

Projekt "2048"

Version 0.1

Generated by Doxygen 1.8.13

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	about Class Reference	7
4.1.1	Detailed Description	7
4.2	Board Class Reference	7
4.2.1	Detailed Description	8
4.2.2	Constructor & Destructor Documentation	8
4.2.2.1	Board() [1/2]	8
4.2.2.2	Board() [2/2]	9
4.2.3	Member Function Documentation	9
4.2.3.1	addRandomTile()	9
4.2.3.2	findFreePosition()	9
4.2.3.3	getBoardAsInt()	10
4.2.3.4	isAnotherMovePossible()	10
4.2.3.5	move()	10
4.2.3.6	moveTile()	10
4.2.3.7	setTile()	11

4.2.3.8	updateDimension()	11
4.3	CParser Class Reference	12
4.3.1	Detailed Description	12
4.3.2	Member Function Documentation	13
4.3.2.1	InitParse()	13
4.3.2.2	IP_MatchToken()	13
4.3.2.3	PushString()	13
4.3.2.4	yylex()	14
4.3.2.5	yyvsparse()	14
4.4	Game Class Reference	14
4.4.1	Detailed Description	15
4.4.2	Constructor & Destructor Documentation	15
4.4.2.1	Game()	15
4.4.3	Member Function Documentation	15
4.4.3.1	getPoints()	16
4.4.3.2	handleMove()	16
4.4.3.3	isGameWon()	16
4.4.3.4	load()	16
4.4.3.5	resetGame()	17
4.5	MainWindow Class Reference	17
4.5.1	Detailed Description	18
4.5.2	Constructor & Destructor Documentation	18
4.5.2.1	MainWindow()	18
4.5.3	Member Function Documentation	18
4.5.3.1	handleKeyPress()	18
4.5.3.2	keyPressEvent()	19
4.5.3.3	recvKeyEvent	19
4.6	newgame Class Reference	19
4.6.1	Detailed Description	20
4.6.2	Constructor & Destructor Documentation	20

4.6.2.1	newgame()	20
4.7	QBoard Class Reference	20
4.7.1	Detailed Description	21
4.7.2	Constructor & Destructor Documentation	21
4.7.2.1	QBoard()	21
4.7.3	Member Function Documentation	21
4.7.3.1	update()	21
4.8	QSolver Class Reference	22
4.8.1	Detailed Description	22
4.8.2	Constructor & Destructor Documentation	22
4.8.2.1	QSolver()	22
4.9	QTile Class Reference	23
4.9.1	Detailed Description	23
4.9.2	Constructor & Destructor Documentation	23
4.9.2.1	QTile()	23
4.10	Solver Class Reference	24
4.10.1	Detailed Description	24
4.10.2	Member Function Documentation	25
4.10.2.1	checkRowMerge()	25
4.10.2.2	compareDiagonalTiles()	25
4.10.2.3	compareNumberTiles()	25
4.10.2.4	getDirection()	26
4.10.2.5	getlastDirection()	26
4.10.2.6	mergevertical()	26
4.10.2.7	setlastDirection()	27
4.11	Tile Class Reference	27
4.11.1	Detailed Description	27
4.11.2	Constructor & Destructor Documentation	28
4.11.2.1	Tile()	28
4.12	CParser::tylval Struct Reference	28
4.13	Worker Class Reference	28
4.13.1	Detailed Description	29

5 File Documentation	31
5.1 C:/dev/git/2048/src/core/board.h File Reference	31
5.1.1 Detailed Description	31
5.2 C:/dev/git/2048/src/core/game.h File Reference	31
5.2.1 Detailed Description	32
5.3 C:/dev/git/2048/src/core/tile.h File Reference	32
5.3.1 Detailed Description	32
5.4 C:/dev/git/2048/src/definitions.h File Reference	32
5.4.1 Detailed Description	33
5.5 C:/dev/git/2048/src/gui/qboard.h File Reference	33
5.5.1 Detailed Description	33
5.6 C:/dev/git/2048/src/gui/qtile.h File Reference	33
5.6.1 Detailed Description	33
5.7 C:/dev/git/2048/src/mainwindow.h File Reference	34
5.7.1 Detailed Description	34
5.8 C:/dev/git/2048/src/qsolver.h File Reference	34
5.8.1 Detailed Description	34
5.9 C:/dev/git/2048/src/solver.h File Reference	35
5.9.1 Detailed Description	35
5.10 C:/dev/git/2048/src/worker.h File Reference	35
5.10.1 Detailed Description	35
Index	37

