Tetris

Remise 2

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

Class Documentation

3.1 mikoli::Block Class Reference

The **Block** (p. 5) class.

`#include <block.h>`

Public Member Functions

- **Block** ()

  Block *(*p. *5)'s Constructor without parameters. This constructor will use the default constructor of* ***Position*** *(*p. *26) for _position and set the color to red.*
- **Block** (int x, int y, Color color)

  Block *(*p. *5)'s Constructor with parameters.*
- ∼**Block** ()

  Block *(*p. *5)'s destructor.*
- **Position getPosition** () const

  getPosition
- QColor **getColor** ()

  getColor
- void **setPosition** (int x, int y)

  setPosition
- void **move** (Direction direction)

3.1.1 Detailed Description

The **Block** (p. 5) class.

This class will be used for building blocks that are part of a figure. A standard figure is composed with 4 blocks. _position the position of the block in the board. _color the color of the block.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Block()

```
mikoli::Block::Block (
            int x,
            int y,
            Color color )
```

Block (p. 5)'s Constructor with parameters.

Parameters

x	The value for the horizontal axis of the block.
y	The value for the vertical axis of the block.
color	The color of the block.

3.1.3 Member Function Documentation

3.1.3.1 getColor()

```
QColor mikoli::Block::getColor ( )
```

getColor

Returns

The Qcolor of the block.

3.1.3.2 getPosition()

```
Position mikoli::Block::getPosition ( ) const
```

getPosition

Returns

return The position of the block.

3.1.3.3 setPosition()

```
void mikoli::Block::setPosition (
            int x,
            int y )
```

setPosition

Parameters

x	The new value for the horizontal axis.
y	The new value for the ordinate axis.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/block.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/block.cpp

3.2 mikoli::Board Class Reference

The **Board** (p. 7) class.

`#include <board.h>`

Public Member Functions

- **Board** ()

  Board *(p. 7)'s constructor without parameters. This constructor uses the constructor with parameters by setting the height to 20 and the width to 10.*
- **Board** (int height, int width)

  Board *(p. 7)'s constructor with parameters.*
- ∼**Board** ()

  ∼Board's inline destructor.
- std::list< **Block** > **getBlocks** () const

  getBlocks
- void **addFigure** (**Figure** &cF)

  addFigure
- bool **canGoLower** (**Figure** &cF)

  canGoLower
- int **validationHeight** (int nb)

  validationHeight This method check if the number received is >= 10.
- int **validationWidth** (int nb)

  validationWidth This method check if the number received is >= 5.
- void **move** (**Figure** &cF, Direction direction)

  move This method uses ***canMove()*** *(p. 10) to check if it's possible to move the figure in this direction, and if so, will actually move the figure by calling her method* ***move()*** *(p. 12).*
- void **rotate** (**Figure** &cF, Direction direction)

  rotate
- bool **canMove** (**Figure** cF, Direction direction)

  canMove This method check if the current figure can move in the direction wished. To do this, she calculates the new positions of the current figure and then check if theses positions are available in the board. So, she compares theses positions with the positions of the blocks in _listBlocks.
- bool **canRotate** (**Figure** cF, Direction direction)

  canRotate This method check if the current figure can rotate in the direction wished. To do this, she calculates the new positions of the current figure and then check if theses positions are available in the board. So, she compares theses positions with the positions of the blocks in _listBlocks.
- bool **canPut** (**Figure** cF)

canPut This method check if the position of the _entryPoint is available. To do this, she calculates the positions the current figure need and check if theses positions are available at the _entryPoint.

- bool **areBlocksAvailable** (**Figure** cF)

  areBlocksAvailable This method check is the new positions of the current figure are available in the board.
- int **checkLines** ()

  checkLines This method check if there are lines in the board to delete. It checks each lines of the board and for each x, check in the _listBlocks if a block has this position.It calculates the number of blocks for each line and if there are as much blocks as the width of the board, it adds the y in a List to send to removeLines().
- void **removeLine** (int line)

  removeLines This method receives a list of all the y in the board to delete and then, will just delete them.
- void **reorganize** (int line)

  reorganize This method will get down all the blocks in the middle of the board after a suppression of lines. She's called after isFragmented() in the case she returns true.
- **Position entryPoint** ()

  entryPoint This method calculate the position of the _entryPoint. To do this, it calculates it from the width and the height of the board.
- int **fall** (**Figure** &cF)

  fall This method will get the current figure straight to the bottom of the board. It is called after the player pressed a certain button.
- void **setBlockDown** (**Block** &bl)

  setBlockDown
- void **reset** ()

  reset This method cleans the board's list of blocks.
- std::pair< int, int > **getBoardSize** ()

  getBoardSize

3.2.1 Detailed Description

The **Board** (p. 7) class.

This class is used to represent a fictive board in which the user will play. Here will stand the blocks and the current figure of the player. A "fictive board" because it's not really a board. It is a list of **Block** (p. 5). This list contains only the blocks placed in the graphical board. There are methods that allow to handle the moves of the current figure, to rotate the current figure and the suppression of lines, the reorganization of the blocks in the "board".

_listBlocks The list of the blocks inside the board. _width The width of the board. _height The height of the board. _entryPoint The position where the current figure appears when she arrives in the board.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Board()

```
mikoli::Board::Board (
            int height,
            int width )
```

Board (p. 7)'s constructor with parameters.

