CellHandler::generationTypes::random;
00083         else genType = CellHandler::generationTypes::empty;
00084         QStringList dimList = m_dimensionsEdit->text().split(",");
00085         QVector<unsigned int> dimensions;
00086         for(int i = 0; i < dimList.size(); i++) dimensions.append(dimList.at(i).toInt());
00087
00088         emit settingsFilled(dimensions, genType, m_stateMaxBox->value(),
00089             m_densityBox->value());
00089         this->close();
00090     }
00091
00092 }
00093

```

7.11 creationdialog.h File Reference

```

#include <QtWidgets>
#include "cellhandler.h"

```

Classes

- class [CreationDialog](#)
Automaton creation dialog box.

7.12 creationdialog.h

```

00001 #ifndef CREATIONDIALOG_H
00002 #define CREATIONDIALOG_H
00003
00004 #include <QtWidgets>
00005 #include "cellhandler.h"
00006
00013 class CreationDialog : public QDialog
00014 {
00015     Q_OBJECT
00016
00017 public:
00018     CreationDialog(QWidget *parent = 0);
00019
00020 signals:
00021     void settingsFilled(const QVector<unsigned int> dimensions,
00022                         CellHandler::generationTypes type =
00023                         CellHandler::generationTypes::empty,
00024                         unsigned int stateMax = 1, unsigned int density = 20);
00025
00026 public slots:
00027     void processSettings();
00028
00029 private:
00030     QLineEdit *m_dimensionsEdit;
00031     QSpinBox *m_densityBox;
00032     QSpinBox *m_stateMaxBox;
00033     QPushButton *m_doneBt;
00034
00035     QGroupBox *m_groupBox;
00036     QRadioButton *m_empGen;
00037     QRadioButton *m_randGen;
00038     QRadioButton *m_symGen;
00039
00040     QGroupBox *createGenButtons();
00041
00042
00043
00044
00045
00046 };
00047
00048 #endif // CREATIONDIALOG_H

```

7.13 main.cpp File Reference

```

#include <QApplication>
#include <QDebug>
#include "cell.h"
#include "mainwindow.h"

```

Functions

- int [main](#) (int argc, char *argv[])

7.13.1 Function Documentation

7.13.1.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

Definition at line 6 of file [main.cpp](#).

7.14 main.cpp

```
00001 #include <QApplication>
00002 #include <QDebug>
00003 #include "cell.h"
00004 #include "mainwindow.h"
00005
00006 int main(int argc, char * argv[])
00007 {
00008     QApplication app(argc, argv);
00009     QApplication::setAttribute(Qt::AA_UseHighDpiPixmaps);
00010     MainWindow w;
00011     w.show();
00012     return app.exec();
00013
00014 }
```

7.15 mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include <iostream>
```

