Catch Me If You Can

Generated by Doxygen 1.8.6

Fri Jan 10 2014 20:03:32

Contents

1 Main Page 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 CBoard Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 CBoard . . . 8

4.1.2.2 CBoard . . . 8

4.1.2.3 ∼CBoard . . . 8

4.1.3 Member Function Documentation . . . 8

4.1.3.1 ApplyPowerUp . . . 8

4.1.3.2 Display . . . 9

4.1.3.3 Move . . . 9

4.1.3.4 PlayRound . . . 9

4.1.3.5 SpawnItem . . . 9

4.1.3.6 UserInput . . . 9

4.1.4 Member Data Documentation . . . 10

4.1.4.1 PlayerHere . . . 10

4.2 CForm Class Reference . . . 10

4.2.1 Detailed Description . . . 11

4.2.2 Constructor & Destructor Documentation . . . 11

4.2.2.1 CForm . . . 11

4.2.2.2 ∼CForm . . . 11

4.2.3 Member Function Documentation . . . 11

4.2.3.1 Display . . . 11

4.3 CHelp Class Reference . . . 11

4.3.1 Detailed Description . . . 12
4.3.2 Constructor & Destructor Documentation . . . 12
4.3.2.1 CHelp . . . 12
4.3.2.2 ∼CHelp . . . 12
4.3.3 Member Function Documentation . . . 12
4.3.3.1 DisplayHelp . . . 12
4.3.3.2 ReadHelpFile . . . 12
4.4 CMenu Class Reference . . . 13
4.4.1 Detailed Description . . . 13
4.4.2 Constructor & Destructor Documentation . . . 13
4.4.2.1 CMenu . . . 13
4.4.2.2 ∼CMenu . . . 13
4.4.3 Member Function Documentation . . . 14
4.4.3.1 Display . . . 14
4.4.4 Member Data Documentation . . . 14
4.4.4.1 itemCount . . . 14
4.5 CSquare Class Reference . . . 14
4.5.1 Detailed Description . . . 15
4.5.2 Constructor & Destructor Documentation . . . 15
4.5.2.1 CSquare . . . 15
4.5.3 Member Function Documentation . . . 15
4.5.3.1 AddItem . . . 15
4.5.3.2 ContainsBarrier . . . 15
4.5.3.3 ContainsItem . . . 15
4.5.3.4 ContainsPlayer . . . 16
4.5.3.5 ContainsPlayer . . . 16
4.6 user_exit Struct Reference . . . 16
4.6.1 Detailed Description . . . 16

5 File Documentation **17**

5.1 src/board.cpp File Reference . . . 17
5.1.1 Detailed Description . . . 17
5.2 src/board.hpp File Reference . . . 17
5.2.1 Detailed Description . . . 18
5.3 src/common.hpp File Reference . . . 18
5.3.1 Detailed Description . . . 19
5.4 src/confmanager.cpp File Reference . . . 19
5.4.1 Detailed Description . . . 20
5.4.2 Function Documentation . . . 20
5.4.2.1 ReadBarrierApparitionFrequency . . . 20

5.4.2.2 ReadBoardDimensions . . . 20
5.4.2.3 ReadDotsForBackground . . . 21
5.4.2.4 ReadItemApparitionFrequency . . . 21
5.4.2.5 ReadItemColor . . . 21
5.4.2.6 ReadPlayerColor . . . 21
5.4.2.7 ReadStartingPlayer . . . 21
5.4.2.8 ReadVariable . . . 22
5.4.2.9 WriteCommentInFile . . . 23
5.4.2.10 WriteFloatInFile . . . 23
5.4.2.11 WriteIntInFile . . . 23
5.4.2.12 WriteVariable . . . 23
5.5 src/confmanager.hpp File Reference . . . 24
5.5.1 Detailed Description . . . 25
5.5.2 Function Documentation . . . 25
5.5.2.1 ReadBarrierApparitionFrequency . . . 25
5.5.2.2 ReadBoardDimensions . . . 25
5.5.2.3 ReadDotsForBackground . . . 25
5.5.2.4 ReadItemApparitionFrequency . . . 26
5.5.2.5 ReadItemColor . . . 26
5.5.2.6 ReadPlayerColor . . . 26
5.5.2.7 ReadStartingPlayer . . . 26
5.5.2.8 ReadVariable . . . 26
5.5.2.9 WriteCommentInFile . . . 27
5.5.2.10 WriteFloatInFile . . . 27
5.5.2.11 WriteIntInFile . . . 27
5.5.2.12 WriteVariable . . . 27
5.6 src/form.cpp File Reference . . . 28
5.6.1 Detailed Description . . . 28
5.7 src/form.hpp File Reference . . . 28
5.7.1 Detailed Description . . . 28
5.8 src/help.cpp File Reference . . . 29
5.8.1 Detailed Description . . . 29
5.9 src/help.hpp File Reference . . . 29
5.9.1 Detailed Description . . . 30
5.10 src/main.cpp File Reference . . . 30
5.10.1 Detailed Description . . . 30
5.11 src/main.hpp File Reference . . . 31
5.11.1 Detailed Description . . . 31
5.12 src/menu.cpp File Reference . . . 31
5.12.1 Detailed Description . . . 31