Parameters

height	The height of the board.
width	The width of the board.

3.2.3 Member Function Documentation

3.2.3.1 addFigure()

```
void mikoli::Board::addFigure (
            Figure & cF )
```

addFigure

Parameters

cF	The figure to add in the board. Add the current figure to the board's list of blocks.

3.2.3.2 areBlocksAvailable()

```
bool mikoli::Board::areBlocksAvailable (
            Figure cF )
```

areBlocksAvailable This method check is the new positions of the current figure are available in the board.

Parameters

cF	The current figure

Returns

true if the positions are available, false otherwise.

3.2.3.3 canGoLower()

```
bool mikoli::Board::canGoLower (
            Figure & cF )
```

canGoLower

Parameters

cF	The current figure to test.

Returns

True if the figure can go once down, false otherwise.

3.2.3.4 canMove()

```
bool mikoli::Board::canMove (
              Figure cF,
             Direction direction )
```

canMove This method check if the current figure can move in the direction wished. To do this, she calculates the new positions of the current figure and then check if theses positions are available in the board. So, she compares theses positions with the positions of the blocks in _listBlocks.

Parameters

cF	The current figure.
direction	The direction we want to move.

Returns

true if the positions are available, false otherwise.

3.2.3.5 canPut()

```
bool mikoli::Board::canPut (
              Figure cF )
```

canPut This method check if the position of the _entryPoint is available. To do this, she calculates the positions the current figure need and check if theses positions are available at the _entryPoint.

Parameters

cF	The figure we want to put in the board.

Returns

true if the positions are available, false otherwise.

3.2.3.6 canRotate()

```
bool mikoli::Board::canRotate (
              Figure cF,
             Direction direction )
```

canRotate This method check if the current figure can rotate in the direction wished. To do this, she calculates the new positions of the current figure and then check if theses positions are available in the board. So, she compares theses positions with the positions of the blocks in _listBlocks.

Parameters

cF	The current figure.
direction	The direction we want to move.

Returns

true if the positions are available, false otherwise.

3.2.3.7 checkLines()

```
int mikoli::Board::checkLines ( )
```

checkLines This method check if there are lines in the board to delete. It checks each lines of the board and for each x, check in the _listBlocks if a block has this position.It calculates the number of blocks for each line and if there are as much blocks as the width of the board, it adds the y in a List to send to removeLines().

Returns

The list with all the lines to delete. If there isn't any lines to delete, this list is empty.

3.2.3.8 entryPoint()

```
Position mikoli::Board::entryPoint ( )
```

entryPoint This method calculate the position of the _entryPoint. To do this, it calculates it from the width and the height of the board.

Returns

The position of the _entryPoint.

3.2.3.9 getBlocks()

```
std::list< Block > mikoli::Board::getBlocks ( ) const
```

getBlocks

Returns

The list of the blocks in the board.

3.2.3.10 getBoardSize()

```
std::pair< int, int > mikoli::Board::getBoardSize ( )
```

getBoardSize

Returns

A pair with two integers, the width and height of the board.

3.2.3.11 move()

```
void mikoli::Board::move (
              Figure & cF,
             Direction direction )
```

move This method uses **canMove()** (p. 10) to check if it's possible to move the figure in this direction, and if so, will actually move the figure by calling her method **move()** (p. 12).

If itsn't possible to move the figure, the method does nothing.

Parameters

cF	The current figure.
direction	The direction into move the figure.

3.2.3.12 removeLine()

```
void mikoli::Board::removeLine (
             int line )
```

removeLines This method receives a list of all the y in the board to delete and then, will just delete them.

Parameters

line	The line where all the blocks will be removed from the board's list.

3.2.3.13 rotate()

```
void mikoli::Board::rotate (
              Figure & cF,
             Direction direction )
```

rotate

Parameters

cF	The current figure to rotate
direction	The direction we want to rotate

3.2.3.14 setBlockDown()

```
void mikoli::Board::setBlockDown (
              Block & bl )
```

setBlockDown

Parameters

bl	The block to move This method is used in **reorganize()** (p. 8). It changes the position of the block to y - 1.

3.2.3.15 validationHeight()

```
int mikoli::Board::validationHeight (
             int nb )
```

validationHeight This method check if the number received is >= 10.

Parameters

nb	The number to validate.

Returns

The number if he's >= 10, an exception otherwise.

Exceptions

TetrisException *(p. 33)*	if nb < 10.

3.2.3.16 validationWidth()

```
int mikoli::Board::validationWidth (
            int nb )
```

validationWidth This method check if the number received is >= 5.

Parameters

nb	The number to validate.

Returns

The number if he's >= 5, an exception otherwise.

Exceptions

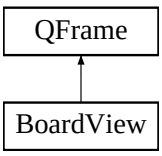
TetrisException *(p. 33)*	if nb < 5.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/board.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/board.cpp

3.3 BoardView Class Reference

Inheritance diagram for BoardView:



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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/boardview.h

3.4 mikoli::Buttons Class Reference

The **Buttons** (p. 15) class.