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Board	7
CParser	12
Game	14
QDialog	
about	7
newgame	19
QSolver	22
QLabel	
QTile	23
QMainWindow	
MainWindow	17
QThread	
Worker	28
QWidget	
QBoard	20
Solver	24
Tile	27
CParser::tyylval	28

Chapter 2

Class Index

2.1 Class List

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

about	
About window	7
Board	
Board (p. 7) Class, manages all tiles	7
CParser	
Mapping token to the corresponding index	12
Game	
Game (p. 14) class, manages the board, points, game state and loads savegames	14
MainWindow	
MainWindows class	17
newgame	
Window to start a new game	19
QBoard	
QBoard (p. 20) class, Graphical proxy class for core/board.h (p. 31)	20
QSolver	
QSolver (p. 22) class, Qt Dialog for solver.h (p. 35)	22
QTile	
Tile (p. 27) class, Graphical proxy class for core/tile.h (p. 32)	23
Solver	
Solver (p. 24) Class, contains optimized methods for solving the game	24
Tile	
Tile (p. 27) Class, represents one tile of the game	27
CParser::tyylval	28
Worker	
Worker (p. 28) class, contains algorithms for solving the game	28

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

C:/dev/git/2048/src/ about.h	??
C:/dev/git/2048/src/ definitions.h	
Global definitions used in most classes	32
C:/dev/git/2048/src/ mainwindow.h	
Main Window	34
C:/dev/git/2048/src/ newgame.h	??
C:/dev/git/2048/src/ qsolver.h	
Qt proxy class for solver	34
C:/dev/git/2048/src/ solver.h	
Class with optimized methods for solving the game. Algorithms are implemented in worker.h (p. 35)	35
C:/dev/git/2048/src/ worker.h	
Class which contains algorithms for solving the game. Is run in QThread	35
C:/dev/git/2048/src/core/ board.h	
Class represents the board of the game. Uses tiles (core/tile.h (p. 32))	31
C:/dev/git/2048/src/core/ cparser.h	??
C:/dev/git/2048/src/core/ game.h	
Main game class for the game 2048. Contains one board (core/board.h (p. 31))	31
C:/dev/git/2048/src/core/ tile.h	
Class represents one tile of the game	32
C:/dev/git/2048/src/gui/ qboard.h	
Graphical proxy class for core/board.h (p. 31)	33
C:/dev/git/2048/src/gui/ qtile.h	
Graphical proxy class for core/ctile.h	33

Chapter 4

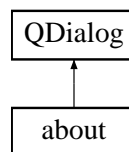
Class Documentation

4.1 about Class Reference

About window.

```
#include <about.h>
```

Inheritance diagram for about:



Public Member Functions

- **about** (QWidget *parent=0)

4.1.1 Detailed Description

About window.

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

- C:/dev/git/2048/src/about.h
- C:/dev/git/2048/src/about.cpp

4.2 Board Class Reference

Board (p. 7) Class, manages all tiles.

```
#include <board.h>
```

Public Types

- enum **Direction** { **LEFT**, **RIGHT**, **UP**, **DOWN** }

Public Member Functions

- **Board** (int dimension)
*Constructor for **Board** (p. 7).*
- **Board** (const **Board** &ref)
Copy constructor for board that copies the m_board member.
- void **updateDimension** (const unsigned int dimension)
Update dimension of the board.
- unsigned int **getDimension** (void)
- void **clear** (void)
Clear the board of all tiles.
- **Tile** * **getTile** (const unsigned int i, const unsigned int j)
- void **setTile** (unsigned int i, unsigned int j, int value)
Set value of a tile.
- T_CORD **findFreePosition** (void)
Find a coordinate in the board without any tile.
- void **addRandomTile** (void)
Add a new random tile to the board.
- bool **move** (Direction dir)
Apply a move direction to the board.
- bool **moveTile** (unsigned int i, unsigned int j)
Main method for calculation moves and collisions with other tiles.
- void **rotate** ()
Rotate the whole board 90° clockwise.
- bool **isAnotherMovePossible** (void)
Test if any tile can be moved in the board.
- std::vector< std::vector< **Tile** * > > **getBoard** (void)
- std::vector< std::vector< int > > **getBoardAsInt** (void)
Returns the board as int vector.

4.2.1 Detailed Description

Board (p. 7) Class, manages all tiles.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 **Board**() [1/2]

```
Board::Board (
    int dimension )
```

Constructor for **Board** (p. 7).