7.16 mainwindow.cpp

```
00001 #include "mainwindow.h"
00002 #include <iostream>
00003 MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent)
00004 {
00005     createIcons();
00006     createActions();
00007     createToolBar();
00008
00009
00010     setMinimumSize(500,500);
00011     setWindowTitle("AutoCell");
00012
00013     m_tabs = NULL;
00014 }
00015
00021 void MainWindow::createIcons(){
00022     QPixmap fastBackwardPm(":/icons/icons/fast-backward.svg");
00023     QPixmap fastBackwardHoveredPm(":/icons/icons/fast-backward-full.svg");
00024     QPixmap fastForwardPm(":/icons/icons/fast-forward.svg");
00025     QPixmap fastForwardHoveredPm(":/icons/icons/fast-forward-full.svg");
00026     QPixmap playPm(":/icons/icons/play.svg");
00027     QPixmap playHoveredPm(":/icons/icons/play-full.svg");
00028     QPixmap newPm(":/icons/icons/new.svg");
00029     QPixmap openPm(":/icons/icons/open.svg");
00030     QPixmap savePm(":/icons/icons/save.svg");
00031     QPixmap pausePm(":/icons/icons/pause.svg");
00032     QPixmap resetPm(":/icons/icons/reset.svg");
00033
00034     m_fastBackwardIcon.addPixmap(fastBackwardPm, QIcon::Normal, QIcon::Off);
00035     m_fastBackwardIcon.addPixmap(fastBackwardHoveredPm, QIcon::Active, QIcon::Off);
00036     m_fastForwardIcon.addPixmap(fastForwardPm, QIcon::Normal, QIcon::Off);
00037     m_fastForwardIcon.addPixmap(fastForwardHoveredPm, QIcon::Active, QIcon::Off);
```

```

00038     m_playIcon.addPixmap(playPm, QIcon::Normal, QIcon::Off);
00039     m_playIcon.addPixmap(playHoveredPm, QIcon::Active, QIcon::Off);
00040     m_pauseIcon.addPixmap(pausePm, QIcon::Normal, QIcon::Off);
00041     m_newIcon.addPixmap(newPm, QIcon::Normal, QIcon::Off);
00042     m_saveIcon.addPixmap(savePm, QIcon::Normal, QIcon::Off);
00043     m_openIcon.addPixmap(openPm, QIcon::Normal, QIcon::Off);
00044     m_resetIcon.addPixmap(resetPm, QIcon::Normal, QIcon::Off);
00045 }
00046
00051 void MainWindow::createActions(){
00052     m_fastBackward = new QAction(m_fastBackwardIcon, tr("&fast backward"),
00053     this);
00054     m_fastForward = new QAction(m_fastForwardIcon, tr("&fast forward"), this);
00055
00054     m_playPause = new QAction(m_playIcon, tr("Play"), this);
00055     m_saveAutomaton = new QAction(m_saveIcon, tr("Save automaton"), this);
00056     m_newAutomaton = new QAction(m_newIcon, tr("New automaton"), this);
00057     m_openAutomaton = new QAction(m_openIcon, tr("Open automaton"), this);
00058     m_resetAutomaton = new QAction(m_resetIcon, tr("Reset automaton"), this);
00059
00060
00061
00062     m_fastBackwardBt = new QToolButton(this);
00063     m_fastForwardBt = new QToolButton(this);
00064     m_playPauseBt = new QToolButton(this);
00065     m_saveAutomatonBt = new QToolButton(this);
00066     m_newAutomatonBt = new QToolButton(this);
00067     m_openAutomatonBt = new QToolButton(this);
00068     m_resetBt = new QToolButton(this);
00069
00070     m_fastBackwardBt->setDefaultAction(m_fastBackward);
00071     m_fastForwardBt->setDefaultAction(m_fastForward);
00072     m_playPauseBt->setDefaultAction(m_playPause);
00073     m_saveAutomatonBt->setDefaultAction(m_saveAutomaton);
00074     m_newAutomatonBt->setDefaultAction(m_newAutomaton);
00075     m_openAutomatonBt->setDefaultAction(m_openAutomaton);
00076     m_resetBt->setDefaultAction(m_resetAutomaton);
00077
00078     m_fastBackwardBt->setIconSize(QSize(30,30));
00079     m_fastForwardBt->setIconSize(QSize(30,30));
00080     m_playPauseBt->setIconSize(QSize(30,30));
00081     m_saveAutomatonBt->setIconSize(QSize(30,30));
00082     m_newAutomatonBt->setIconSize(QSize(30,30));
00083     m_openAutomatonBt->setIconSize(QSize(30,30));
00084     m_resetBt->setIconSize(QSize(30,30));
00085
00086     connect(m_openAutomatonBt, SIGNAL(clicked(bool)), this, SLOT(
00087     openFile()));
00087     connect(m_newAutomatonBt, SIGNAL(clicked(bool)), this, SLOT(
00088     openCreationWindow()));
00088     connect(m_saveAutomatonBt, SIGNAL(clicked(bool)), this, SLOT(
00089     saveToFile()));
00089     connect(m_fastForwardBt, SIGNAL(clicked(bool)), this, SLOT(
00090     forward()));
00090
00091 }
00092
00097 void MainWindow::createToolBar(){
00098     m_toolBar = new QToolBar(this);
00099     QLabel *m_speedLabel = new QLabel(tr("Speed : "),this);
00100     m_jumpSpeed = new QSpinBox(this);
00101     m_jumpSpeed->setValue(1);
00102     m_speedLabel->setFixedWidth(50);
00103     m_jumpSpeed->setFixedWidth(40);
00104     m_toolBar->setMovable(false);
00105
00106     QHBoxLayout *tbLayout = new QHBoxLayout(this);
00107     tbLayout->addWidget(m_newAutomatonBt, Qt::AlignCenter);
00108     tbLayout->addWidget(m_openAutomatonBt, Qt::AlignCenter);
00109     tbLayout->addWidget(m_saveAutomatonBt, Qt::AlignCenter);
00110     tbLayout->addWidget(m_fastBackwardBt, Qt::AlignCenter);
00111     tbLayout->addWidget(m_playPauseBt, Qt::AlignCenter);
00112     tbLayout->addWidget(m_fastForwardBt, Qt::AlignCenter);
00113     tbLayout->addWidget(m_speedLabel, Qt::AlignCenter);
00114     tbLayout->addWidget(m_jumpSpeed, Qt::AlignCenter);
00115     tbLayout->addWidget(m_resetBt, Qt::AlignCenter);
00116
00117
00118     tbLayout->setAlignment(Qt::AlignCenter);
00119     QWidget* wrapper = new QWidget(this);
00120     wrapper->setLayout(tbLayout);
00121     m_toolBar->addWidget(wrapper);
00122     addToolBar(m_toolBar);
00123
00124
00125 }
00126