`#include <buttons.h>`

Public Member Functions

- **Buttons** ()

  Buttons *(p. 15)'s constructor without parameters. This constructor uses the constructor with parameters by setting the height to 20 and the width to 10.*
- **Buttons** (QWidget &fenetre)

  Buttons *(p. 15)'s constructor with QWidget as parameter. This constructor is used to create a QPushButton for the "Quick Game".*
- **Buttons** (QWidget &fenetre, int x, int y, int width, int height, const QString title)

  Buttons *(p. 15)'s constructor with parameters. This constructor is used to create a* ***Buttons*** *(p. 15) configured.*
- void **setVisibility** (bool visibility)

  To set the visibility of the QPushButton.
- QPushButton * **getButton** ()

  To get the QPushButton.

3.4.1 Detailed Description

The **Buttons** (p. 15) class.

This class construct an Object that contains a QPushButtons configured

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Buttons() [1/2]

```
mikoli::Buttons::Buttons (
            QWidget & fenetre )
```

Buttons (p. 15)'s constructor with QWidget as parameter. This constructor is used to create a QPushButton for the "Quick Game".

Parameters

fenetre	the **Widget** (p. 46) in which the Button has to appear

3.4.2.2 Buttons() [2/2]

```
mikoli::Buttons::Buttons (
            QWidget & fenetre,
            int x,
            int y,
            int width,
            int height,
            const QString title )
```

Buttons (p. 15)'s constructor with parameters. This constructor is used to create a **Buttons** (p. 15) configured.

Parameters

fenetre	the **Widget** (p. 46) in which the Button has to appear
x	the x coordonate of the QPushButton
y	the y coordonate of the QPushButton
width	the QPushButton's with
height	the QPushButton's height
title	the QPushButton's label

3.4.3 Member Function Documentation

3.4.3.1 setVisibility()

```
void mikoli::Buttons::setVisibility (
            bool visibility )
```

To set the visibility of the QPushButton.

Parameters

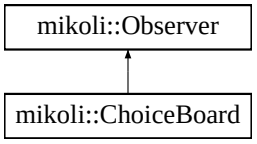
visibility	QPushButton visible or not

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/buttons.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/buttons.cpp

3.5 mikoli::ChoiceBoard Class Reference

Inheritance diagram for mikoli::ChoiceBoard:



Public Member Functions

- **ChoiceBoard** ()

  *The constructor of **ChoiceBoard** (p. 16) without parameters.*
- **ChoiceBoard** (QWidget &fenetre, **TetrisGame** &game)

  *The constructor of **ChoiceBoard** (p. 16) with parameters.*
- Gamemode **getMode** ()

  To get the GameMode.
- QPushButton * **getButtonRestart** ()

  To get the restart Button.
- QPushButton * **getButtonNewGame** ()

  To get the NewGame Button.
- QPushButton * **getButtonQuickGame** ()

  To get the QuickGame Button.
- int **getModeValue** (Gamemode mode)

  To get the Game mode value (which is the goal)
- void **hide** ()

  *To hide the **ChoiceBoard** (p. 16).*
- void **show** ()

  *To show the **ChoiceBoard** (p. 16) with the **Buttons** (p. 15).*
- void **showChooseMenu** ()

  *To show the **ChoiceBoard** (p. 16) (only the radiobuttons and spin boxes)*
- void **Update** ()

  The method executed when the observable changed.

Additional Inherited Members

3.5.1 Constructor & Destructor Documentation

3.5.1.1 ChoiceBoard()

```
mikoli::ChoiceBoard::ChoiceBoard (
            QWidget & fenetre,
             TetrisGame & game )
```

The constructor of **ChoiceBoard** (p. 16) with parameters.

Parameters

fenetre	the **Widget** (p. 46) in which the **ChoiceBoard** (p. 16) has to appear
game	the **TetrisGame** (p. 33)

3.5.2 Member Function Documentation

3.5.2.1 getModeValue()

```
int mikoli::ChoiceBoard::getModeValue (
            Gamemode mode )
```

To get the Game mode value (which is the goal)

Parameters

mode	the Gamemode

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/choiceboard.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/choiceboard.cpp

3.6 Controller Class Reference

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/controller.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/controller.cpp

3.7 mikoli::Figure Class Reference

The **Figure** (p. 18) class.

```
#include <figure.h>
```

Public Member Functions

- **Figure** ()

  *Figures constructor without parameters. This constructor will use the constructor without parameters og **Position** (p. 26) for initializing the attribute _position and settings the _color attribute to red.*
- **Figure** (TypeShape typeShape)

  Figures Constructor with parameters.
- ∼**Figure** ()

  Figures destructor.
- std::vector< **Position** > **getPositions** ()

  getPositions
- std::vector< **Block** > **getBlocks** ()

getBlocks

- TypeShape **getTypeFigure** ()

  getTypeFigure
- void **setBlocks** (std::vector< **Block** >)
- void **move** (Direction direction)

  move Makes it possible to move a figure to the left, right or down.
- void **rotate** (Direction direction)

  rotate Makes it possible to rotate a figure to the left or to the right.
- void **newPosition** (**Position** position)

  newPosition Makes it possible to displace a figure by modifying the location of its central point.

3.7.1 Detailed Description

The **Figure** (p. 18) class.

This class is used to construct a figure. A figure is composed with 4 blocks and a center point around which the figure can rotate. _Blocks An array of 4 blocks . _axePoint The point around which the figure can rotate also used to put a figure in the board at a certain position.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Figure()

```
mikoli::Figure::Figure (
            TypeShape typeShape )
```

Figures Constructor with parameters.

