mkRPG

Generated by Doxygen 1.7.6.1

Mon Nov 14 2016 19:12:00

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 src::parsing::map_parser Namespace Reference

Functions

- def parse_cell
- def map_parser
- def get_size
- def **gen_map**

5.1.1 Detailed Description

```
This module handles xml parsing for maps description files.
```

5.1.2 Function Documentation

5.1.2.1 def src.parsing.map_parser.get_size (*tree*)

```
Gets the size of the map.
```

5.1.2.2 def src.parsing.map_parser.map_parser (*map_xml*)

```
The main parser for the map xml file.
```

5.1.2.3 def src.parsing.map_parser.parse_cell (*cell_object*)

```
Parses a CellType attribute.
```

Chapter 6

Class Documentation

6.1 src.path.Archi Class Reference

Public Member Functions

- def **__init__**
- def get_src_file
- def get_static_file
- def get_xml_file
- def list_files
- def get_src_dir
- def get_static_dir
- def get_xml_dir

Public Attributes

- **main_directory**

6.1.1 Detailed Description

```
This class manages the arhitecture of the project.
It allows the user to travel in the file system of the game, to get
the XML files and others (PNG, configuration files...)
Moreover, it should be cross-platform compliant
```

6.1.2 Member Function Documentation

6.1.2.1 def src.path.Archi.get_src_dir (*self, dir_path*)

```
Gets the given dir_path with respect to the src folder.
```

6.1.2.2 def src.path.Archi.get_src_file (*self, file_path, mode = 'r'*)

```
Gets the path of the src directory.
At least used by the src scripts.
```

6.1.2.3 def src.path.Archi.get_static_dir (*self, dir_path*)

```
Gets the given dir_path with respect to the static folder.
```

6.1.2.4 def src.path.Archi.get_static_file (*self, file_path, mode = 'r'*)

```
Gets the path of the static files directory.  Static files are
basically all graphical files, and a description of the common world
```

6.1.2.5 def src.path.Archi.get_xml_dir (*self, dir_path*)

```
Gets the given dir_path with respect to the xml folder.
```

6.1.2.6 def src.path.Archi.get_xml_file (*self, file_path, mode = 'r'*)

```
Gets the path of a xml file describing a world, a scenario, or a
campaign.
```

6.1.2.7 def src.path.Archi.list_files (*self, dir_path*)

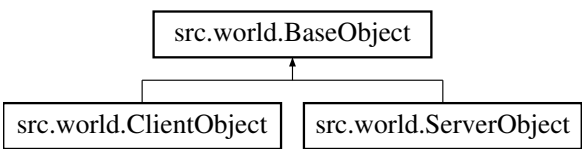
```
List all the files present in the dir_path, if it is a dir.
Else raise a FileNotFoundError.
```

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

- src/path.py

6.2 src.world.BaseObject Class Reference

Inheritance diagram for src.world.BaseObject:



Public Member Functions

- def **__init__**
- def **__getattr__**
- def **load**
- def **contextEval**

Public Attributes

- **ident**
- **params**

Static Public Attributes

- int **ident** = 0
- dictionary **ids** = {}

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

- src/world.py

6.3 BDock Class Reference

Public Slots

- void **setTitle** (QString s)

Public Member Functions

- **BDock** (QString title, BDockWidget ∗dock, QWidget ∗parent=0)
- bool **unfold** () const
- void **setUnfold** (bool v)
- int **currentSize** () const
- void **setCurrentSize** (int t)

Properties

- bool **unfold**
- int **currentSize**

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

- src/editor/GUI/Tabs/Docks/bdock.h
- src/editor/GUI/Tabs/Docks/bdock.cpp

6.4 BDocksZone Class Reference

Public Types

- enum ScrollBarMode { AlwaysVisible, Adaptative, Fixed }

  The ScrollBarMode enum describe the way the BDocksZone reacts when a scroll bar is needed.

Public Slots

- void **swap** (bool anim=true)

Public Member Functions

- **BDocksZone** (QWidget *parent=0)
- void **setUnfold** (bool u, bool anim=true)
- const BinaryStateMachine * **states** () const
- int **length** () const
- void **setLength** (int t)
- ScrollBarMode **scrollBarMode** () const
- void **setScrollBarMode** (ScrollBarMode m)
- int **currentLength** () const
- void **addDock** (QString title, BDockWidget *dock)

Protected Slots

- void **setCurrentLenght** (int t)

Properties

- int **length**
- int **currentLength**

6.4.1 Member Enumeration Documentation

6.4.1.1 enum BDocksZone::ScrollBarMode

The ScrollBarMode enum describe the way the BDocksZone reacts when a scroll bar is needed.