```

```

00131 QWidget* MainWindow::createTab(){
00132     QWidget *tab = new QWidget(this);
00133     QVBoxLayout *layout = new QVBoxLayout(this);
00134
00135     QVector<unsigned int> dimensions = m_cellHandlers.last()->getDimensions();
00136     int boardVSize = 0;
00137     int boardHSize = 0;
00138     if(dimensions.size() > 1){
00139         boardVSize = dimensions[0];
00140         boardHSize = dimensions[1];
00141     }
00142     else{
00143         boardVSize = 1;
00144         boardHSize = dimensions[0];
00145     }
00146
00147     QTableWidgetItem* board = new QTableWidgetItem(boardVSize, boardHSize, this);
00148     board->setFixedSize(boardHSize*m_cellSize,boardVSize*
m_cellSize);
00149     //setMinimumSize(m_boardHSize*m_cellSize,100+m_boardVSize*m_cellSize);
00150     board->horizontalHeader()->setVisible(false);
00151     board->verticalHeader()->setVisible(false);
00152     board->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
00153     board->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
00154     board->setEditTriggers(QAbstractItemView::NoEditTriggers);
00155     for(unsigned int col = 0; col < boardHSize; ++col)
00156         board->setColumnWidth(col, m_cellSize);
00157     for(unsigned int row = 0; row < boardVSize; ++row) {
00158         board->setRowHeight(row, m_cellSize);
00159         for(unsigned int col = 0; col < boardHSize; ++col) {
00160             board->setItem(row, col, new QTableWidgetItem(""));
00161             board->item(row, col)->setBackgroundColor("white");
00162             board->item(row, col)->setTextColor("black");
00163         }
00164     }
00165     QScrollArea *scrollArea = new QScrollArea(this);
00166     scrollArea->setWidget(board);
00167     layout->setContentsMargins(0,0,0,0);
00168     layout->addWidget(scrollArea);
00169     tab->setLayout(layout);
00170     return tab;
00171 }
00172
00176 void MainWindow::openFile(){
00177     QString fileName = QFileDialog::getOpenFileName(this, tr("Open Cell file"), ".",
tr("Automaton cell files (*.atc)"));
00178
00179     if(!fileName.isEmpty()){
00180         m_cellHandlers.append(new CellHandler(fileName));
00181         std::cout << "m_cellHandlers size : " <<m_cellHandlers.size() << std::endl<<std::flush
;
00182         if(m_tabs == NULL) createTabs();
00183         m_tabs->addTab(createTab(), "Automaton "+ QString::number(
m_cellHandlers.size()));
00184         updateBoard(m_cellHandlers.size()-1);
00185     }
00186 }
00187
00192 void MainWindow::saveToFile(){
00193     if(m_cellHandlers.size() > 0){
00194         QString fileName = QFileDialog::getSaveFileName(this, tr("Save Automaton"),
".", tr("Automaton Cells file (*.atc)"));
00195         m_cellHandlers[m_tabs->currentIndex()->save(fileName+".atc");
00196     }
00197     else{
00198         QMessageBox msgBox;
00199         msgBox.critical(0,"Error","Please create or import an Automaton first !");
00200         msgBox.setFixedSize(500,200);
00201     }
00202 }
00203
00210 void MainWindow::openCreationWindow(){
00211     QDialog *window = new QDialog(this);
00212     connect(window, SIGNAL(settingsFilled(QVector<uint>,
CellHandler::generationTypes,uint,uint)),
this, SLOT(setCellHandler(QVector<uint>,
CellHandler::generationTypes,uint,uint)));
00213     window->show();
00214 }
00215
00223 void MainWindow::setCellHandler(const QVector<unsigned int> dimensions,
CellHandler::generationTypes type,
unsigned int stateMax, unsigned int density){
00224     CellHandler* newCellHandler = new CellHandler(dimensions, type, stateMax, density
);
00225 }
00226
00227