5.13 src/menu.hpp File Reference 32
5.13.1 Detailed Description 32
5.14 src/square.cpp File Reference 32
5.14.1 Detailed Description 33
5.15 src/square.hpp File Reference 33
5.15.1 Detailed Description 33

Index **34**

Chapter 1

Main Page

This program uses the `ncurses` library. To compile it you need install this library (the libncurses5-dev package for Debian for instance).

Chapter 2

Class Index

2.1 Class List

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

CBoard
: The board on which the game is played 7

CForm
: The form used as settings panel 10

CHelp
: The help window 11

CMenu
: A user-friendly menu 13

CSquare
: The squares of the board 14

user_exit
: The exception type thrown when the user wants to exit the game prematurely 16

Chapter 3

File Index

3.1 File List

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

src/board.cpp
    Board class implementation 17
src/board.hpp
    Board class header 17
src/common.hpp
    Common header 18
src/confmanager.cpp
    Configuration file management implementation 19
src/confmanager.hpp
    Configuration file management header 24
src/form.cpp
    Implementation of the Form class 28
src/form.hpp
    Header of the Form class 28
src/help.cpp
    Help display implementation 29
src/help.hpp
    Help class header 29
src/main.cpp
    Main file 30
src/main.hpp
    Main file header 31
src/menu.cpp
    Menu class implementation 31
src/menu.hpp
    Menu class header 32
src/square.cpp
    Square class implementation 32
src/square.hpp
    Square class header 33

Chapter 4

Class Documentation

4.1 CBoard Class Reference

The board on which the game is played.

`#include <board.hpp>`

Public Member Functions

- CBoard ()

  The default constructor.
- CBoard (Coor dimensions, Coor firstPlayer, Coor secondPlayer)

  The detailed constructor.
- ~CBoard ()

  The destructor.
- void Display (Player currentRound)

  Displays the current round of the game.
- bool PlayRound (Player currentRound, unsigned Player1WinNb, unsigned Player2WinNb)

  Plays a turn of the game.

Private Member Functions

- void Move (Coor squareToMove, Direction direction)

  Moves a square of the board.
- Direction UserInput ()

  Gets input from the player.
- void SpawnItem ()

  Spawns an item in the board.
- void ApplyPowerUp (Player player, PowerUp powerUp)

  Apply a powerUp on a player.
- void ChangeWall ()

  Delete all wall in the board and generate another board with wall.

Private Attributes

- bool AreWallsBroken

  Contain the information if wall has been broke.

- bool PlayerHere

  Contain the information if the player is in the Square.
- std::vector< std::vector < CSquare > > playfield

  Square matrix to store the playfield.
- WINDOW * window

  ncurses window to display the playfield.
- unsigned Player1Wins

  Player 1 number of wins.
- unsigned Player2Wins

  Player 2 number of wins.

4.1.1 Detailed Description

The board on which the game is played.

One Board object represents one game, it has methods to advance the state of the game or display it.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 CBoard::CBoard ( )

The default constructor.

It creates a 10x10 board with players in the upper-right-most and lower-left-most positions. It's only a delegation from the detailed constructor.

4.1.2.2 CBoard::CBoard (Coor *dimensions*, Coor *firstPlayer*, Coor *secondPlayer*)

The detailed constructor.

Parameters

dimensions	The dimensions of the desired board.
firstPlayer	The starting coordinates of the first player.
secondPlayer	The starting coordinates of the second player.

It creates a board according to the specified dimensions and will place the players if they are at a valid position. It will also initialize a ncurses window and a border accordingly.

4.1.2.3 CBoard::~CBoard ( )

The destructor.

It will get rid of the ncurses window and its border.

4.1.3 Member Function Documentation

4.1.3.1 void CBoard::ApplyPowerUp (Player *player*, PowerUp *powerUp*) [private]

Apply a powerUp on a player.

Parameters

player	the player that get the Power up
powerUp	the powerup the apply

4.1.3.2 void CBoard::Display (Player *currentRound*)

Displays the current round of the game.