Enumerator:

AlwaysVisible Always show the scroll bar, even if it is usless

Adaptative Show the scroll bar when needed, adaptating the docks length

Fixed Show the scroll bar when needed, keeping the docks length fixed

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

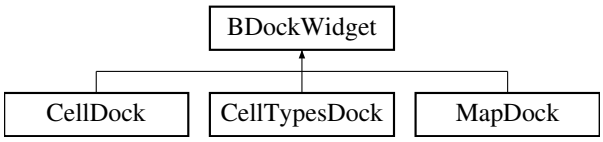
- src/editor/GUI/Tabs/Docks/bdockszone.h
- src/editor/GUI/Tabs/Docks/bdockszone.cpp

6.5 BDockWidget Class Reference

The BDockWidget class is the base class for game-related docks.

`#include <bdockwidget.h>`

Inheritance diagram for BDockWidget:



Public Slots

- virtual void **updateGame** ()

Signals

- void **gameModified** ()
- void **changeDockName** (QString)

Public Member Functions

- **BDockWidget** (QWidget ∗parent=0)
- void **setGame** (Game ∗g)

Protected Attributes

- Game ∗ **game**

6.5.1 Detailed Description

The BDockWidget class is the base class for game-related docks.

It provides common functions for set game, update, ...

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

- src/editor/GUI/Tabs/Docks/bdockwidget.h
- src/editor/GUI/Tabs/Docks/bdockwidget.cpp

6.6 BHandler Class Reference

Public Member Functions

- bool **startElement** (const QString &namespaceURI, const QString &localName, const QString &qName, const QXmlAttributes &atts)
- bool **endElement** (const QString &namespaceURI, const QString &localName, const QString &qName)

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

- src/editor/GUI/editor.h

6.7 BinaryStateMachine Class Reference

The BinaryStateMachine class is a simple QStateMachine with two states.

`#include <intertie.h>`

Public Slots

- void **swap** ()
- void **setPositive** (bool p)
- void **setNegative** (bool n)

Signals

- void **swapped** (bool)
- void **__swap** ()

Public Member Functions

- **BinaryStateMachine** (QObject ∗parent=0)
- void **defineProperty** (QObject ∗obj, const char ∗prop)
- void **defineProperty** (QObject ∗obj, const char ∗prop, QVariant yesValue, QVariant noValue)
- bool **isPositive** () const
- bool **isNegative** () const

6.7.1 Detailed Description

The BinaryStateMachine class is a simple QStateMachine with two states.

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

- src/editor/GUI/Tabs/Docks/intertie.h
- src/editor/GUI/Tabs/Docks/intertie.cpp

6.8 BLayout Class Reference

Signals

- void **sizeChanged** (int)

Public Member Functions

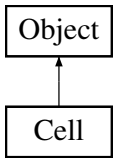
- **BLayout** (QWidget ∗parent=0)
- void **setOrientation** (Qt::Orientation o)
- void **insert** (BDock ∗d, int ind=-1)
- void **setSpacing** (int e)
- void **setLength** (int t)
- int **spacing** () const

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

- src/editor/GUI/Tabs/Docks/bdockszone.h
- src/editor/GUI/Tabs/Docks/bdockszone.cpp

6.9 Cell Class Reference

Inheritance diagram for Cell:



Public Member Functions

- **Cell** (Game ∗g=nullptr)
- bool **isSelected** () const

- void **setSelected** (bool s=true)
- void **invertSelected** ()
- **ParamObj** (cellType, CellType, c) ObjectsMap(c

Public Attributes

- **o**
- **O**
- **bject**

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

- src/editor/Game/map.h
- src/editor/Game/map.cpp

6.10 src.world.Cell Class Reference

Public Member Functions

- def **__init__**

Public Attributes

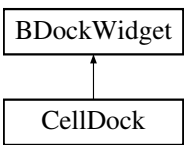
- **entities**
- **objects**

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

- src/world.py

6.11 CellDock Class Reference

Inheritance diagram for CellDock:



Public Slots

- void **updateGame** ()
- void **selectionChanged** ()

Public Member Functions

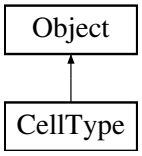
- **CellDock** (QWidget ∗parent=0)

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

- src/editor/GUI/Tabs/celldock.h
- src/editor/GUI/Tabs/celldock.cpp

6.12 CellType Class Reference

Inheritance diagram for CellType:



Public Member Functions

- **CellType** (Game ∗g)

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

- src/editor/Game/map.h
- src/editor/Game/map.cpp

6.13 CellTypeListModel Class Reference

Public Member Functions

- **CellTypeListModel** (World ∗w, QObject ∗parent=0)
- int **rowCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- QVariant **data** (const QModelIndex &index, int role) const Q_DECL_OVERRIDE
- bool **insertRows** (int row, int count, const QModelIndex &parent) Q_DECL_OVERRIDE

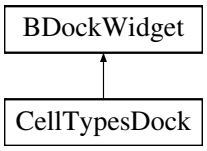
- bool **removeRows** (int row, int count, const QModelIndex &parent) Q_DECL_OVERRIDE

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

- src/editor/Game/mapslistmodel.h
- src/editor/Game/mapslistmodel.cpp

6.14 CellTypesDock Class Reference

Inheritance diagram for CellTypesDock:



Public Member Functions

- **CellTypesDock** (QWidget ∗parent=0)
- void **updateGame** ()

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

- src/editor/GUI/Tabs/celltypesdock.h
- src/editor/GUI/Tabs/celltypesdock.cpp

6.15 src.client.CellViewer Class Reference

Public Member Functions

- def **__init__**
- def **display**

Public Attributes

- **cell**

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

- src/client.py

6.16 ClCoords Class Reference

The ClCoords class describe positions with cell coordinates.