Parameters

typeShape	The type of the **Figure** (p. 18).

3.7.3 Member Function Documentation

3.7.3.1 getBlocks()

```
std::vector< Block > mikoli::Figure::getBlocks ( )
```

getBlocks

Returns

return a list of Blocks which forms the figure.

3.7.3.2 getPositions()

```
std::vector< Position > mikoli::Figure::getPositions ( )
```

getPositions

Returns

return The list of positions which forms the figure.

3.7.3.3 getTypeFigure()

```
TypeShape mikoli::Figure::getTypeFigure ( )
```

getTypeFigure

Returns

The type of the figure

3.7.3.4 move()

```
void mikoli::Figure::move (
            Direction direction )
```

move Makes it possible to move a figure to the left, right or down.

Parameters

direction	The direction in which the figure should be moved.

3.7.3.5 newPosition()

```
void mikoli::Figure::newPosition (
             Position position )
```

newPosition Makes it possible to displace a figure by modifying the location of its central point.

Parameters

position	The new position of the figures's central point.

3.7.3.6 rotate()

```
void mikoli::Figure::rotate (
            Direction direction )
```

rotate Makes it possible to rotate a figure to the left or to the right.

Parameters

direction	The direction in which the figure should rotate.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/figure.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/figure.cpp

3.8 mikoli::FiguresBag Class Reference

The **FiguresBag** (p. 21) class.

```
#include <figuresbag.h>
```

Public Member Functions

- **FiguresBag** ()

  FiguresBag *(p. 21) constructor without parameters. This constructor will initialize the* ***FiguresBag*** *(p. 21) with default parameters.*
- ∼**FiguresBag** ()

  FiguresBag *(p. 21) destructor. Deallocate the memory that was previously reserved for the* ***FiguresBag*** *(p. 21).*
- **Figure getNextFigure** ()

  getType() Recover the type of the figure
- void **refresh** ()

  refresh Reinitialize the ***FiguresBag*** *(p. 21) when the* ***FiguresBag*** *(p. 21) is empty.*

3.8.1 Detailed Description

The **FiguresBag** (p. 21) class.

_nextFigure The **Figure** (p. 18) that will become the current **Figure** (p. 18). _listFigures The list of differents figures that will be played.

3.8.2 Member Function Documentation

3.8.2.1 getNextFigure()

```
Figure mikoli::FiguresBag::getNextFigure ( )
```

getType() Recover the type of the figure

Returns

The next **Figure** (p. 18) that will become the current **Figure** (p. 18).

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/figuresbag.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/figuresbag.cpp

3.9 mikoli::MainControl Class Reference

The **MainControl** (p. 22) class This class is the controller of the game. It is the interface between the GUI and the model. It receives the informations from the user and call the methods necessary. It will start the game.

```
#include <maincontrol.h>
```

Public Member Functions

- void **startNewGame** ()
  startNewGame It create a new game with standard options
- void **selectMode** (GameMode gameMode)
  selectMode It makes possible to choose differents game mode.
- void **customize** ()
  customize It makes possible to create cusztomized Figures that could be added to the game
- void **exitApp** ()
  exitApp Method that closes the application.

3.9.1 Detailed Description

The **MainControl** (p. 22) class This class is the controller of the game. It is the interface between the GUI and the model. It receives the informations from the user and call the methods necessary. It will start the game.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/maincontrol.h

3.10 mikoli::Mode Class Reference

The **Mode** (p. 22) class Manage the game mode of the game. By this class, it's possible to switch between different game modes.

```
#include <mode.h>
```

Public Member Functions

- **Mode** ()

  Mode *(p. 22) Constructor by default.*
- **Mode** (Gamemode, int)

  Mode *(p. 22) Constructor with parameters. Gamemode is the game mode of the game. int is the goal to reach.*
- Gamemode **getGameMode** ()

  getGameMode
- int **getGoal** ()

  getGoal
- void **setGameMode** (Gamemode)

  setGameMode Set a new game mode to the game.
- void **setGoal** (int)

  setGoal Set a new goal to reach.

3.10.1 Detailed Description

The **Mode** (p. 22) class Manage the game mode of the game. By this class, it's possible to switch between different game modes.

3.10.2 Member Function Documentation

3.10.2.1 getGameMode()

```
Gamemode mikoli::Mode::getGameMode ( )
```

getGameMode

Returns

The current game mode of the game.

3.10.2.2 getGoal()

```
int mikoli::Mode::getGoal ( )
```

getGoal

Returns

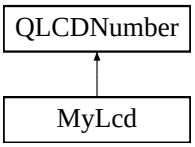
The goal to reach.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/mode.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/mode.cpp

3.11 MyLcd Class Reference

Inheritance diagram for MyLcd:



Public Member Functions

- **MyLcd** (int value)
- void **setValue** (int value)

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

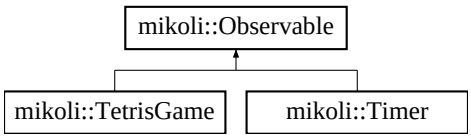
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/mylcd.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/mylcd.cpp

3.12 mikoli::Observable Class Reference

The **Observable** (p. 24) class Interface implemented by the classes that have to be observed.

`#include <observable.h>`

Inheritance diagram for mikoli::Observable:



Public Member Functions

- void **AddObs** (**Observer** ∗obs)
- void **DelObs** (**Observer** ∗obs)
- void **Notify** (void)

3.12.1 Detailed Description

The **Observable** (p. 24) class Interface implemented by the classes that have to be observed.