Parameters

<i>dimension</i>	Size of board (e.g. 4 for 4x4)
------------------	--------------------------------

4.2.2.2 Board() [2/2]

```
Board::Board (  
    const Board & ref )
```

Copy constructor for board that copies the m_board member.

Parameters

<i>ref</i>	
------------	--

4.2.3 Member Function Documentation

4.2.3.1 addRandomTile()

```
void Board::addRandomTile (  
    void )
```

Add a new random tile to the board.

Probability for 2: 90% Probability for 4: 10%

4.2.3.2 findFreePosition()

```
T_CORD Board::findFreePosition (  
    void )
```

Find a coordinate in the board without any tile.

Returns

coordinates (i,j)

4.2.3.3 getBoardAsInt()

```
std::vector< std::vector< int > > Board::getBoardAsInt (
    void )
```

Returns the board as int vector.

Returns

int vector

4.2.3.4 isAnotherMovePossible()

```
bool Board::isAnotherMovePossible (
    void )
```

Test if any tile can be moved in the board.

Returns

True when a tile can be moved, otherwise False

4.2.3.5 move()

```
bool Board::move (
    Direction dir )
```

Apply a move direction to the board.

Parameters

<i>dir</i>	Direction for the tiles to move
------------	---------------------------------

Returns

True when any tiles has been moved, otherwise False

4.2.3.6 moveTile()

```
bool Board::moveTile (
    unsigned int pos_i,
    unsigned int pos_j )
```

Main method for calculation moves and collisions with other tiles.

Parameters

<i>pos</i> ↔ <i>_i</i>	coordinate i
<i>pos</i> ↔ <i>_j</i>	coordinate j

Returns

True, when a move was done, otherwise False

In this method a tile with given coordinates (i,j) starts to "fall" to the ground. Any collisions (and merges) are handled here.

4.2.3.7 setTile()

```
void Board::setTile (
    unsigned int i,
    unsigned int j,
    int value )
```

Set value of a tile.

Parameters

<i>i</i>	coordinate i
<i>j</i>	coordinate j
<i>value</i>	Tile (p. 27) value

4.2.3.8 updateDimension()

```
void Board::updateDimension (
    const unsigned int dimension )
```

Update dimension of the board.

Parameters

<i>dimension</i>	Dimension of board (e.g. 4 for 4x4)
------------------	-------------------------------------

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

- C:/dev/git/2048/src/core/ **board.h**
- C:/dev/git/2048/src/core/board.cpp

4.3 CParser Class Reference

Mapping token to the corresponding index.

```
#include <cparser.h>
```

Classes

- struct **tyylval**

Public Member Functions

- int **yylex** ()
Copy from script.
- void **yyerror** (char *ers)
- int **IP_MatchToken** (string &tok)
Return the corresponding index for the token.
- void **InitParse** (FILE *inp, FILE *err, FILE *lst)
Init parser.
- int **yyparse** ()
CParser::pr_tokenable.
- void **IP_init_token_table** ()
Init token table.
- void **Load_tokenentry** (string str, int index)
- void **PushString** (char c)
Adds a character to the string value.

Public Attributes

- string **yytext**
- struct **CParser::tyylval** **yyval**
- FILE * **IP_Input**
- FILE * **IP_Error**
- FILE * **IP_List**
- int **IP_LineNumber**
- int **ugetflag**
- int **prflag**
- map< string, int > **IP_Token_table**
- map< int, string > **IP_revToken_table**

4.3.1 Detailed Description

Mapping token to the corresponding index.

Parameters

<i>str</i>	
<i>index</i>	

4.3.2 Member Function Documentation

4.3.2.1 InitParse()

```
void CParser::InitParse (
    FILE * inp,
    FILE * err,
    FILE * lst )
```

Init parser.

Parameters

<i>inp</i>	File input
<i>err</i>	Error output
<i>lst</i>	output

4.3.2.2 IP_MatchToken()

```
int CParser::IP_MatchToken (
    string & tok )
```

Return the corresponding index for the token.

Parameters

<i>tok</i>	
------------	--

Returns

4.3.2.3 PushString()

```
void CParser::PushString (
    char c )
```

Adds a character to the string value.

Parameters

<i>c</i>	
----------	--

4.3.2.4 yylex()

```
int CParser::yylex ( )
```

Copy from script.

Returns

4.3.2.5 yyparse()

```
int CParser::yyparse ( )
```

CParser::pr_tokentable.