`#include <mappainter.h>`

Public Member Functions

- **ClCoords** (qreal x, qreal y)
- **ClCoords** (const QPointF &p)

6.16.1 Detailed Description

The ClCoords class describe positions with cell coordinates.

Theses coordinates describe each point relatively to the cell grid. They correspond to the isometric 3D world.

See also RlCoords, PtCoords and PxCoords

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

- src/editor/Game/mappainter.h

6.17 src.client.Client Class Reference

Public Member Functions

- def **__init__**
- def **__del__**
- def **run**
- def **handleOrder**

Public Attributes

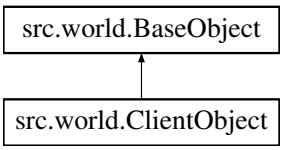
- **net**
- **world**
- **win**
- **mv**
- **interactions**
- **perso**
- **orderDispatcher**
- **lastUpdate**

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

- src/client.py

6.18 src.world.ClientObject Class Reference

Inheritance diagram for src.world.ClientObject:



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

- src/world.py

6.19 Editor Class Reference

The Editor class is the main window of the Editor.

`#include <editor.h>`

Public Member Functions

- **Editor** (QStringList args, QWidget ∗parent=0)

6.19.1 Detailed Description

The Editor class is the main window of the Editor.

It is composed of tabs that offer editing facilities.

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

- src/editor/GUI/editor.h
- src/editor/GUI/editor.cpp

6.20 src.world.Entity Class Reference

Public Member Functions

- def **__init__**

Public Attributes

- **quests**
- **inventory**

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

- src/world.py

6.21 Game Class Reference

The Game class gather the differents parts needed to describe a game.

`#include <game.h>`

Public Member Functions

- int **newIdent** ()
- World * **world** ()
- Map * **currentMap** ()
- void **setCurrentMap** (Map *m)
- void **addImage** (Image *im)

6.21.1 Detailed Description

The Game class gather the differents parts needed to describe a game.

It contains mainly the World, and the ressources used by it (images and strings)

For editing purposes, it contains also the active map (the one being editing)

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

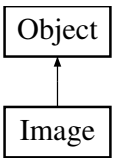
- src/editor/Game/game.h
- src/editor/Game/game.cpp

6.22 Image Class Reference

The Image class stores an external file in a QImage, and gives each image ressources a unique identifier.

`#include <object.h>`

Inheritance diagram for Image:



Public Member Functions

- **Image** (Game ∗g, const QString &fileName)
- bool isValid () const
- const QImage & **image** () const
- const QSize **size** () const

6.22.1 Detailed Description

The Image class stores an external file in a QImage, and gives each image ressources a unique identifier.

6.22.2 Member Function Documentation

6.22.2.1 bool Image::isValid ( ) const [inline, virtual]

return true if the object has been initialised

Reimplemented from Object.

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

- src/editor/Game/object.h
- src/editor/Game/object.cpp

6.23 src.interactions.Interaction Class Reference

Public Member Functions

- def **__init__**
- def **load**

Public Attributes

- **target**
- **type**
- **key**
- **event**

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

- src/interactions.py

6.24 Intertie Class Reference

The Intertie class provide int that move smoothly from their value to an objective.

`#include <intertie.h>`

Public Slots

- void **setValue** (int v, bool inert=true)
- void **setMaximumSpeed** (int vM)
- void **setAcceleration** (int a)
- void **setUpdateInterval** (int d)

Signals

- void **modificationFinished** (int)
- void **valueChanged** (int)

Public Member Functions

- **Intertie** (QObject ∗parent=0)
- int **value** () const
- void **link** (QObject ∗obj, const char ∗prop)

6.24.1 Detailed Description

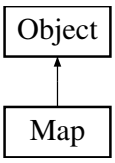
The Intertie class provide int that move smoothly from their value to an objective.