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

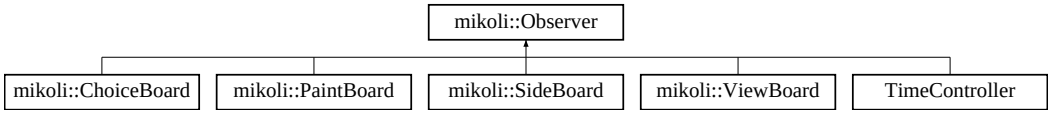
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/observable.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/observable.cpp

3.13 mikoli::Observer Class Reference

The **Observer** (p. 25) class Implemented by the class that have to Observe another class(observable)

`#include <observer.h>`

Inheritance diagram for mikoli::Observer:



Public Member Functions

- virtual void **Update** ()=0
- void **AddObs** (**Observable** ∗obs)
- void **DelObs** (**Observable** ∗obs)

Protected Types

- typedef std::list< **Observable** ∗ >::iterator **iterator**
- typedef std::list< **Observable** ∗ >::const_iterator **const_iterator**

Protected Attributes

- std::list< **Observable** ∗ > **m_list**

3.13.1 Detailed Description

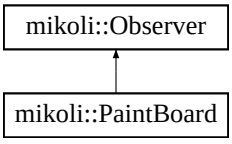
The **Observer** (p. 25) class Implemented by the class that have to Observe another class(observable)

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/observer.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/observer.cpp

3.14 mikoli::PaintBoard Class Reference

Inheritance diagram for mikoli::PaintBoard:



Public Member Functions

- **PaintBoard** (QWidget &fenetre, **TetrisGame** &game)
- void **Update** (const **Observable** ∗observable)

Protected Member Functions

- void **paintEvent** (QWidget &fenetre, QPaintEvent ∗event)

Additional Inherited Members

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/paintboard.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/paintboard.cpp

3.15 mikoli::Position Class Reference

The **Position** (p. 26) class.

`#include <position.h>`

Public Member Functions

- **Position** ()
  Position *(p. 26)'s constructor without parameters. This constructor will set the _x attribute and Y to 0.*
- **Position** (int x, int y)
  Position *(p. 26)'s constructor with 2 parameters.*
- ∼**Position** ()
  Position *(p. 26)'s destructor.*
- int **getX** ()
  getX
- int **getY** ()
  getY
- void **setX** (int x)
  setX
- void **setY** (int y)
  setY
- bool **isSame** (**Position** position)
  isSame

3.15.1 Detailed Description

The **Position** (p. 26) class.

This class will be used to determinate the **Block** (p. 5)'s **Position** (p. 26) in the board. _x The abscissa. _y The ordinate.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Position()

```
mikoli::Position::Position (
            int x,
            int y )
```

Position (p. 26)'s constructor with 2 parameters.

Parameters

x	the value for horizontal axis.
y	the value for vertical axis.

3.15.3 Member Function Documentation

3.15.3.1 getX()

```
int mikoli::Position::getX ( )
```

getX

Returns

The value of _x.

3.15.3.2 getY()

```
int mikoli::Position::getY ( )
```

getY

Returns

The value of _y.

3.15.3.3 isSame()

```
bool mikoli::Position::isSame (
             Position position )
```

isSame

Parameters

position	The position to compare to.

Returns

true if The position is the same, false otherwise.

3.15.3.4 setX()

```
void mikoli::Position::setX (
            int x )
```

setX

Parameters

x	The new value for _x.

3.15.3.5 setY()

```
void mikoli::Position::setY (
            int y )
```

setY

Parameters

y	The new value for _y.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/position.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/position.cpp

3.16 qt_meta_stringdata_Widget_t Struct Reference

Public Attributes

- QByteArrayData **data** [1]
- char **stringdata0** [7]

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/moc_widget.cpp

3.17 mikoli::Score Class Reference

The **Score** (p. 29) class This class will inform the score of the user and the number of lines he's done. Also used buy the GUI.

`#include <score.h>`

Public Member Functions

- **Score** ()

  Score *(p. 29)'s constructor without parameters. This only constructor will set the score to 0 and the number of lines to 0 at the start of the game.*
- ∼**Score** ()

  Score *(p. 29)'s destructor.*
- int **getNbLines** () const

  getNbLines
- int **getScore** () const

  getScore
- int **getLevel** () const

  getLevel
- void **updateScore** (int nbL, int nbDrop)

  updateScore
- int **calculScore** (int nbL, int nbDrop)

  calculScore

3.17.1 Detailed Description

The **Score** (p. 29) class This class will inform the score of the user and the number of lines he's done. Also used buy the GUI.

3.17.2 Member Function Documentation

3.17.2.1 calculScore()

```
int mikoli::Score::calculScore (
            int nbL,
            int nbDrop )
```

calculScore

Parameters

nbL	the number of lines the player made at the last move.
nbDrop	the number of lines the player cross during a fall.

Returns

the amount to add to the score.

3.17.2.2 getLevel()

```
int mikoli::Score::getLevel ( ) const
```

getLevel

Returns

The current level

3.17.2.3 getNbLines()

```
int mikoli::Score::getNbLines ( ) const
```

getNbLines

Returns

The number of lines made by the player from the start of the game.