Return the class of word and print the result

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

- C:/dev/git/2048/src/core/cparser.h
- C:/dev/git/2048/src/core/cparser.cpp

4.4 Game Class Reference

Game (p. 14) class, manages the board, points, game state and loads savegames.

```
#include <game.h>
```

Public Types

- enum **State** { **GAME_RUNNING**, **GAME_WON**, **GAME_LOST** }
- enum **lexstate** {
 L_START, **L_INT**, **L_IDENT**, **L_OPERATOR**,
 L_VARIABLE, **L_ADRESS**, **L_FILEEND** }

Public Member Functions

- **Game** (int dimension)
Chhbonstructor for the game class.
- bool **handleMove** (Board::Direction direction)
Checks if a move is possible and applys the direction. Also adds a new random tile.
- unsigned int **getPoints** (void)
Calculates the points for the game.
- void **resetGame** (void)
Clears the board, reset the game state and adds random tiles.
- void **resetGame** (int dimension)
*Resets the game like **resetGame(void)** (p. 15), but with dimension update.*
- **Board** * **getBoard** (void)
- State **getState** (void)
- bool **isGameWon** (void)
Checks if the game is won (one 2048 tile exists)
- void **debugPrint** (void)
Debug output of board. Not used in production.
- bool **load** (string filename, string &loadmsg)
Load gamedata from file.

4.4.1 Detailed Description

Game (p. 14) class, manages the board, points, game state and loads savegames.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Game()

```
Game::Game (
    int dimension )
```

Chhbonstructor for the game class.

Parameters

<i>dimension</i>	Dimension of the board (e.g. 4, 4x4).
------------------	---------------------------------------

4.4.3 Member Function Documentation

4.4.3.1 getPoints()

```
unsigned int Game::getPoints (
    void )
```

Calculates the points for the game.

Returns

Points

Sums the value of all tiles in the board.

4.4.3.2 handleMove()

```
bool Game::handleMove (
    Board::Direction direction )
```

Checks if a move is possible and applies the direction. Also adds a new random tile.

Parameters

<i>direction</i>	Direction to apply.
------------------	---------------------

Returns

True, when a move was done, otherwise False.

4.4.3.3 isGameWon()

```
bool Game::isGameWon (
    void )
```

Checks if the game is won (one 2048 tile exists)

Returns

True / False

4.4.3.4 load()

```
bool Game::load (
    string filename,
    string & loadmsg )
```

Load gamedata from file.

Parameters

<i>filename</i>	
<i>loadmsg</i>	Give the result of the load process (not) successful

Returns

4.4.3.5 resetGame()

```
void Game::resetGame (
    int dimension )
```

Resets the game like **resetGame(void)** (p. 15), but with dimension update.

Parameters

<i>dimension</i>	Dimension of the board (e.g. 4, 4x4)
------------------	--------------------------------------

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

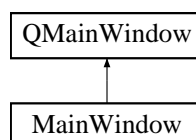
- C:/dev/git/2048/src/core/ **game.h**
- C:/dev/git/2048/src/core/game.cpp

4.5 MainWindow Class Reference

MainWindows class.

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

Inheritance diagram for MainWindow:



Public Slots

- void **recvKeyEvent** (QKeyEvent *event)
Receives QKeyEvent from external window (e.g. solver window)

Public Member Functions

- **MainWindow** (QWidget *parent=0)
Main Window of GUI.
- **~MainWindow** ()
*Destructor of **MainWindow** (p. 17) (gets called when windows is closed)*
- void **handleKeyPress** (QKeyEvent *event)
Maps keypress events to game directions, updates gui output, points and displays a win/lost dialog.
- void **updateGame** ()
*Update **Board** (p. 7), calculate points and check if game is won or lost.*

Protected Member Functions

- void **keyPressEvent** (QKeyEvent *event)
Receives QKeyEvent from keypress on keyboard.

4.5.1 Detailed Description

MainWindows class.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = 0 ) [explicit]
```

Main Window of GUI.

Parameters

<i>parent</i>	Parent Widget
---------------	---------------

4.5.3 Member Function Documentation

4.5.3.1 handleKeyPress()

```
void MainWindow::handleKeyPress (
    QKeyEvent * k )
```

Maps keypress events to game directions, updates gui output, points and displays a win/lost dialog.

Parameters

<i>k</i>	
----------	--

4.5.3.2 keyPressEvent()

```
void MainWindow::keyPressEvent (
    QKeyEvent * event ) [protected]
```

Receives QKeyEvent from keypress on keyboard.

Parameters

<i>event</i>	
--------------	--

4.5.3.3 recvKeyEvent

```
void MainWindow::recvKeyEvent (
    QKeyEvent * event ) [slot]
```

Receives QKeyEvent from external window (e.g. solver window)

Parameters

<i>event</i>	
--------------	--

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

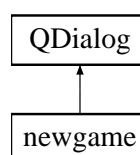
- C:/dev/git/2048/src/ **mainwindow.h**
- C:/dev/git/2048/src/mainwindow.cpp

4.6 newgame Class Reference

Window to start a new game.