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

- src/editor/GUI/Tabs/Docks/intertie.h
- src/editor/GUI/Tabs/Docks/intertie.cpp

6.25 Map Class Reference

Inheritance diagram for Map:



Public Member Functions

- **Map** (Game ∗g=nullptr)
- **Getter** (width) Getter(height) QSize size() const
- void **setWidth** (int w)
- void **setHeight** (int h)
- void **resize** (int w, int h)
- **Param** (angleX, AngleX) Param(angleY
- AngleY Cell & **cell** (int i, int j) const
- Cell & **cell** (const QPoint &p) const

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

- src/editor/Game/map.h
- src/editor/Game/map.cpp

6.26 src.world.Map Class Reference

Public Member Functions

- def **__init__**
- def fill

Public Attributes

- **cells**
- **cellsGrid**

6.26.1 Member Function Documentation

6.26.1.1 def src.world.Map.fill (*self*)

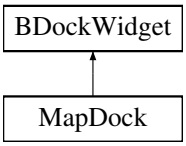
```
Complète les cases par défaut
```

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

- src/world.py

6.27 MapDock Class Reference

Inheritance diagram for MapDock:



Public Member Functions

- **MapDock** (QWidget *parent=0)
- void **updateGame** ()

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

- src/editor/GUI/Tabs/mapdock.h
- src/editor/GUI/Tabs/mapdock.cpp

6.28 MapPainter Class Reference

The MapPainter class that can paint a Map using a QPainter.

`#include <mappainter.h>`

Public Types

- enum Element { Nothing = 0, CellBackground = 1, Grid = 2, CellSelection = 4, CellHighlighting = 8, Objects = 16, All = 31 }

  The Element enum discribes the different elements that can be render.

Signals

- void mapSizeChanged (QSize)
- void viewCenterChanged (QPoint)

Public Member Functions

- MapPainter (QObject *parent=0)
- MapPainter (Map *m, QObject *parent=0)
- void setPaintedElement (Element e, bool painted=true)
- void setPaintedElements (Element e)

- void setMap (Map *m)
- void paint (QPainter &p)
- const QImage & render ()
- RlCoords viewCenter () const
- void setViewCenter (RlCoords relativeCenter)
- void setViewCenter (double relativeCenterX, double relativeCenterY)
- void setViewCenterQuiet (double x, double y)
- double scale () const
- void setScale (double scale)
- void setScaleDomain (double scaleMin, double scaleMax)
- bool setHighlightedCell (const ClCoords &p)
- bool setHighlightedCell (int i, int j)
- QPoint highlightedCell () const
- bool hasHighlightedCell () const
- void resize (QSize s)
- void resize (int wi, int he)
- QSize size () const
- void zoom (double factor, QPointF fixedPoint)
- QPair< bool, bool > move (PxCoords delta, QPointF center)
- QSize virtualSize () const
- PxCoords ptToPxl (PtCoords p) const
- PtCoords pxlToPt (PxCoords p) const
- PtCoords cooToPt (ClCoords p) const
- ClCoords ptToCoo (PtCoords p) const
- PxCoords cooToPxl (ClCoords p) const
- ClCoords pxlToCoo (PxCoords p) const
- PtCoords indToPt (int i, int j) const

6.28.1 Detailed Description

The MapPainter class that can paint a Map using a QPainter.

The class take charge of the different ratios of the map rendering and the area in which it will be rendered.

Note

The view is kept updated with the associated map at each paint or render call. It is thus just needed to call one of these functions to update the view after a modification.

To ensure a type checking security about the different types of coordinates that are used, four different types that inherit from QPointF are used : RlCoords, ClCoords, PtCoords and PxCoords

6.28.2 Member Enumeration Documentation

6.28.2.1 enum MapPainter::Element

The Element enum discribes the different elements that can be render.

This includes both map's objects and user interaction and editing elements.

Element value can be used as flags using the operators operator| "|", operator| "&", operator| "^".

See also Cell, CellType

Enumerator:

Nothing Represent no elements

CellBackground The bacground associated to the cell type

Grid A thin grid that separate cells

CellSelection Graphical information about the selection state

CellHighlighting Graphical visualisation of the cells the mouse is over

Objects The objects that lay on the cells

All Represent all elements

6.28.3 Constructor & Destructor Documentation

6.28.3.1 MapPainter::MapPainter (QObject * *parent* = 0)

Constructs a new MapPainter with a default size of (42,42).

6.28.3.2 MapPainter::MapPainter (Map * *m*, QObject * *parent* = 0)

Constructs a new MapPainter with a default size of (42,42), and loads the map m.

6.28.4 Member Function Documentation

6.28.4.1 PtCoords MapPainter::cooToPt (ClCoords *p*) const

Converts cells indice to virtual point coordinates

6.28.4.2 PxCoords MapPainter::cooToPxl (ClCoords *p*) const

Convenient function equivalent to ptToPxl(cooToPt(p))

6.28.4.3 bool MapPainter::hasHighlightedCell ( ) const

Returns true if a cell is highligthed.

See also highlightedCell and setHighlightedCell.

6.28.4.4 QPoint MapPainter::highlightedCell ( ) const

Returns the integer index of the cell the is highlighted.