3.17.2.4 getScore()

```
int mikoli::Score::getScore ( ) const
```

getScore

Returns

The current score

3.17.2.5 updateScore()

```
void mikoli::Score::updateScore (
            int nbL,
            int nbDrop )
```

updateScore

Parameters

the	number of lines the player made at the last move. This number will be added to the previous score.
nbDrop	the number of lines the player cross during a fall.

Exceptions

TetrisException *(p. 33)*	if nb is negative.

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

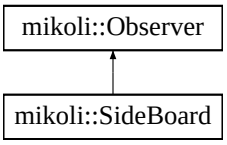
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/score.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/score.cpp

3.18 mikoli::SideBoard Class Reference

The SideBoartd class.

`#include <sideboard.h>`

Inheritance diagram for mikoli::SideBoard:



Public Member Functions

- **SideBoard** ()

  Constructor of ***SideBoard*** *(p. 31) without parameters.*
- **SideBoard** (QWidget &fenetre, **TetrisGame** &game)

  Constructor of ***SideBoard*** *(p. 31) with parameters.*
- void **setDisplay** ()

  To display the ***SideBoard*** *(p. 31).*
- void **Update** ()

  The method executed when the observable changed.

Additional Inherited Members

3.18.1 Detailed Description

The SideBoartd class.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 SideBoard()

```
mikoli::SideBoard::SideBoard (
            QWidget & fenetre,
             TetrisGame & game )
```

Constructor of **SideBoard** (p. 31) with parameters.

Parameters

fenetre	the **Widget** (p. 46) in which the **SideBoard** (p. 31) has to appear
game	the **TetrisGame** (p. 33) (observable)

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/sideboard.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/sideboard.cpp

3.19 mikoli::SoundPlayer Class Reference

The **SoundPlayer** (p. 32) class Each instance of this class is a sound. Through this class, we can handle the sound : play, pause, ...

```
#include <soundplayer.h>
```

Public Member Functions

- **SoundPlayer** ()

  SoundPlayer *(p. 32) Constructor by default.*
- **SoundPlayer** (std::string, bool)

  SoundPlayer *(p. 32) Constructor with parameter: A string for sound's name A boolean set to true if the sound must play in loop, false otherwise.*
- void **play** ()

  play Play the sound.
- void **setVolume** (int)

  setVolume Change the volume of the sound.
- void **stop** ()

  stop Stop the sound.
- void **switchMute** ()

  switchMute Mute the sound.
- void **reset** ()

  reset Replace the sound to the beginning.

3.19.1 Detailed Description

The **SoundPlayer** (p. 32) class Each instance of this class is a sound. Through this class, we can handle the sound : play, pause, ...

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

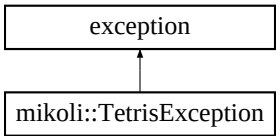
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/soundplayer.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/soundplayer.cpp

3.20 mikoli::TetrisException Class Reference

The **TetrisException** (p. 33) class This is the exception class used for the game .

`#include <tetrisexception.h>`

Inheritance diagram for mikoli::TetrisException:



Public Member Functions

- **TetrisException** (const char ∗Msg)
- const char ∗ **what** () const throw ()

3.20.1 Detailed Description

The **TetrisException** (p. 33) class This is the exception class used for the game .

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

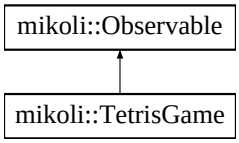
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/tetrisexception.h

3.21 mikoli::TetrisGame Class Reference

The **TetrisGame** (p. 33) class.

`#include <tetrisgame.h>`

Inheritance diagram for mikoli::TetrisGame:



Public Member Functions

- **TetrisGame** ()

  TetrisGame *(p. 33)'s constructor without parameter. Build a standard game with level mode "normal" and difficulty "normal".*
- **TetrisGame** (int width, int height)

  TetrisGame *(p. 33)'s constructor with size parameters. Build a standard game with level mode "normal" and difficulty "normal".*
- ∼**TetrisGame** ()

  TetrisGame *(p. 33) destructor. Deallocate the memory that was previously reserved for the Game.*
- **Figure getCurrentFig** ()

  getCurrentFig
- std::vector< **Block** > **getShadowCF** () const

  getShadowCF
- **Figure getNextFig** ()

  getNextFig
- **Board getBoard** ()

  getBoard
- bool **isGameOver** ()

  isGameOver
- bool **isWon** ()

  isWon
- int **getScore** ()

  getScore
- int **getLevel** ()

  getLevel
- int **getNbLines** ()

  getNbLines
- **Mode getMode** ()

  getMode
- bool **isBegin** ()

  isBegin
- bool **isPaused** ()

  isPaused
- int **getSpeed** ()

  statutSpeed
- bool **getIsFalling** ()

  statutIsFalling
- bool **checkIfIsBlocked** ()
- **Timer** ∗ **getTimer** ()

  getTimer
- void **isWin** ()

  isWin Check if the conditions of win are completed.
- void **move** (Direction direction)

  move
- void **rotate** (Direction direction)

  rotate
- void **fall** ()

  fall
- void **fallSlow** ()

  fallSlow

- void **endMove** (int nbDrop)

  endMove
- void **endGame** ()