```
#include <newgame.h>
```

Inheritance diagram for newgame:



Public Member Functions

- **newgame** (QWidget *parent=0)
Constructor for newgame window.
- **~newgame** ()
Destructor from the window newgame.
- void **setGame** (**Game** *game)
- void **setqboard** (**QBoard** *qboard)
- void **setLabel** (QLabel *labelpoints)
- void **start_newgame** ()
Depending on the value from the spinbox, a new game with the dimension starts or a when the value is wrong a error message appears.

4.6.1 Detailed Description

Window to start a new game.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 newgame()

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

Constructor for newgame window.

Parameters

<i>parent</i>	
---------------	--

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

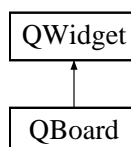
- C:/dev/git/2048/src/newgame.h
- C:/dev/git/2048/src/newgame.cpp

4.7 QBoard Class Reference

QBoard (p. 20) class, Graphical proxy class for **core/board.h** (p. 31).

```
#include <qboard.h>
```

Inheritance diagram for QBoard:



Public Member Functions

- **QBoard** (**Board** *board)
*Constructor for class **QBoard** (p. 20).*
- void **init** (void)
Update dimension and initialize `m_layout_tiles` from `m_board`.
- void **update** (void)
*Update **QBoard** (p. 20) with new QTiles from `m_board`.*

4.7.1 Detailed Description

QBoard (p. 20) class, Graphical proxy class for **core/board.h** (p. 31).

4.7.2 Constructor & Destructor Documentation

4.7.2.1 QBoard()

```
QBoard::QBoard (
    Board * board )
```

Constructor for class **QBoard** (p. 20).

Parameters

<code>board</code>	
--------------------	--

4.7.3 Member Function Documentation

4.7.3.1 update()

```
void QBoard::update (
    void )
```

Update **QBoard** (p. 20) with new QTiles from `m_board`.

Method is run everytime a move is applied.

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

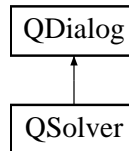
- C:/dev/git/2048/src/gui/ **qboard.h**
- C:/dev/git/2048/src/gui/qboard.cpp

4.8 QSolver Class Reference

QSolver (p. 22) class, Qt Dialog for **solver.h** (p. 35).

```
#include <qsolver.h>
```

Inheritance diagram for QSolver:



Public Slots

- void **recvKeyEvent** (QKeyEvent *event)

Public Member Functions

- **QSolver** (QWidget *parent=0)
*Constructor of **QSolver** (p. 22).*
- void **setGame** (**Game** *game)
- void **start** (void)
- void **stop** (void)

4.8.1 Detailed Description

QSolver (p. 22) class, Qt Dialog for **solver.h** (p. 35).

4.8.2 Constructor & Destructor Documentation

4.8.2.1 QSolver()

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

Constructor of **QSolver** (p. 22).

Parameters

<i>parent</i>	
---------------	--

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

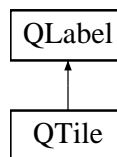
- C:/dev/git/2048/src/ **qsolver.h**
- C:/dev/git/2048/src/qsolver.cpp

4.9 QTile Class Reference

Tile (p. 27) class, Graphical proxy class for **core/tile.h** (p. 32).

```
#include <qtile.h>
```

Inheritance diagram for QTile:



Public Member Functions

- **QTile** (**Tile** *tile)
*Constructor for **QTile** (p. 23).*
- void **update** ()
Update the color and styling when the value of a tile changes.
- void **mouseDoubleClickEvent** (QMouseEvent *event)

4.9.1 Detailed Description

Tile (p. 27) class, Graphical proxy class for **core/tile.h** (p. 32).

4.9.2 Constructor & Destructor Documentation

4.9.2.1 QTile()

```
QTile::QTile (
    Tile * tile )
```

Constructor for **QTile** (p. 23).

Parameters

<i>tile</i>	Input is a Tile (p. 27) (core/tile.h (p. 32))
-------------	--

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

- C:/dev/git/2048/src/gui/ **qtile.h**
- C:/dev/git/2048/src/gui/qtile.cpp

4.10 Solver Class Reference

Solver (p. 24) Class, contains optimized methods for solving the game.