See also sethighlightedCell and hasHighlightedCell.

6.28.4.5 PtCoords MapPainter::indToPt (int *i*, int *j*) const

Converts to coordinates

6.28.4.6 void MapPainter::mapSizeChanged (QSize) [signal]

This signal is emitted when the total size of the map's view change.

It appends mainly during scale change and modification on the map (resize, angles setting, ...).

6.28.4.7 QPair< bool, bool > MapPainter::move (PxCoords *delta*, QPointF *center*)

Change the center position from the given center and a pixel difference.

The return value indicate if the expected center was valid (regarding x or y coordinate).

See also setViewCenter.

6.28.4.8 void MapPainter::paint (QPainter & *p*)

Draws the map in the QPaintDevice.

See also render.

6.28.4.9 ClCoords MapPainter::ptToCoo (PtCoords *p*) const

Converts virtual point to cell indice

6.28.4.10 PxCoords MapPainter::ptToPxl (PtCoords *p*) const

Converts virtual point to real pixel coordinates

6.28.4.11 ClCoords MapPainter::pxlToCoo (PxCoords *p*) const

Convenient function equivalent to ptToCoo(pxlToPt(p))

6.28.4.12 PtCoords MapPainter::pxlToPt (PxCoords *p*) const

Converts real pixel to virtual point coordinates

6.28.4.13 const QImage & MapPainter::render ( )

Provides a QImage with a view of the map.

See also paint.

6.28.4.14 void MapPainter::resize (QSize *s*)

Change the size of the view, *ie* the rectangle in which the map will be render.

See also size.

6.28.4.15 void MapPainter::resize (int *wi,* int *he*)

This is an overload function, see resize

6.28.4.16 double MapPainter::scale ( ) const

Returns the current scale of the view.

See also setScale.

6.28.4.17 bool MapPainter::setHighlightedCell (const ClCoords & *p*)

Set the highligthed cell to the one at the ClCoords p

See also highlightedCell and hasHighlightedCell.

6.28.4.18 bool MapPainter::setHighlightedCell (int *i,* int *j*)

This is an overload function, see setViewCenter.

6.28.4.19 void MapPainter::setMap (Map * *m*)

Loads the map, computing the new size of the view area.

6.28.4.20 void MapPainter::setPaintedElement (MapPainter::Element *e,* bool *painted* = `true`)

Enables or disables the render of an element.

See also setPaintedElements.

6.28.4.21 void MapPainter::setPaintedElements (Element *e*)

Set the rendered elements.

See also setPaintedElement.

6.28.4.22 void MapPainter::setScale (double *scale*)

Set the current view scale. This closest value in the scale domain will be used.

See also scale and setScaleDomain.

6.28.4.23 void MapPainter::setScaleDomain (double *scaleMin,* double *scaleMax*)

Set the valid values for the scale.

See also scale and setScale.

6.28.4.24 void MapPainter::setViewCenter (RICoords *relativeCenter*)

Change the view center, using relative coordinates.

If the new center is invalid (the view exceed the map area), the closest valid center is used.

See also viewCenter.

6.28.4.25 void MapPainter::setViewCenter (double *relativeCenterX,* double *relativeCenterY*)

This is an overload function, see setViewCenter.

6.28.4.26 void MapPainter::setViewCenterQuiet (double *x,* double *y*)

does the same as setViewCenter, without emitting the signal viewCenterChanged to avoid event loop.

6.28.4.27 QSize MapPainter::size ( ) const

Return the size of the rectangle in which the map is render. This is also the size of the image returned by render.

See also resize.

6.28.4.28 RICoords MapPainter::viewCenter ( ) const

Return the relative coordinates of the current view center.

See also setViewCenter.

6.28.4.29 void MapPainter::viewCenterChanged (QPoint) [signal]

This signal is emitted when the center of the map change.

It appends mainly during moving on the view and zooming.

6.28.4.30 QSize MapPainter::virtualSize ( ) const

Computes the total size of the image of the map

6.28.4.31 void MapPainter::zoom (double *factor,* QPointF *fixedPoint*)

Multiplying the scale of the view by factor, trying to leave the point center at the same position.

Note

It is not always possible to keep this point fixed, in particulary when the view is resulting view would exceed the map region. In that case, the center is adapt to minimise the difference.

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

- src/editor/Game/mappainter.h
- src/editor/Game/mappainter.cpp

6.29 MapsEditor Class Reference

The MapsEditor class is the tab offering map editing facilities.

```
#include <mapseditor.h>
```

Public Slots

- void **updateGame** ()

Public Member Functions

- **MapsEditor** (QWidget ∗parent=0)
- void **setGame** (Game ∗g)

6.29.1 Detailed Description

The MapsEditor class is the tab offering map editing facilities.

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

- src/editor/GUI/Tabs/mapseditor.h
- src/editor/GUI/Tabs/mapseditor.cpp

6.30 MapsListModel Class Reference

The MapsListModel class provides a presentation class for the Qt Model-View framework.