```

```

00228     if(m_tabs == NULL) createTabs();
00229
00230     m_cellHandlers.append(newCellHandler);
00231     std::cout << "m_cellHandlers size : " << m_cellHandlers.size() << std::endl<<std::flush;
00232     QWidget* newTab = createTab();
00233     m_tabs->addTab(newTab, "Automaton "+ QString::number(m_cellHandlers.size()));
00234     m_tabs->setCurrentWidget(newTab);
00235     updateBoard(m_cellHandlers.size()-1);
00236
00237 }
00238
00243 void MainWindow::nextState(int n){
00244     if(m_cellHandlers.size() == 0){
00245         QMessageBox msgBox;
00246         msgBox.critical(0,"Error","Please create or import an Automaton first !");
00247         msgBox.setFixedSize(500,200);
00248     }
00249     else{
00250         for(unsigned int i = 0; i < n; i++) m_cellHandlers[m_tabs->currentIndex()->
nextStates();
00251         updateBoard(m_tabs->currentIndex());
00252     }
00253 }
00254
00259 void MainWindow::updateBoard(int index){
00260     if(m_cellHandlers.size()==0){
00261         QMessageBox msgBox;
00262         msgBox.critical(0,"Error","Please create or import an Automaton first !");
00263         msgBox.setFixedSize(500,200);
00264     }
00265     else{
00266         CellHandler* cellHandler = m_cellHandlers[index];
00267         QVector<unsigned int> dimensions = cellHandler->getDimensions();
00268         QTableWidgetItem* board = getBoard(index);
00269         if(dimensions.size() > 1){
00270             int i = 0;
00271             int j = 0;
00272             for (CellHandler::const_iterator it =
CellHandler::const_iterator(cellHandler); it != cellHandler->
end() && it.changedDimension() < 2; ++it){
00274                 if(it.changedDimension() > 0){
00275                     i = 0;
00276                     j++;
00277                     std::cout << std::endl;
00278                 }
00279                 board->item(i,j)->setText(QString::number(it->getState()));
00280                 i++;
00281             }
00282         }
00283         else{
00284             int i = 0;
00285             int j = 0;
00286             for (CellHandler::const_iterator it =
CellHandler::const_iterator(cellHandler); it != cellHandler->
end() && it.changedDimension() < 1; ++it){
00287                 board->item(i,j)->setText(QString::number(it->getState()));
00288                 j++;
00289             }
00290         }
00291     }
00292 }
00293
00294 }
00295
00300 void MainWindow::forward(){
00301     nextState(m_jumpSpeed->value());
00302 }
00303
00304 QTableWidgetItem* MainWindow::getBoard(int n){
00305     return m_tabs->widget(n)->findChild<QTableWidgetItem *>();
00306 }
00307
00312 void MainWindow::createTabs(){
00313     m_tabs = new QTabWidget(this);
00314     m_tabs->setMovable(true);
00315     m_tabs->setTabsClosable(true);
00316     setCentralWidget(m_tabs);
00317     connect(m_tabs, SIGNAL(tabCloseRequested(int)), this, SLOT(closeTab(int)));
00318 }
00319
00324 void MainWindow::closeTab(int n){
00325     m_tabs->setCurrentIndex(n);
00326     saveToFile();
00327     m_tabs->removeTab(n);
00328 }

```

7.17 mainwindow.h File Reference

```
#include <QMainWindow>
#include <QtWidgets>
#include "cellhandler.h"
#include "creationdialog.h"
```

Classes

- class [MainWindow](#)
Simulation window.