```
#include <solver.h>
```

Public Types

- enum **Direction** { **LEFT**, **RIGHT**, **UP**, **DOWN** }

Public Member Functions

- **Solver** (int lastDirection)
- Direction **getDirection** (int direction)
return depending on the last direction, the new direction
- int **getlastDirection** ()
getter function for lastDirction
- void **setlastDirection** (int a)
setter function for lastDirection
- Direction **getBestDirection** (const T_BOARD &board, unsigned int runs)
- void **addRandomTile** (T_BOARD &board)
- unsigned int **randomRun** (const T_BOARD &board, Direction direction)
- unsigned int **randomRun** (const T_BOARD &board, Direction direction, unsigned int runs)
- std::vector< T_CORD > **getFreePositions** (const T_BOARD &board)
- bool **mergevertical** (const T_BOARD &board)
checks if a vertical merge is possible
- bool **checkRowMerge** (const T_BOARD &board)
Checks if Tiles in a Row are mergeable.
- bool **isMovePossible** (const T_BOARD &board)
- bool **isMovePossible** (const T_BOARD &board, Direction direction)
- bool **isRightDownPossible** (const T_BOARD &board)
- bool **compareNumberTiles** (const T_BOARD &board, int mode)
Count the number of tiles in a row.
- bool **compareDiagonalTiles** (const T_BOARD &board)
Checks the diagonal Tiles from left to right.
- unsigned int **evaluateMove** (const T_BOARD &board, Direction direction)
- unsigned int **getPoints** (T_BOARD &board)
- unsigned int **moveBoard** (T_BOARD &board, Direction direction)
- void **printBoard** (const T_BOARD &board)

4.10.1 Detailed Description

Solver (p. 24) Class, contains optimized methods for solving the game.

4.10.2 Member Function Documentation

4.10.2.1 checkRowMerge()

```
bool Solver::checkRowMerge (
    const T_BOARD & board )
```

Checks if Tiles in a Row are mergeable.

Parameters

Board (p. 7)	from the running Game (p. 14)
---------------------	--------------------------------------

Returns

If Tiles are mergable return false, else true

4.10.2.2 compareDiagonalTiles()

```
bool Solver::compareDiagonalTiles (
    const T_BOARD & board )
```

Checks the diagonal Tiles from left to right.

Parameters

Board (p. 7)	from the running game
---------------------	-----------------------

Returns

If the diagonal Tiles have the same value return true, else return false

4.10.2.3 compareNumberTiles()

```
bool Solver::compareNumberTiles (
    const T_BOARD & board,
    int mode )
```

Count the number of tiles in a row.

Parameters

<i>board</i>	
<i>mode</i>	

Returns**4.10.2.4 getDirection()**

```
Solver::Direction Solver::getDirection (
    int lastDirection )
```

return depending on the last direction, the new direction

Parameters

<i>the</i>	last direction move
------------	---------------------

Returns

the next direction move

4.10.2.5 getlastDirection()

```
int Solver::getlastDirection ( )
```

getter function for lastDircetion

Returns

the last direction move

4.10.2.6 mergevertical()

```
bool Solver::mergevertical (
    const T_BOARD & board )
```

checks if a vertical merge is possible

Parameters

Board (p. 7)	from the running game
---------------------	-----------------------

Returns

if a merge is possible return true else false

4.10.2.7 setlastDirection()

```
void Solver::setlastDirection (
    int lastDirection )
```

setter function for lastDirection

Parameters

<i>the</i>	last Direction move
------------	---------------------

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

- C:/dev/git/2048/src/ **solver.h**
- C:/dev/git/2048/src/solver.cpp

4.11 Tile Class Reference

Tile (p. 27) Class, represents one tile of the game.

```
#include <tile.h>
```

Public Member Functions

- **Tile** (int value)
Constructor of tile class.
- void **updateValue** (const int value)
- int **getValue** (void)
- void **upgrade** (void)
- bool **getFlagUpgraded** (void)
- void **resetFlagUpgraded** (void)

4.11.1 Detailed Description

Tile (p. 27) Class, represents one tile of the game.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Tile()

```
Tile::Tile (
    int value )
```

Constructor of tile class.

Parameters

<i>value</i>	Initial value of the tile
--------------	---------------------------

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

- C:/dev/git/2048/src/core/ **tile.h**
- C:/dev/git/2048/src/core/tile.cpp

4.12 CParser::tyylval Struct Reference

Public Attributes

- string **s**
- int **i**

The documentation for this struct was generated from the following file:

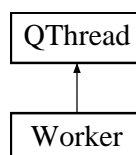
- C:/dev/git/2048/src/core/cparser.h

4.13 Worker Class Reference

Worker (p. 28) class, contains algorithms for solving the game.

