Projekt "2048"

Version 0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

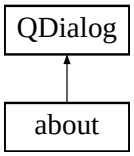
Class Documentation

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

dimension	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

ref	

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

dir	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

pos_i	coordinate i
pos_j	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	coordinate i
j	coordinate j
value	**Tile** (p. 27) value

4.2.3.8 updateDimension()

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

Update dimension of the board.

Parameters

dimension	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_tokentable.
- 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 yylval**
- 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

str	
index	

4.3.2 Member Function Documentation

4.3.2.1 InitParse()

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

Init parser.

Parameters

inp	File input
err	Error output
lst	output

4.3.2.2 IP_MatchToken()

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

Return the corresponding index for the token.

Parameters

tok	

Returns

4.3.2.3 PushString()

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

Adds a character to the string value.

Parameters

c	

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

dimension	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 applys the direction. Also adds a new random tile.

Parameters

direction	Direction to apply.

Returns

True, when a move was done, otherwhise 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

filename	
loadmsg	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

dimension	Dimension of the board (e.g. 4, 4x4)

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

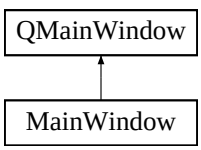
- C:/dev/git/2048/src/core/ **game.h**
- C:/dev/git/2048/src/core/game.cpp

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

parent	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

k	

4.5.3.2 keyPressEvent()

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

Receives QKeyEvent from keypress on keyboard.

Parameters

event	

4.5.3.3 recvKeyEvent

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

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

Parameters

event	

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

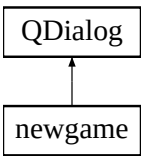
- C:/dev/git/2048/src/ **mainwindow.h**
- C:/dev/git/2048/src/mainwindow.cpp

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** ()

  Deconstructor 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

parent	

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

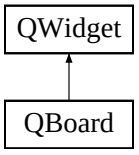
- C:/dev/git/2048/src/newgame.h
- C:/dev/git/2048/src/newgame.cpp

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

board	

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:

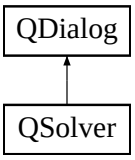
- C:/dev/git/2048/src/gui/ **qboard.h**
- C:/dev/git/2048/src/gui/qboard.cpp

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

parent	

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

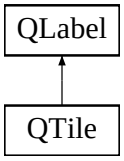
- C:/dev/git/2048/src/ **qsolver.h**
- C:/dev/git/2048/src/qsolver.cpp

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

tile	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 lastDircetion
- 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

board	
mode	

Returns

4.10.2.4 getDirection()

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

return depending on the last direction, the new direction

Parameters

the	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

the	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

value	Inital 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:

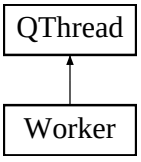
- C:/dev/git/2048/src/core/cparser.h

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