7.18 mainwindow.h

```
00001 #ifndef MAINWINDOW_H
00002 #define MAINWINDOW_H
00003
00004 #include <QMainWindow>
00005 #include <QtWidgets>
00006 #include "cellhandler.h"
00007 #include "creationdialog.h"
00008
00009
00016 class MainWindow : public QMainWindow
00017 {
00018     Q_OBJECT
00019
00020     QTabWidget *m_tabs; //Tabs for the main window
00021     QVector<CellHandler *> m_cellHandlers; //QVector containing each tab's cellHandler
00022
00024     QIcon m_fastBackwardIcon;
00025     QIcon m_fastForwardIcon;
00026     QIcon m_playIcon;
00027     QIcon m_pauseIcon;
00028     QIcon m_newIcon;
00029     QIcon m_saveIcon;
00030     QIcon m_openIcon;
00031     QIcon m_resetIcon;
00032
00034     QAction *m_playPause;
00035     QAction *m_nextState;
00036     QAction *m_previousState;
00037     QAction *m_fastForward;
00038     QAction *m_fastBackward;
00039     QAction *m_openAutomaton;
00040     QAction *m_saveAutomaton;
00041     QAction *m_newAutomaton;
00042     QAction *m_resetAutomaton;
00043
00045     QToolButton *m_playPauseBt;
00046     QToolButton *m_nextStateBt;
00047     QToolButton *m_previousStateBt;
00048     QToolButton *m_fastForwardBt;
00049     QToolButton *m_fastBackwardBt;
00050     QToolButton *m_openAutomatonBt;
00051     QToolButton *m_saveAutomatonBt;
00052     QToolButton *m_newAutomatonBt;
00053     QToolButton *m_resetBt;
00054
00055
00056     QSpinBox *m_jumpSpeed;
00057     QLabel *m_speedLabel;
00058
00059     QToolBar *m_toolBar;
00060
00062     unsigned int m_boardHSize = 25;
00063     unsigned int m_boardVSize = 25;
00064     unsigned int m_cellSize = 30;
00065
00066     void createIcons();
00067     void createActions();
```



```

00068     void createToolBar();
00069     void createBoard();
00070     QWidget* createTab();
00071     void createTabs();
00072
00073
00074     void updateBoard(int index);
00075     void nextState(int n);
00076     QTableWidgetItem* getBoard(int n);
00077
00078
00079 public:
00080     explicit MainWindow(QWidget *parent = nullptr);
00081
00082
00083 signals:
00084
00085 public slots:
00086     void openFile();
00087     void saveToFile();
00088     void openCreationWindow();
00089     void setCellHandler(const QVector<unsigned int> dimensions,
00090                        CellHandler::generationTypes type =
00091                        CellHandler::generationTypes::empty,
00092                        unsigned int stateMax = 1, unsigned int density = 20);
00093     void forward();
00094     void closeTab(int n);
00095 };
00096
00097 #endif // MAINWINDOW_H

```

7.19 presentation.md File Reference

7.20 presentation.md

```

00001 \page Presentation
00002 # What is AutoCell
00003 The purpose of this project is to create a Cellular Automate Simulator.
00004
00005 \includedoc CellHandler

```

7.21 README.md File Reference

7.22 README.md

```

00001 \mainpage
00002
00003 To generate the Documentation, go in Documentation directory and run 'make'.
00004
00005 It will generate html doc (in 'output/html/index.html') and latex doc (pdf output directly in
    Documentation directory ('docPdf.pdf')).