Parameters

currentRound	The player that will play the next turn.

4.1.3.3 void CBoard::Move (Coor *squareToMove,* Direction *direction*) [private]

Moves a square of the board.

Parameters

squareToMove	The coordinates of the Square that will be moved.
direction	The Direction in which the Square will be moved.

No check are made in the validity of the input. The Square is swapped with the one in the specified direction, its content is not modified.

4.1.3.4 bool CBoard::PlayRound (Player *currentRound,* unsigned *Player1WinNb,* unsigned *Player2WinNb*)

Plays a turn of the game.

Parameters

currentRound	The player that will play the turn. If it is no_player, the turn is skipped.
Player1WinNb	the number of rounds won by player1.
Player2WinNb	the number of rounds won by player2.

Returns

true if a player moves over the other, false otherwise.

This procedure will get the input of the player until it is valid and will then move the player square accordingly.

4.1.3.5 void CBoard::SpawnItem ( ) [private]

Spawns an item in the board.

This method will spawn an item in a free Square of the board. The frequency of spawn is variable in the setting.

4.1.3.6 Direction CBoard::UserInput ( ) [private]

Gets input from the player.

Returns

The direction chosen by the user.

Exceptions

user_exit	if the player presses ESC, resulting in the program closing successfuly.

4.1.4 Member Data Documentation

4.1.4.1 bool CBoard::PlayerHere [private]

Contain the information if the player is in the Square.

Use only for the DoubleSpeed effect.

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

- src/board.hpp
- src/board.cpp

4.2 CForm Class Reference

The form used as settings panel.

`#include <form.hpp>`

Public Member Functions

- CForm (const std::vector< std::string > &KVariableNames, const std::vector< FIELDTYPE * > &KFieldTypes, std::vector< void * > VariablesValuesPt)

  The constructor.
- ~CForm ()

  The destructor.
- void Display ()

  Displays the setting form.

Private Attributes

- std::vector< void * > variables

  contain the values of the variables that appear in the fields
- std::vector< std::string > VarNamesVector

  contain the names of the variables that appear in the fields
- std::vector< FIELDTYPE * > RightFieldTypes

  contain the types of the variables that appear in the fields
- FIELD ** fields

  the fields of the form
- unsigned FieldCount

  the number of fields that appear in the form
- unsigned CurrentField

  the indice of the current field
- FORM * settingsform

  the form that contains all the fields
- WINDOW * formwin

  the windows that will contain the form

4.2.1 Detailed Description

The form used as settings panel.

The fom will generate a setting window that will enable the user to set the option as wanted.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CForm::CForm (const std::vector< std::string > & *KVariableNames,* const std::vector< FIELDTYPE * > & *KFieldTypes,* std::vector< void * > *VariablesValuesPt*)

The constructor.

Parameters

KVariableNames	The names of the variables that appear in the form.
KFieldTypes	The types of the variables that appear in the form.
VariablesValues-Pt	The values of the variables that appear in the form.

Will create the form windows and generate all the fields according to the variables. we give in first, second and thirs place parameters.

4.2.2.2 CForm::~CForm ( )

The destructor.

Will free the memory the fields used and remove the WINDOW.

4.2.3 Member Function Documentation

4.2.3.1 void CForm::Display ( )

Displays the setting form.

The method will diplay the setting form that the user is able to modify and will send the new values to the configuration manager.

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

- src/form.hpp
- src/form.cpp

4.3 CHelp Class Reference

The help window.

`#include <help.hpp>`

Public Member Functions

- CHelp ()
  The constructor.
- ~CHelp ()
  The destructor.
- void DisplayHelp ()
  Displays help window.

Private Member Functions

- void ReadHelpFile (const std::string &KFileName)

  Read the help file.

Private Attributes

- WINDOW * helpwin

  The WINDOW the help information are displayed into.
- std::string HelpString

  The string the information contained in the help file are stored into.

4.3.1 Detailed Description

The help window.

The help is the window that will show help to the user about how to play the game and about how the cofiguration file was made.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CHelp::CHelp ( )

The constructor.

The constructor will generate the window and read the help file to display.

4.3.2.2 CHelp::~CHelp ( )

The destructor.

It will free the memory allocated by the constructor and remove the window.

4.3.3 Member Function Documentation

4.3.3.1 void CHelp::DisplayHelp ( )

Displays help window.

The function will open a new window that will display the informations the player needs to play. It contains commands, the rules of the game and the available options in the configuration file.

4.3.3.2 void CHelp::ReadHelpFile (const std::string & *KFileName*) [private]

Read the help file.

Parameters

KFileName	The path of the help file.