  endGame Stop the "tetris" music and play the "game over" sound, replace all the blocks from the board with grey blocks.
- void **calculateShadow** ()

  calculateShadow Calcul the positions of the shadow according to the positions of the current figure.
- void **switchPause** ()

  switchPause Switch the game into paused / not paused mode.
- void **setIsBegin** (bool **isBegin**)

  setIsBegin
- void **setIsFalling** (bool isFalling)

  setIsFalling
- void **setAutoDown** (bool autoDown)

  setAutoDown
- void **start** ()

  start() *(p. 35) To start the game initialized. Actives the timer and make te "first current figure" moving down.*
- void **startWithMode** (Gamemode, int)

  startWithMode Do the same as ***start()*** *(p. 35) but change the game mode and the goal.*
- void **restart** ()

  restart Restart the game: reset the score, level, number of lines, board, figure's bag, reset the sounds, reset the attriubutes, ...

3.21.1 Detailed Description

The **TetrisGame** (p. 33) class.

This class will be used for build a new game.

_figuresBag The list in which are each figure that will become the next figure and then the current **Figure** (p. 18). _currentFigure The figure that can be rotated or moved during its descent. _nextFigure

3.21.2 Member Function Documentation

3.21.2.1 endMove()

```
void mikoli::TetrisGame::endMove (
            int nbDrop )
```

endMove

Parameters

nbDrop	The number of lines crossed by a fall. Handle the end of a move: Add the current figure to the board, check and remove lines if necessary, update the score, change the current figure with the next one, check if the next current figure can be placed in the board, check the conditions of win, ..., call **endGame()** (p. 35) is necessary.

3.21.2.2 getBoard()

```
Board mikoli::TetrisGame::getBoard ( )
```

getBoard

Returns

The board.

3.21.2.3 getCurrentFig()

```
Figure mikoli::TetrisGame::getCurrentFig ( )
```

getCurrentFig

Returns

The current figure

3.21.2.4 getIsFalling()

```
bool mikoli::TetrisGame::getIsFalling ( )
```

statutIsFalling

Returns

True if the current figure is falling.

3.21.2.5 getLevel()

```
int mikoli::TetrisGame::getLevel ( )
```

getLevel

Returns

The current level.

3.21.2.6 getMode()

```
Mode mikoli::TetrisGame::getMode ( )
```

getMode

Returns

The mode of the game.

3.21.2.7 getNbLines()

```
int mikoli::TetrisGame::getNbLines ( )
```

getNbLines

Returns

The current number of lines made.

3.21.2.8 getNextFig()

```
Figure mikoli::TetrisGame::getNextFig ( )
```

getNextFig

Returns

The next figure.

3.21.2.9 getScore()

```
int mikoli::TetrisGame::getScore ( )
```

getScore

Returns

The current score.

3.21.2.10 getShadowCF()

```
std::vector< Block > mikoli::TetrisGame::getShadowCF ( ) const
```

getShadowCF

Returns

A vector with the blocks of the shadow.

3.21.2.11 getSpeed()

```
int mikoli::TetrisGame::getSpeed ( )
```

statutSpeed

Returns

The actual speed of the game. It's a calcul made according of the level.

3.21.2.12 getTimer()

```
Timer * mikoli::TetrisGame::getTimer ( )
```

getTimer

Returns

The timer instance with the elapsed time.

3.21.2.13 isBegin()

```
bool mikoli::TetrisGame::isBegin ( )
```

isBegin

Returns

True if the game is began, false otherwise.

3.21.2.14 isGameOver()

```
bool mikoli::TetrisGame::isGameOver ( )
```

isGameOver

Returns

True is the game is over, false otherwise.

3.21.2.15 isPaused()

```
bool mikoli::TetrisGame::isPaused ( )
```

isPaused

Returns

True if the game is paused, false otherwise.

3.21.2.16 isWon()

```
bool mikoli::TetrisGame::isWon ( )
```

isWon

Returns

True if the player reached it's goal, false otherwise.

3.21.2.17 move()

```
void mikoli::TetrisGame::move (
            Direction direction )
```

move

Parameters

direction	The direction we want to move. Move the current figure in the direction "direction".

3.21.2.18 rotate()

```
void mikoli::TetrisGame::rotate (
            Direction direction )
```

rotate

Parameters

direction	The direction we want to rotate. Rotate the current figure in the direction "direction".

3.21.2.19 setAutoDown()

```
void mikoli::TetrisGame::setAutoDown (
            bool autoDown )
```

setAutoDown

Parameters

autoDown	Set the attribute _autoDown with the parameter.

3.21.2.20 setIsBegin()

```
void mikoli::TetrisGame::setIsBegin (
            bool isBegin )
```

setIsBegin

Parameters

isBegin	Set the attribute _isBegin with the parameter.

3.21.2.21 setIsFalling()

```
void mikoli::TetrisGame::setIsFalling (
            bool isFalling )
```

setIsFalling

Parameters

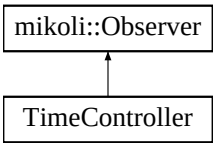
isFalling	Set the attribute _isFalling with the parameter.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/tetrisgame.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/tetrisgame.cpp

3.22 TimeController Class Reference

Inheritance diagram for TimeController:



Public Member Functions

- **TimeController** (**Widget** &w)
- void **Update** ()

Friends

- class **Widget**

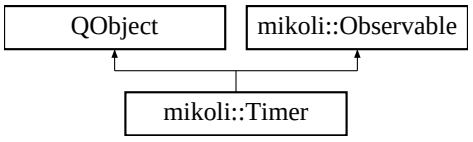
Additional Inherited Members

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/widget.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/widget.cpp

3.23 mikoli::Timer Class Reference

Inheritance diagram for mikoli::Timer:



Public Slots

- void **MySlot** ()

  MySlot Inform the view to update the time elapsed view.