```


Index

- ~CellHandler
 - CellHandler, 19
- addNeighbour
 - Cell, 12
- begin
 - CellHandler, 19
- Cell, 11
 - addNeighbour, 12
 - Cell, 12
 - forceState, 13
 - getNeighbour, 13
 - getNeighbours, 13
 - getRelativePosition, 13
 - getState, 14
 - m_neighbours, 15
 - m_nextState, 15
 - m_state, 15
 - setState, 14
 - validState, 15
- cell.cpp, 49
- cell.h, 50
- CellHandler, 16
 - ~CellHandler, 19
 - begin, 19
 - CellHandler, 18
 - CellHandler::iteratorT, 33
 - const_iterator, 17
 - end, 20
 - foundNeighbours, 20
 - generate, 20
 - generationTypes, 17
 - getCell, 21
 - getDimensions, 21
 - getListNeighboursPositions, 21
 - getListNeighboursPositionsRecursive, 22
 - iterator, 17
 - load, 22
 - m_cells, 26
 - m_dimensions, 26
 - nextStates, 23
 - positionIncrement, 23
 - print, 25
 - save, 25
- CellHandler::iteratorT< CellHandler_T, Cell_T >, 30
- CellHandler::iteratorT
 - CellHandler, 33
 - changedDimension, 32
 - iteratorT, 31
 - m_changedDimension, 33
 - m_finished, 33
 - m_handler, 34
 - m_position, 34
 - m_zero, 34
 - operator!=, 32
 - operator*, 32
 - operator++, 32
 - operator->, 33
- cellhandler.cpp, 50, 51
- cellhandler.h, 56
- changedDimension
 - CellHandler::iteratorT, 32
- closeTab
 - MainWindow, 37
- const_iterator
 - CellHandler, 17
- createActions
 - MainWindow, 37
- createBoard
 - MainWindow, 37
- createGenButtons
 - CreationDialog, 28
- createIcons
 - MainWindow, 38
- createTab
 - MainWindow, 38
- createTabs
 - MainWindow, 38
- createToolBar
 - MainWindow, 38
- CreationDialog, 26
 - createGenButtons, 28
 - CreationDialog, 27
 - m_densityBox, 28
 - m_dimensionsEdit, 29
 - m_doneBt, 29
 - m_empGen, 29
 - m_groupBox, 29
 - m_randGen, 29
 - m_stateMaxBox, 30
 - m_symGen, 30
 - processSettings, 28
 - settingsFilled, 28
- creationdialog.cpp, 57
- creationdialog.h, 58, 59
- end
 - CellHandler, 20

- forceState
 - Cell, [13](#)
- forward
 - MainWindow, [39](#)
- foundNeighbours
 - CellHandler, [20](#)
- generate
 - CellHandler, [20](#)
- generationTypes
 - CellHandler, [17](#)
- getBoard
 - MainWindow, [39](#)
- getCell
 - CellHandler, [21](#)
- getDimensions
 - CellHandler, [21](#)
- getListNeighboursPositions
 - CellHandler, [21](#)
- getListNeighboursPositionsRecursive
 - CellHandler, [22](#)
- getNeighbour
 - Cell, [13](#)
- getNeighbours
 - Cell, [13](#)
- getRelativePosition
 - Cell, [13](#)
- getState
 - Cell, [14](#)
- iterator
 - CellHandler, [17](#)
- iteratorT
 - CellHandler::iteratorT, [31](#)
- load
 - CellHandler, [22](#)
- m_boardHSize
 - MainWindow, [41](#)
- m_boardVSize
 - MainWindow, [41](#)
- m_cellHandlers
 - MainWindow, [41](#)
- m_cellSize
 - MainWindow, [41](#)
- m_cells
 - CellHandler, [26](#)
- m_changedDimension
 - CellHandler::iteratorT, [33](#)
- m_densityBox
 - CreationDialog, [28](#)
- m_dimensions
 - CellHandler, [26](#)
- m_dimensionsEdit
 - CreationDialog, [29](#)
- m_doneBt
 - CreationDialog, [29](#)
- m_empGen
 - CreationDialog, [29](#)
- m_fastBackward
 - MainWindow, [42](#)
- m_fastBackwardBt
 - MainWindow, [42](#)
- m_fastBackwardIcon
 - MainWindow, [42](#)
- m_fastForward
 - MainWindow, [42](#)
- m_fastForwardBt
 - MainWindow, [42](#)
- m_fastForwardIcon
 - MainWindow, [43](#)
- m_finished
 - CellHandler::iteratorT, [33](#)
- m_groupBox
 - CreationDialog, [29](#)
- m_handler
 - CellHandler::iteratorT, [34](#)
- m_jumpSpeed
 - MainWindow, [43](#)
- m_neighbours
 - Cell, [15](#)
- m_newAutomaton
 - MainWindow, [43](#)
- m_newAutomatonBt
 - MainWindow, [43](#)
- m_newIcon
 - MainWindow, [43](#)
- m_nextState
 - Cell, [15](#)
 - MainWindow, [44](#)
- m_nextStateBt
 - MainWindow, [44](#)
- m_openAutomaton
 - MainWindow, [44](#)
- m_openAutomatonBt
 - MainWindow, [44](#)
- m_openIcon
 - MainWindow, [44](#)
- m_pauseIcon
 - MainWindow, [45](#)
- m_playIcon
 - MainWindow, [45](#)
- m_playPause
 - MainWindow, [45](#)
- m_playPauseBt
 - MainWindow, [45](#)
- m_position
 - CellHandler::iteratorT, [34](#)
- m_previousState
 - MainWindow, [45](#)
- m_previousStateBt
 - MainWindow, [46](#)
- m_randGen
 - CreationDialog, [29](#)
- m_resetAutomaton
 - MainWindow, [46](#)