The function will read the help file that have the path we give by parameters and store the file in a vector.

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

- src/help.hpp
- src/help.cpp

4.4 CMenu Class Reference

A user-friendly menu.

`#include <menu.hpp>`

Public Member Functions

- CMenu (std::vector< std::string > choices)

  The constructor.
- ~CMenu ()

  The destructor.
- std::string Display ()

  Displays the menu window and returns the choice of the user.

Private Attributes

- MENU * menu

  A pointer to the ncurses MENU used by the Menu.
- ITEM ** items

  A pointer to an array of the ITEMs used by the MENU.
- unsigned itemCount

  The number of elements of the array items points to.
- WINDOW * window

  The WINDOW the menu is displayed into.

4.4.1 Detailed Description

A user-friendly menu.

A Menu object lets the user choose between different options with arrow keys. The Menu class uses directly the MENU type from ncurses as defined in menu.h

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CMenu::CMenu (std::vector< std::string > *choices*)

The constructor.

Parameters

choices	A vector of all the different choices the user can make.

It will create the window the menu is displayed into and allocate the memory used to store the menu.

4.4.2.2 CMenu::~CMenu ( )

The destructor.

It will free the memory allocated by the constructor and remove the window.

4.4.3 Member Function Documentation

4.4.3.1 std::string CMenu::Display ( )

Displays the menu window and returns the choice of the user.

Returns

The choice of the user.

4.4.4 Member Data Documentation

4.4.4.1 unsigned CMenu::itemCount [private]

The number of elements of the array items points to.

This is necessary for proper destruction of the object.

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

- src/menu.hpp
- src/menu.cpp

4.5 CSquare Class Reference

The squares of the board.

`#include <square.hpp>`

Public Member Functions

- CSquare (Player content=no_player, bool hasBarrier=false)

  The constructor.
- bool ContainsPlayer ()

  Checks if the Square contains any player.
- bool ContainsPlayer (Player p)

  Checks if the Square contains a particular player.
- bool ContainsBarrier ()

  Checks if the Square contains a barrier.
- bool ContainsItem ()

  Checks if the Square contains an item.
- void AddItem (PowerUp itemToAdd)

  Adds an item to the Square.
- void DeleteItem ()

  Removes the item in the Square.
- PowerUp WhichPowerUp ()

  Checks which PowerUp the player have.
- void BreakWall ()

  Removes barrier from the Square.
- void AddWall ()

  Adds barrier to the Square.

Private Attributes

- Player content

  The content of the Square.
- bool hasBarrier

  Stores if the Square contains a barrier.
- PowerUp Item

  Stores the kind of item the Square contains.

4.5.1 Detailed Description

The squares of the board.

One Square object represents one square of the playfield and its content. The content of a Square cannot be modified after its creation.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CSquare::CSquare (Player *content* = `no_player`, bool *hasBarrier* = `false`)

The constructor.

Parameters

content	The content of the Square, defaults to no_player.
hasBarrier	Specifies if the Square contains a barrier, default to false.

4.5.3 Member Function Documentation

4.5.3.1 void CSquare::AddItem (PowerUp *itemToAdd*)

Adds an item to the Square.

Parameters

itemToAdd	The item to add to the Square.

4.5.3.2 bool CSquare::ContainsBarrier ( )

Checks if the Square contains a barrier.

Returns

true if the Square contains a barrier, false otherwise.

4.5.3.3 bool CSquare::ContainsItem ( )

Checks if the Square contains an item.

Returns

true if the Square contains an item, false otherwise.

4.5.3.4 bool CSquare::ContainsPlayer ( )

Checks if the Square contains any player.

Returns

false if the Square contains no_player, true otherwise.

4.5.3.5 bool CSquare::ContainsPlayer (Player *p*)

Checks if the Square contains a particular player.

Parameters

p	The player to look for.

Returns

true if the Square contains p, false otherwise.

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

- src/square.hpp
- src/square.cpp

4.6 user_exit Struct Reference

The exception type thrown when the user wants to exit the game prematurely.

`#include <common.hpp>`

4.6.1 Detailed Description

The exception type thrown when the user wants to exit the game prematurely.

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

- src/common.hpp

Chapter 5

File Documentation

5.1 src/board.cpp File Reference

Board class implementation.

```
#include "board.hpp"
```

5.1.1 Detailed Description

Board class implementation.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

board.hpp

5.2 src/board.hpp File Reference

Board class header.

```
#include "common.hpp"
#include "square.hpp"
#include "confmanager.hpp"
#include <vector>
#include <limits>
#include <exception>
#include <ncurses.h>
#include <cmath>
#include <cstdlib>
#include <chrono>
#include <thread>
```

Classes

- class CBoard

  The board on which the game is played.