`#include <mapslistmodel.h>`

Public Slots

- void **update** ()

Public Member Functions

- **MapsListModel** (World ∗w, QObject ∗parent=0)
- int **rowCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- QVariant **data** (const QModelIndex &index, int role) const Q_DECL_OVERRIDE
- bool **insertRows** (int row, int count, const QModelIndex &parent) Q_DECL_OVERRIDE
- bool **removeRows** (int row, int count, const QModelIndex &parent) Q_DECL_OVERRIDE

6.30.1 Detailed Description

The MapsListModel class provides a presentation class for the Qt Model-View framework.

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

- src/editor/Game/mapslistmodel.h
- src/editor/Game/mapslistmodel.cpp

6.31 MapViewer Class Reference

The MapViewer class provides a widget to display and edit a Map using a MapPainter.

`#include <mapviewer.h>`

Public Slots

- void **updateRequest** ()

Signals

- void **viewSizeChanged** (QSize)
- void **selectionChanged** ()

Public Member Functions

- **MapViewer** (QWidget ∗parent=0)
- void **setMap** (Map ∗m)
- void **updateMap** ()
- MapPainter & **mapPainter** ()

6.31.1 Detailed Description

The MapViewer class provides a widget to display and edit a Map using a MapPainter.

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

- src/editor/GUI/Tabs/mapviewer.h
- src/editor/GUI/Tabs/mapviewer.cpp

6.32 src.client.MapViewer Class Reference

Public Member Functions

- def **__init__**
- def **display**

Public Attributes

- **map**
- **world**
- **cellViews**

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

- src/client.py

6.33 src.network.NetworkClient Class Reference

Inherits Thread.

Public Member Functions

- def **__init__**
- def **run**
- def **send**
- def **sendEvent**
- def **kill**

Public Attributes

- **handle**
- **soc**
- **alive**

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

- src/network.py

6.34 src.networkudp.NetworkClient Class Reference

Inherits Thread.

Public Member Functions

- def **__init__**
- def **run**
- def **send**
- def **sendEvent**
- def **kill**

Public Attributes

- **handle**
- **soc**
- **alive**

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

- src/networkudp.py

6.35 src.network.NetworkServer Class Reference

Inherits Thread.

Public Member Functions

- def **__init__**
- def **waitForClients**
- def **run**
- def **sendOrder**
- def **broadcast**
- def **kill**

Public Attributes

- **handle**
- **soc**
- **alive**
- **co**

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

- src/network.py

6.36 src.networkudp.NetworkServer Class Reference

Inherits Thread.

Public Member Functions

- def **__init__**
- def **waitForClients**
- def **run**
- def **sendOrder**
- def **broadcast**
- def **kill**

Public Attributes

- **handle**
- **soc**
- **alive**
- **addr**

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

- src/networkudp.py

6.37 NewGame Class Reference

Public Member Functions

- **NewGame** (QWidget *parent=0)
- QString **name** () const
- QString **folder** () const
- bool **createFolder** () const

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

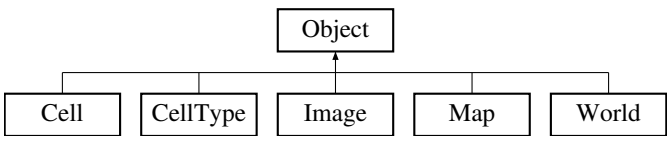
- src/editor/GUI/newgame.h
- src/editor/GUI/newgame.cpp

6.38 Object Class Reference

The Object class is the base class for every part of games.

`#include <object.h>`

Inheritance diagram for Object:



Public Member Functions

- **Object** (Game *g=nullptr)
- void init (Game *g)
- int **ident** () const
- virtual bool isValid () const
- const QDateTime & lastModification () const
- int **getParam** (const QString &p)
- void **touch** ()

Protected Attributes

- Game * **game**
- int **id**
- QMap< QString, int > **params**
- QString **fileName**
- QDateTime **lastEdit**

6.38.1 Detailed Description

The Object class is the base class for every part of games.

Each instance is identified by a game-wide unique identifier.

On each modification, the lastEdit attribute has to be updated, in order that other objects can see that modifications occured.

6.38.2 Member Function Documentation

6.38.2.1 void Object::init (Game * *g*)

initialise the object in case it had been construct with a NULL pointer (array of objects)

6.38.2.2 virtual bool Object::isValid ( ) const [inline, virtual]

return true if the object has been initialised

Reimplemented in Image.

6.38.2.3 const QDateTime& Object::lastModification ( ) const [inline]

return the last time of modification

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

- src/editor/Game/object.h
- src/editor/Game/object.cpp

6.39 src.world.ObjectType Class Reference

Public Member Functions

- def **__init__**
- def **create**

Public Attributes

- **type**

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

- src/world.py

6.40 Options Struct Reference

Classe singleton encapsulant la gestion des options permanentes.