- m_resetBt
 - MainWindow, 46
- m_resetIcon
 - MainWindow, 46
- m_saveAutomaton
 - MainWindow, 46
- m_saveAutomatonBt
 - MainWindow, 47
- m_saveIcon
 - MainWindow, 47
- m_speedLabel
 - MainWindow, 47
- m_state
 - Cell, 15
- m_stateMaxBox
 - CreationDialog, 30
- m_symGen
 - CreationDialog, 30
- m_tabs
 - MainWindow, 47
- m_toolBar
 - MainWindow, 47
- m_zero
 - CellHandler::iteratorT, 34
- main
 - main.cpp, 59
- main.cpp, 59, 60
 - main, 59
- MainWindow, 35
 - closeTab, 37
 - createActions, 37
 - createBoard, 37
 - createIcons, 38
 - createTab, 38
 - createTabs, 38
 - createToolBar, 38
 - forward, 39
 - getBoard, 39
 - m_boardHSize, 41
 - m_boardVSize, 41
 - m_cellHandlers, 41
 - m_cellSize, 41
 - m_fastBackward, 42
 - m_fastBackwardBt, 42
 - m_fastBackwardIcon, 42
 - m_fastForward, 42
 - m_fastForwardBt, 42
 - m_fastForwardIcon, 43
 - m_jumpSpeed, 43
 - m_newAutomaton, 43
 - m_newAutomatonBt, 43
 - m_newIcon, 43
 - m_nextState, 44
 - m_nextStateBt, 44
 - m_openAutomaton, 44
 - m_openAutomatonBt, 44
 - m_openIcon, 44
 - m_pauseIcon, 45
 - m_playIcon, 45
 - m_playPause, 45
 - m_playPauseBt, 45
 - m_previousState, 45
 - m_previousStateBt, 46
 - m_resetAutomaton, 46
 - m_resetBt, 46
 - m_resetIcon, 46
 - m_saveAutomaton, 46
 - m_saveAutomatonBt, 47
 - m_saveIcon, 47
 - m_speedLabel, 47
 - m_tabs, 47
 - m_toolBar, 47
 - MainWindow, 37
 - nextState, 39
 - openCreationWindow, 39
 - openFile, 40
 - saveToFile, 40
 - setCellHandler, 40
 - updateBoard, 40
- mainwindow.cpp, 60
- mainwindow.h, 64
- nextState
 - MainWindow, 39
- nextStates
 - CellHandler, 23
- openCreationWindow
 - MainWindow, 39
- openFile
 - MainWindow, 40
- operator!=
 - CellHandler::iteratorT, 32
- operator*
 - CellHandler::iteratorT, 32
- operator++
 - CellHandler::iteratorT, 32
- operator->
 - CellHandler::iteratorT, 33
- positionIncrement
 - CellHandler, 23
- presentation.md, 65
- print
 - CellHandler, 25
- processSettings
 - CreationDialog, 28
- README.md, 65
- save
 - CellHandler, 25
- saveToFile
 - MainWindow, 40
- setCellHandler
 - MainWindow, 40
- setState

- Cell, [14](#)
- settingsFilled
 - CreationDialog, [28](#)
- updateBoard
 - MainWindow, [40](#)
- validState
 - Cell, [15](#)