5.2.1 Detailed Description

Board class header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

board.cpp

5.3 src/common.hpp File Reference

Common header.

```
#include <utility>
```

Classes

- struct user_exit

  The exception type thrown when the user wants to exit the game prematurely.

Typedefs

- typedef std::pair< unsigned, unsigned > Coor

  Coordinates in a two dimensionnal matrix.

Enumerations

- enum Player { **player1** = 1, **player2** = 2, **no_player** = 3 }

  One of the players or no_player.
- enum Direction {
  north, **north_east**, **east**, **south_east**,
  south, **south_west**, **west**, **north_west** }

One of the eight cardinal points.

- enum PowerUp {
  double_speed, **teleportation**, **break_wall**, **change_wall**,
  no_power_up }

  Different sort of power up.

5.3.1 Detailed Description

Common header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

Contains the type definitions used throughout the project.

See Also

board.cpp

5.4 src/confmanager.cpp File Reference

Configuration file management implementation.

```
#include "confmanager.hpp"
#include <iomanip>
```

Functions

- template<typename TYPE >
  TYPE ReadVariable (const std::string &KFileName, const std::string &KVariableName)

  Configuration file parser for one variable.
- template<typename TYPE >
  void WriteVariable (const std::string &KFileName, const std::string &KVariableName, const TYPE &KVariable-Value, bool EraseFile)

  Configuration file writer for one variable.
- Player ReadStartingPlayer ()

  Reads the starting Player in the config file.
- Coor ReadBoardDimensions ()

  Reads the dimensions of the board in the config file.
- int ReadPlayerColor (Player player)

  Reads the player color in the config file.
- int ReadItemColor ()

  Reads the item color in the config file.
- float ReadBarrierApparitionFrequency ()

Reads the apparition frequency of barriers.

- float ReadItemApparitionFrequency ()

  Reads the apparition frequency of items.

- bool ReadDotsForBackground ()

  Read if the background display dots or not.

- void WriteFloatInFile (const std::string &KVariableName, char *VariableValue)

  Write a float number in the configuration file.

- void WriteIntInFile (const std::string &KVariableName, char *VariableValue)

  Write an integer number in the configuration file.

- void WriteCommentInFile (const std::string &KComment, bool EraseFile)

  Write a comment in the configuration file.

5.4.1 Detailed Description

Configuration file management implementation.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

confmanager.hpp

5.4.2 Function Documentation

5.4.2.1 ReadBarrierApparitionFrequency ( )

Reads the apparition frequency of barriers.

Returns

The apparition frequency of barriers.

The returned value is between 0 and 1.

5.4.2.2 Coor ReadBoardDimensions ( )

Reads the dimensions of the board in the config file.

Returns

The dimensions of the board.

The function will return the dimensions of the board as set in the configuration file. It uses the ReadVariable function. The path of the config file is specified in a define made in confmanager.hpp

5.4.2.3 ReadDotsForBackground ( )

Read if the background display dots or not.

Returns

If the background has dots or not.

The returned value is a booleen that will set the background character to none or dots.

5.4.2.4 ReadItemApparitionFrequency ( )

Reads the apparition frequency of items.

Returns

The apparition frequency of items.

The returned value is between 0 and 1.

5.4.2.5 int ReadItemColor ( )

Reads the item color in the config file.

Returns

The Item's color equivalent as an ncurses define.

It uses the ReadVariable function to read the configuration file. The path of the config file is specified in a define made in confmanager.hpp

5.4.2.6 int ReadPlayerColor (Player *player*)

Reads the player color in the config file.

Parameters

player	Player to get the color of.

Returns

The Player's color equivalent as an ncurses define.

The function will return the color of the player whose name is the parameter. It uses the ReadVariable function to read the configuration file. The path of the config file is specified in a define made in confmanager.hpp

5.4.2.7 Player ReadStartingPlayer ( )

Reads the starting Player in the config file.

Returns

The starting Player.

The function will return which player will begin to play. It uses the ReadVariable Function. The path of the config file is specified in a define made in confmanager.hpp

5.4.2.8 template<typename TYPE > template< typename TYPE > TYPE ReadVariable (const std::string & *KFileName,* const std::string & *KVariableName*)

Configuration file parser for one variable.

Parameters

KFileName	File path of the configuration file.
KVariableName	Name of the parsed variable.

Returns

The parsed value of the variable.

The function will parse the file we give him in first place parameter. To achieve this, it will read the variable's value in the configuration file and return the value. It uses a template to be able to parse any type using operator >>

5.4.2.9 void WriteCommentInFile (const std::string & *KComment,* bool *EraseFile* = `false`)

Write a comment in the configuration file.