```
#include <worker.h>
```

Inheritance diagram for Worker:



Signals

- void **sendKeyEvent** (QKeyEvent *event)

Public Member Functions

- void **setInterval** (unsigned int interval)
- unsigned int **getInterval** (void)
- void **setAlgorithm** (Algorithm algorithm)
- void **setGame** (**Game** *game)

Public Attributes

- bool **m_enabled**
- bool **m_single**

4.13.1 Detailed Description

Worker (p. 28) class, contains algorithms for solving the game.

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

- C:/dev/git/2048/src/ **worker.h**
- C:/dev/git/2048/src/worker.cpp

Chapter 5

File Documentation

5.1 C:/dev/git/2048/src/core/board.h File Reference

Class represents the board of the game. Uses tiles (**core/tile.h** (p. 32)).

```
#include "definitions.h"
#include "core/tile.h"
#include <vector>
#include <algorithm>
#include <iterator>
```

Classes

- class **Board**
***Board** (p. 7) Class, manages all tiles.*

5.1.1 Detailed Description

Class represents the board of the game. Uses tiles (**core/tile.h** (p. 32)).

5.2 C:/dev/git/2048/src/core/game.h File Reference

Main game class for the game 2048. Contains one board (**core/board.h** (p. 31)).

```
#include "core/board.h"
#include "core/cparser.h"
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
```

Classes

- class **Game**
Game (p. 14) class, manages the board, points, game state and loads savegames.

Macros

- #define **C_STRING1** 3
- #define **C_IDENTIFIER** 4
- #define **C_INTEGER1** 5

5.2.1 Detailed Description

Main game class for the game 2048. Contains one board (**core/board.h** (p. 31)).

5.3 C:/dev/git/2048/src/core/tile.h File Reference

Class represents one tile of the game.

Classes

- class **Tile**
Tile (p. 27) Class, represents one tile of the game.

5.3.1 Detailed Description

Class represents one tile of the game.

5.4 C:/dev/git/2048/src/definitions.h File Reference

Global definitions used in most classes.

```
#include <vector>
```

Typedefs

- typedef std::pair< unsigned int, unsigned int > **T_CORD**

Enumerations

- enum **Algorithm** { **ALGO_RANDOM**, **ALGO_RIGHT_DOWN**, **ALGO_PURE_MONTE_CARLO** }
- enum **Command** {
MOVE_UP, **MOVE_DOWN**, **MOVE_LEFT**, **MOVE_RIGHT**,
IDLE }

5.4.1 Detailed Description

Global definitions used in most classes.

5.5 C:/dev/git/2048/src/gui/qboard.h File Reference

Graphical proxy class for **core/board.h** (p. 31).

```
#include <QObject>
#include <QWidget>
#include <QLayout>
#include <QLabel>
#include <vector>
#include "core/board.h"
#include "gui/qtile.h"
```

Classes

- class **QBoard**

QBoard (p. 20) class, Graphical proxy class for **core/board.h** (p. 31).

5.5.1 Detailed Description

Graphical proxy class for **core/board.h** (p. 31).

5.6 C:/dev/git/2048/src/gui/qtile.h File Reference

Graphical proxy class for **core/qtile.h**.

```
#include <QObject>
#include <QLabel>
#include <QGraphicsDropShadowEffect>
#include "core/tile.h"
#include <iostream>
#include <QDialog>
#include <QVBoxLayout>
```

Classes

- class **QTile**

Tile (p. 27) class, Graphical proxy class for **core/tile.h** (p. 32).

5.6.1 Detailed Description

Graphical proxy class for **core/qtile.h**.

5.7 C:/dev/git/2048/src/mainwindow.h File Reference

Main Window.

```
#include <QtWidgets/QMainWindow>
#include <QKeyEvent>
#include <QMessageBox>
#include "gui/qboard.h"
#include "core/game.h"
#include "qsolver.h"
#include "newgame.h"
#include "about.h"
```

Classes

- class **MainWindow**
MainWindows class.

5.7.1 Detailed Description

Main Window.

5.8 C:/dev/git/2048/src/qsolver.h File Reference

Qt proxy class for solver.