`#include <options.h>`

Public Member Functions

- template<class T >
  T **load** (QString group, QString opt)
- template<class T >
  void **save** (QString group, QString opt, T val)
- bool **isAdaptaive** (QString group, QString opt, bool adapt=true)
- void **setAdaptaive** (QString group, QString opt, bool adapt)
- void **reinitialise** ()

Static Public Member Functions

- static Options & **options** ()

6.40.1 Detailed Description

Classe singleton encapsulant la gestion des options permanentes.

Ajoute à QSettings quelques fonctionnalités :

La possiblité de définir si les options s'adaptent lors de la modificatoin ou non (par exemple, la taille de la fenêtre), voir isAdaptaive, setAdaptaive

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

- src/editor/GUI/options.h
- src/editor/GUI/options.cpp

6.41 src.orders.Order Class Reference

Public Member Functions

- def **__init__**
- def **__getattr__**
- def **__setattr__**
- def **copy**
- def **load**
- def **toBytes**
- def **fromBytes**

Public Attributes

- **type**
- **args**

Static Public Attributes

- list **params** = [None]

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

- src/orders.py

6.42 src.orders.OrderDispatcher Class Reference

Public Member Functions

- def **__init__**
- def treat

Public Attributes

- **world**
- **handle**

6.42.1 Detailed Description

```
pour diminuer la redondance de code client/serveur
```

6.42.2 Member Function Documentation

6.42.2.1 def src.orders.OrderDispatcher.treat (*self, emitter, order*)

```
-> ordre à retransmettre
```

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

- src/orders.py

6.43 src.utils.Perf Class Reference

Public Member Functions

- def **__init__**
- def tic
- def toc
- def show

Public Attributes

- **num**
- **avg**
- **min**
- **max**
- **t**

6.43.1 Detailed Description

```
Calcule les performances d'un morceau de code
```

6.43.2 Member Function Documentation

6.43.2.1 def src.utils.Perf.show (*self*)

```
Affiche le rapport
```

6.43.2.2 def src.utils.Perf.tic (*self*)

```
À lancer avant la fonction
```

6.43.2.3 def src.utils.Perf.toc (*self*)

```
À lancer après la fonction
```

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

- src/utils.py

6.44 PtCoords Class Reference

The PtCoords class describe positions with virtual point coordinates.

```
#include <mappainter.h>
```

Public Member Functions

- **PtCoords** (qreal x, qreal y)
- **PtCoords** (const QPointF &p)

6.44.1 Detailed Description

The PtCoords class describe positions with virtual point coordinates.

Theses coordinates describe each point relatively to the view. They correspond to a point in the image containing the entire map.

See also RlCoords, ClCoords and PxCoords

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

- src/editor/Game/mappainter.h

6.45 PxCoords Class Reference

The PxCoords class describe positions with real pixel coordinates.

```
#include <mappainter.h>
```

Public Member Functions

- **PxCoords** (qreal x, qreal y)
- **PxCoords** (const QPointF &p)
- **PxCoords** (const QPoint &p)
- **PxCoords** (int x, int y)

6.45.1 Detailed Description

The PxCoords class describe positions with real pixel coordinates.

Theses coordinates describe the pixel position.

See also RlCoords, ClCoords and PtCoords

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

- src/editor/Game/mappainter.h

6.46 RlCoords Class Reference

The RlCoords class describe positions with relative coordinates.

`#include <mappainter.h>`

Public Member Functions

- **RlCoords** (qreal x, qreal y)
- **RlCoords** (const QPointF &p)

6.46.1 Detailed Description

The RlCoords class describe positions with relative coordinates.

Theses coordinates have values in $[0, 1]$, for every point in the view.

See also ClCoords, PtCoords and PxCoords

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

- src/editor/Game/mappainter.h

6.47 src.server.Server Class Reference

Public Member Functions

- def **__init__**
- def **__del__**
- def **run**
- def **handle**
- def **handleEvent**

Public Attributes

- **net**
- **world**
- **actions**
- **persos**
- **orderDispatcher**
- **events**

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

- src/server.py

6.48 src.network.ServerConnection Class Reference

Inherits Thread.