Parameters

KComment	The comment to inject.
EraseFile	Set if the configuration file have to be erased.

The function will write the the comment gave in parameters thanks to the WriteVariable template

5.4.2.10 void WriteFloatInFile (const std::string & *KVariableName,* char * *VariableValue*)

Write a float number in the configuration file.

Parameters

KVariableName	The name of the variable to inject.
VariableValue	The value of the variable to inject.

The function will write the name of the float variable and her value thanks to the WriteVariable template.

5.4.2.11 void WriteIntInFile (const std::string & *KVariableName,* char * *VariableValue*)

Write an integer number in the configuration file.

Parameters

KVariableName	Name of the variable to inject.
VariableValue	Value of the variable to inject.

The function will write the name of the integer variable and her value thanks to the WriteVariable template.

5.4.2.12 template<typename TYPE > template< typename TYPE > void WriteVariable (const std::string & *KFileName,* const std::string & *KVariableName,* const TYPE & *KVariableValue,* bool *EraseFile*)

Configuration file writer for one variable.

Parameters

KFileName	File path of the configuration file.
KVariableName	Name of the variable to inject.
KVariableValue	Value of the variable to inject.
EraseFile	Set if the file must be erased before being writed.

The function will write the file we give him in first place parameter and her value we give in second place parameter. To achieve this, it will write the variable and then value in the configuration file. It uses a template to be able to write any type properly using operator >>

5.5 src/confmanager.hpp File Reference

Configuration file management header.

```
#include "common.hpp"
#include <fstream>
#include <string>
#include <stdexcept>
#include <exception>
#include <vector>
#include <iostream>
#include <cctype>
```

Macros

- #define CONFIG_FILE_PATH "config.cfg"

  The (relative) path of the config file at execution.

Functions

- template<typename TYPE >
  TYPE ReadVariable (const std::string &KFileName, const std::string &KVariableName)

  Configuration file parser for one variable.
- template<typename TYPE >
  void WriteVariable (const std::string &KFileName, const std::string &KVariableName, const TYPE &KVariable-Value, bool EraseFile)

  Configuration file writer for one variable.
- Player ReadStartingPlayer ()

  Reads the starting Player in the config file.
- Coor ReadBoardDimensions ()

  Reads the dimensions of the board in the config file.
- int ReadPlayerColor (Player player)

  Reads the player color in the config file.
- int ReadItemColor ()

  Reads the item color in the config file.
- float ReadBarrierApparitionFrequency ()

  Reads the apparition frequency of barriers.
- float ReadItemApparitionFrequency ()

  Reads the apparition frequency of items.
- bool ReadDotsForBackground ()

  Read if the background display dots or not.
- void WriteFloatInFile (const std::string &KVariableName, char *VariableValue)

  Write a float number in the configuration file.
- void WriteIntInFile (const std::string &KVariableName, char *VariableValue)

  Write an integer number in the configuration file.
- void WriteCommentInFile (const std::string &KComment, bool EraseFile=false)

  Write a comment in the configuration file.

5.5.1 Detailed Description

Configuration file management header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

confmanager.cpp

5.5.2 Function Documentation

5.5.2.1 float ReadBarrierApparitionFrequency ( )

Reads the apparition frequency of barriers.

Returns

The apparition frequency of barriers.

The returned value is between 0 and 1.

5.5.2.2 Coor ReadBoardDimensions ( )

Reads the dimensions of the board in the config file.

Returns

The dimensions of the board.

The function will return the dimensions of the board as set in the configuration file. It uses the ReadVariable function. The path of the config file is specified in a define made in confmanager.hpp

5.5.2.3 bool ReadDotsForBackground ( )

Read if the background display dots or not.

Returns

If the background has dots or not.

The returned value is a booleen that will set the background character to none or dots.

5.5.2.4 float ReadItemApparitionFrequency ( )

Reads the apparition frequency of items.

Returns

The apparition frequency of items.

The returned value is between 0 and 1.

5.5.2.5 int ReadItemColor ( )

Reads the item color in the config file.

Returns

The Item's color equivalent as an ncurses define.

It uses the ReadVariable function to read the configuration file. The path of the config file is specified in a define made in confmanager.hpp

5.5.2.6 int ReadPlayerColor (Player *player*)

Reads the player color in the config file.

Parameters

player	Player to get the color of.

Returns

The Player's color equivalent as an ncurses define.