Public Member Functions

- **Timer** ()

  Timer *(p. 41) Constructor by default.*
- std::pair< int, int > **statutTimeGame** (void)

  statutTimeGame
- int **getSeconds** ()

  getSeconds
- int **getMinutes** ()

  getMinutes
- int **getHours** ()

  getHours
- int **getTotalTime** ()

  getTotalTime
- void **play** ()

  play Start the timer.
- void **pause** ()

  pause Pause the timer.
- void **updateDuration** ()

  updateDuration Update the duration attribute
- void **reset** ()

  reset Reset the timer.

Public Attributes

- QTimer * **_timer**

  _timer ***Timer*** *(p. 41) that every second, inform the game to update it's view of the time elapsed.*

3.23.1 Member Function Documentation

3.23.1.1 getHours()

```
int mikoli::Timer::getHours ( )
```

getHours

Returns

The number of hours elapsed.

3.23.1.2 getMinutes()

```
int mikoli::Timer::getMinutes ( )
```

getMinutes

Returns

The number of minutes elapsed.

3.23.1.3 getSeconds()

```
int mikoli::Timer::getSeconds ( )
```

getSeconds

Returns

The number of seconds elapsed.

3.23.1.4 getTotalTime()

```
int mikoli::Timer::getTotalTime ( )
```

getTotalTime

Returns

The total time elapsed in seconds.

3.23.1.5 statutTimeGame()

```
std::pair< int, int > mikoli::Timer::statutTimeGame (
            void  )
```

statutTimeGame

Returns

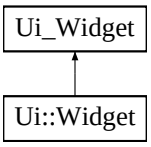
A pair with the minutes and seconds elapsed since the start of the game.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/timer.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/timer.cpp

3.24 Ui_Widget Class Reference

Inheritance diagram for Ui_Widget:



Public Member Functions

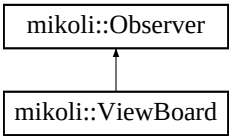
- void **setupUi** (QWidget ∗ **Widget**)
- void **retranslateUi** (QWidget ∗ **Widget**)

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/ui_widget.h

3.25 mikoli::ViewBoard Class Reference

Inheritance diagram for mikoli::ViewBoard:



Public Member Functions

- **ViewBoard** ()
  *The constructor of **ViewBoard** (p. 44) without parameter.*
- **ViewBoard** (QWidget &fenetre, **TetrisGame** &game)
  *The constructor of **ViewBoard** (p. 44) with parameters.*
- void **setDisplay** ()
  The method called to display the board.
- void **paint** (**Block** bl, int blSize, QColor color, int a, int b, int c, int d, double opacity, bool grad)
  Method to paint blocks with parameters With a relief effect.
- void **Update** ()
  The method executed when the observable changed.

Additional Inherited Members

3.25.1 Constructor & Destructor Documentation

3.25.1.1 ViewBoard()

```
mikoli::ViewBoard::ViewBoard (
            QWidget & fenetre,
             TetrisGame & game )
```

The constructor of **ViewBoard** (p. 44) with parameters.

Parameters

fenetre	the **Widget** (p. 46) in which the Viewboard has to appear
game	The **TetrisGame** (p. 33) (observed)

3.25.2 Member Function Documentation

3.25.2.1 paint()

```
void mikoli::ViewBoard::paint (
             Block bl,
            int blSize,
            QColor color,
            int a,
            int b,
            int c,
            int d,
            double opacity,
            bool grad )
```

Method to paint blocks with parameters With a relief effect.

Parameters

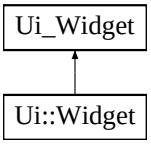
bl	the block to paint
blSize	the block's width
color	the block's color
a	value used to print the first level of the block painting (position)
b	value used to print the first level of the block painting (width)
c	value used to print the second level of the block painting (position)
d	value used to print the second level of the block painting (width)
opacity	the opacity of the block painting

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/viewboard.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/viewboard.cpp

3.26 Ui::Widget Class Reference

Inheritance diagram for Ui::Widget:



Additional Inherited Members

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

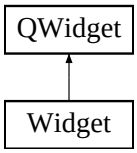
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/ui_widget.h

3.27 Widget Class Reference

Widget (p. 46) class used to display the Game.

`#include <widget.h>`

Inheritance diagram for Widget:



Public Member Functions

- **Widget** (QWidget ∗parent=0)
- **TetrisGame** & **getGame** ()

Public Attributes

- **SoundPlayer** ∗ **_startSound**
- **SoundPlayer** ∗ **_moveSound**

Protected Member Functions

- void **timerEvent** (QTimerEvent ∗event) override

Friends

- class **TimeController**

3.27.1 Detailed Description

Widget (p. 46) class used to display the Game.

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

- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/widget.h
- C:/Users/Olivier/Desktop/Final Tetris cleaned/Tetris_Mikoli/widget.cpp

Index