```
#include <QDialog>
#include <QTimer>
#include <QTime>
#include <QKeyEvent>
#include <algorithm>
#include "core/game.h"
#include "solver.h"
#include "worker.h"
#include "definitions.h"
```

Classes

- class **QSolver**
***QSolver** (p. 22) class, Qt Dialog for **solver.h** (p. 35).*

5.8.1 Detailed Description

Qt proxy class for solver.

5.9 C:/dev/git/2048/src/solver.h File Reference

Class with optimized methods for solving the game. Algorithms are implemented in **worker.h** (p. 35).

```
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <random>
```

Classes

- class **Solver**
***Solver** (p. 24) Class, contains optimized methods for solving the game.*

Typedefs

- typedef std::vector< std::vector< int > > **T_BOARD**
- typedef std::pair< unsigned int, unsigned int > **T_CORD**

5.9.1 Detailed Description

Class with optimized methods for solving the game. Algorithms are implemented in **worker.h** (p. 35).

5.10 C:/dev/git/2048/src/worker.h File Reference

Class which contains algorithms for solving the game. Is run in QThread.

```
#include <QThread>
#include <QKeyEvent>
#include <QTime>
#include <iostream>
#include <ctime>
#include "core/game.h"
#include "solver.h"
#include "definitions.h"
#include <intrin.h>
```

Classes

- class **Worker**
***Worker** (p. 28) class, contains algorithms for solving the game.*

5.10.1 Detailed Description

Class which contains algorithms for solving the game. Is run in QThread.

Index

- about, 7
- addRandomTile
 - Board, 9
- Board, 7
 - addRandomTile, 9
 - Board, 8, 9
 - findFreePosition, 9
 - getBoardAsInt, 9
 - isAnotherMovePossible, 10
 - move, 10
 - moveTile, 10
 - setTile, 11
 - updateDimension, 11
- C:/dev/git/2048/src/core/board.h, 31
- C:/dev/git/2048/src/core/game.h, 31
- C:/dev/git/2048/src/core/tile.h, 32
- C:/dev/git/2048/src/definitions.h, 32
- C:/dev/git/2048/src/gui/qboard.h, 33
- C:/dev/git/2048/src/gui/qtile.h, 33
- C:/dev/git/2048/src/mainwindow.h, 34
- C:/dev/git/2048/src/qsolver.h, 34
- C:/dev/git/2048/src/solver.h, 35
- C:/dev/git/2048/src/worker.h, 35
- CParser, 12
 - IP_MatchToken, 13
 - InitParse, 13
 - PushString, 13
 - yylex, 14
 - yyvsparse, 14
- CParser::tyylval, 28
- checkRowMerge
 - Solver, 25
- compareDiagonalTiles
 - Solver, 25
- compareNumberTiles
 - Solver, 25
- findFreePosition
 - Board, 9
- Game, 14
 - Game, 15
 - getPoints, 15
 - handleMove, 16
 - isGameWon, 16
 - load, 16
 - resetGame, 17
- getBoardAsInt
 - Board, 9
- getDirection
 - Solver, 26
- getPoints
 - Game, 15
- getlastDirection
 - Solver, 26
- handleKeyPress
 - MainWindow, 18
- handleMove
 - Game, 16
- IP_MatchToken
 - CParser, 13
- InitParse
 - CParser, 13
- isAnotherMovePossible
 - Board, 10
- isGameWon
 - Game, 16
- keyPressEvent
 - MainWindow, 19
- load
 - Game, 16
- MainWindow, 17
 - handleKeyPress, 18
 - keyPressEvent, 19
 - MainWindow, 18
 - recvKeyEvent, 19
- mergevertical
 - Solver, 26
- move
 - Board, 10
- moveTile
 - Board, 10
- newgame, 19
 - newgame, 20
- PushString
 - CParser, 13
- QBoard, 20
 - QBoard, 21
 - update, 21
- QSolver, 22
 - QSolver, 22

- QTile, 23
 - QTile, 23
- recvKeyEvent
 - MainWindow, 19
- resetGame
 - Game, 17
- setTile
 - Board, 11
- setlastDirection
 - Solver, 27
- Solver, 24
 - checkRowMerge, 25
 - compareDiagonalTiles, 25
 - compareNumberTiles, 25
 - getDirection, 26
 - getlastDirection, 26
 - mergevertical, 26
 - setlastDirection, 27
- Tile, 27
 - Tile, 28
- update
 - QBoard, 21
- updateDimension
 - Board, 11
- Worker, 28
- yylex
 - CParser, 14
- yyvsparse
 - CParser, 14