The function will return the color of the player whose name is the parameter. It uses the ReadVariable function to read the configuration file. The path of the config file is specified in a define made in confmanager.hpp

5.5.2.7 Player ReadStartingPlayer ( )

Reads the starting Player in the config file.

Returns

The starting Player.

The function will return which player will begin to play. It uses the ReadVariable Function. The path of the config file is specified in a define made in confmanager.hpp

5.5.2.8 template<typename TYPE > TYPE ReadVariable (const std::string & *KFileName,* const std::string & *KVariableName*)

Configuration file parser for one variable.

Parameters

KFileName	File path of the configuration file.
KVariableName	Name of the parsed variable.

Returns

The parsed value of the variable.

The function will parse the file we give him in first place parameter. To achieve this, it will read the variable's value in the configuration file and return the value. It uses a template to be able to parse any type using operator >>

5.5.2.9 void WriteCommentInFile (const std::string & *KComment,* bool *EraseFile*)

Write a comment in the configuration file.

Parameters

KComment	The comment to inject.
EraseFile	Set if the configuration file have to be erased.

The function will write the the comment gave in parameters thanks to the WriteVariable template

5.5.2.10 void WriteFloatInFile (const std::string & *KVariableName,* char * *VariableValue*)

Write a float number in the configuration file.

Parameters

KVariableName	The name of the variable to inject.
VariableValue	The value of the variable to inject.

The function will write the name of the float variable and her value thanks to the WriteVariable template.

5.5.2.11 void WriteIntInFile (const std::string & *KVariableName,* char * *VariableValue*)

Write an integer number in the configuration file.

Parameters

KVariableName	Name of the variable to inject.
VariableValue	Value of the variable to inject.

The function will write the name of the integer variable and her value thanks to the WriteVariable template.

5.5.2.12 template<typename TYPE > void WriteVariable (const std::string & *KFileName,* const std::string & *KVariableName,* const TYPE & *KVariableValue,* bool *EraseFile*)

Configuration file writer for one variable.

Parameters

KFileName	File path of the configuration file.
KVariableName	Name of the variable to inject.
KVariableValue	Value of the variable to inject.
EraseFile	Set if the file must be erased before being writed.

The function will write the file we give him in first place parameter and her value we give in second place parameter. To achieve this, it will write the variable and then value in the configuration file. It uses a template to be able to write any type properly using operator >>

5.6 src/form.cpp File Reference

Implementation of the Form class.

```
#include "form.hpp"
```

5.6.1 Detailed Description

Implementation of the Form class.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

form.hpp

5.7 src/form.hpp File Reference

Header of the Form class.

```
#include "confmanager.hpp"
#include <vector>
#include <form.h>
#include <fstream>
#include <stdexcept>
#include <exception>
#include <ncurses.h>
```

Classes

- class CForm

  The form used as settings panel.

5.7.1 Detailed Description

Header of the Form class.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

form.cpp

5.8 src/help.cpp File Reference

Help display implementation.

```
#include "help.hpp"
```

5.8.1 Detailed Description

Help display implementation.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

help.hpp

5.9 src/help.hpp File Reference

Help class header.

```
#include "common.hpp"
#include <vector>
#include <fstream>
#include <stdexcept>
#include <exception>
#include <ncurses.h>
#include <iostream>
```

Classes

- class CHelp

  The help window.

Macros

- #define HELP_FILE_PATH "help.txt"
  The (relative) path of the help file at execution.

5.9.1 Detailed Description

Help class header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

help.cpp

5.10 src/main.cpp File Reference

Main file.

```
#include "main.hpp"
```

Functions

- int main ()
  Main function.

5.10.1 Detailed Description

Main file.

Authors

: J.Saffi, Y.Vidal, Y.Roux, A.Torres Aurora

Date

: 08/01/14

Version

: 1.0

See Also

main.hpp

5.11 src/main.hpp File Reference

Main file header.

```
#include "common.hpp"
#include "board.hpp"
#include "confmanager.hpp"
#include "menu.hpp"
#include "form.hpp"
#include "help.hpp"
#include <cstdlib>
#include <ncurses.h>
#include <iostream>
#include <chrono>
#include <thread>
```

5.11.1 Detailed Description

Main file header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

main.cpp

5.12 src/menu.cpp File Reference

Menu class implementation.

```
#include "menu.hpp"
```

5.12.1 Detailed Description

Menu class implementation.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

menu.hpp

5.13 src/menu.hpp File Reference

Menu class header.

```
#include "common.hpp"
#include "confmanager.hpp"
#include <ncurses.h>
#include <menu.h>
#include <exception>
#include <vector>
#include <string>
```

Classes

- class CMenu

  A user-friendly menu.

5.13.1 Detailed Description

Menu class header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

menu.cpp

5.14 src/square.cpp File Reference

Square class implementation.

```
#include "square.hpp"
```

5.14.1 Detailed Description

Square class implementation.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

square.hpp

5.15 src/square.hpp File Reference

Square class header.

```
#include "common.hpp"
#include <exception>
#include <stdexcept>
```

Classes

- class CSquare

  The squares of the board.

5.15.1 Detailed Description

Square class header.

Authors

: J. Saffi, Y. Vidal, Y. Roux, A. Torres Aurora Dugo

Date

: 08/01/14

Version

: 1.0

See Also

square.cpp

Index

~CBoard
    CBoard, 8
~CForm
    CForm, 11
~CHelp
    CHelp, 12
~CMenu
    CMenu, 13

AddItem
    CSquare, 15
ApplyPowerUp
    CBoard, 8

CBoard, 7
    ~CBoard, 8
    ApplyPowerUp, 8
    CBoard, 8
    CBoard, 8
    Display, 9
    Move, 9
    PlayRound, 9
    PlayerHere, 10
    SpawnItem, 9
    UserInput, 9
CForm, 10
    ~CForm, 11
    CForm, 11
    CForm, 11
    Display, 11
CHelp, 11
    ~CHelp, 12
    CHelp, 12
    CHelp, 12
    DisplayHelp, 12
    ReadHelpFile, 12
CMenu, 13
    ~CMenu, 13
    CMenu, 13
    CMenu, 13
    Display, 14
    itemCount, 14
CSquare, 14
    AddItem, 15
    CSquare, 15
    ContainsBarrier, 15
    ContainsItem, 15
    ContainsPlayer, 15, 16
    CSquare, 15
confmanager.cpp
    ReadBarrierApparitionFrequency, 20
    ReadBoardDimensions, 20
    ReadDotsForBackground, 20
    ReadItemApparitionFrequency, 21
    ReadItemColor, 21
    ReadPlayerColor, 21
    ReadStartingPlayer, 21
    ReadVariable, 21
    WriteCommentInFile, 23
    WriteFloatInFile, 23
    WriteIntInFile, 23
    WriteVariable, 23
confmanager.hpp
    ReadBarrierApparitionFrequency, 25
    ReadBoardDimensions, 25
    ReadDotsForBackground, 25
    ReadItemApparitionFrequency, 25
    ReadItemColor, 26
    ReadPlayerColor, 26
    ReadStartingPlayer, 26
    ReadVariable, 26
    WriteCommentInFile, 27
    WriteFloatInFile, 27
    WriteIntInFile, 27
    WriteVariable, 27
ContainsBarrier
    CSquare, 15
ContainsItem
    CSquare, 15
ContainsPlayer
    CSquare, 15, 16

Display
    CBoard, 9
    CForm, 11
    CMenu, 14
DisplayHelp
    CHelp, 12

itemCount
    CMenu, 14

Move
    CBoard, 9

PlayRound
    CBoard, 9
PlayerHere
    CBoard, 10

ReadBarrierApparitionFrequency

confmanager.cpp, 20
confmanager.hpp, 25
ReadBoardDimensions
confmanager.cpp, 20
confmanager.hpp, 25
ReadDotsForBackground
confmanager.cpp, 20
confmanager.hpp, 25
ReadHelpFile
CHelp, 12
ReadItemApparitionFrequency
confmanager.cpp, 21
confmanager.hpp, 25
ReadItemColor
confmanager.cpp, 21
confmanager.hpp, 26
ReadPlayerColor
confmanager.cpp, 21
confmanager.hpp, 26
ReadStartingPlayer
confmanager.cpp, 21
confmanager.hpp, 26
ReadVariable
confmanager.cpp, 21
confmanager.hpp, 26

SpawnItem
CBoard, 9
src/board.cpp, 17
src/board.hpp, 17
src/common.hpp, 18
src/confmanager.cpp, 19
src/confmanager.hpp, 24
src/form.cpp, 28
src/form.hpp, 28
src/help.cpp, 29
src/help.hpp, 29
src/main.cpp, 30
src/main.hpp, 31
src/menu.cpp, 31
src/menu.hpp, 32
src/square.cpp, 32
src/square.hpp, 33

user_exit, 16
UserInput
CBoard, 9

WriteCommentInFile
confmanager.cpp, 23
confmanager.hpp, 27
WriteFloatInFile
confmanager.cpp, 23
confmanager.hpp, 27
WriteIntInFile
confmanager.cpp, 23
confmanager.hpp, 27
WriteVariable
confmanager.cpp, 23
confmanager.hpp, 27