Public Member Functions

- def **__init__**
- def **run**
- def **send**

Public Attributes

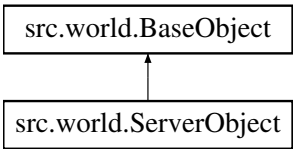
- **soc**
- **handle**

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

- src/network.py

6.49 src.world.ServerObject Class Reference

Inheritance diagram for src.world.ServerObject:



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

- src/world.py

6.50 TabAcces Class Reference

Signals

- void **activated** (int i)

Public Member Functions

- **TabAcces** (int i, const QString &n, const QPixmap &p, QWidget ∗parent=0)
- void **setActive** (bool a)

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

- src/editor/GUI/tabacces.h
- src/editor/GUI/tabacces.cpp

6.51 TabBar Class Reference

Public Slots

- void **setCurrentTab** (int t)

Signals

- void **currentTabChanged** (int)

Public Member Functions

- **TabBar** (QWidget ∗parent=0)
- void **addTabAcces** (const QString &n, const QPixmap &p)
- int **currentTab** () const
- void **setTabsEnabled** (bool e)

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

- src/editor/GUI/tabbar.h
- src/editor/GUI/tabbar.cpp

6.52 Welcome Class Reference

Public Member Functions

- **Welcome** (QWidget ∗parent=0)

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

- src/editor/GUI/Tabs/welcome.h
- src/editor/GUI/Tabs/welcome.cpp

6.53 src.world.World Class Reference

Public Member Functions

- def **__init__**

Public Attributes

- **maps**
- **entities**
- **objects**

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

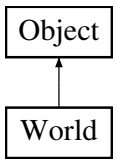
- src/world.py

6.54 World Class Reference

The World class is an Object.

`#include <game.h>`

Inheritance diagram for World:



Public Member Functions

- **World** (Game *g=nullptr)
- **ObjectsMap** (w, m, M, ap„ s) ObjectsMap(w

Public Attributes

- **c**
- **C**
- **ellType**

6.54.1 Detailed Description

The World class is an Object.

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

- src/editor/Game/game.h
- src/editor/Game/game.cpp

6.55 WorldEditor Class Reference

Public Member Functions

- **WorldEditor** (QWidget *parent=0)
- void **setGame** (Game *g)

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

- src/editor/GUI/Tabs/worldeditor.h
- src/editor/GUI/Tabs/worldeditor.cpp

6.56 XmlHandler Class Reference

Public Member Functions

- **XmlHandler** (Game *g)
- bool **startElement** (const QString &, const QString &localName, const QString &, const QXmlAttributes &atts)
- bool **endElement** (const QString &, const QString &localName, const QString &)

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

- src/editor/Game/xmlhandler.h
- src/editor/Game/xmlhandler.cpp

Chapter 7

File Documentation

7.1 src/editor/Game/mappainter.h File Reference

Definition of the class used to render maps.

```
#include "map.h"
```

Classes

- class RlCoords

  The RlCoords class describe positions with relative coordinates.
- class ClCoords

  The ClCoords class describe positions with cell coordinates.
- class PtCoords

  The PtCoords class describe positions with virtual point coordinates.
- class PxCoords

  The PxCoords class describe positions with real pixel coordinates.
- class MapPainter

  The MapPainter class that can paint a Map using a QPainter.

Defines

- #define **MINMAX**(a, x, b) std::min(std::max(a,x),b)

Functions

- MapPainter::Element operator| (MapPainter::Element a, MapPainter::Element b)

  The operator | is the flag OR operation.
- MapPainter::Element operator& (MapPainter::Element a, MapPainter::Element b)

  The operator & is the flag AND operation.

- MapPainter::Element operator^ (MapPainter::Element a, MapPainter::Element b)

  The operator ^ is the flag substraction operation.

7.1.1 Detailed Description

Definition of the class used to render maps. This file defines four types of coordinates : RlCoords, ClCoords, PtCoords and PxCoords. They all inherit from QPointF, and give a static type checking for the consistency of the coordinates which are used.

7.1.2 Function Documentation

7.1.2.1 MapPainter::Element operator^ (MapPainter::Element *a*, MapPainter::Element *b*) [inline]

The operator ^ is the flag substraction operation.

Warning

This is not a XOR operation, it corresponds to a&!b

7.2 src/editor/Game/object.h File Reference

This header defines the base class Object and Image.

```
#include <QtCore> #include <QtGui> #include <assert.h>
```

Classes

- class Object

  The Object class is the base class for every part of games.
- class Image

  The Image class stores an external file in a QImage, and gives each image ressources a unique identifier.

Defines

- #define Editing lastEdit = QDateTime::currentDateTime()
- #define Getter(name) inline int name() const{return params[#name];}
- #define ParamDef(name, value) params[#name] = value; Editing
- #define Param(name, Name)
- #define ParamObj(name, Name, pref)
- #define ObjectsMapC(name, names, Type, Types, pref, arg)
- #define ObjectsMap(pref, ini, Ini, body, sg, pl) ObjectsMapC(ini##body##sg, ini##body##pl, Ini##body##sg, Ini##body##pl, pref,ini)

7.2.1 Detailed Description

This header defines the base class Object and Image. Blablabla.

7.2.2 Define Documentation

7.2.2.1 #define Editing lastEdit = QDateTime::currentDateTime()

Facility to notify modifications.

7.2.2.2 #define Getter(*name*) inline int name() const{return params[#name];}

The macro Getter create a getter for the parameter name.

7.2.2.3 #define ObjectsMap(*pref, ini, Ini, body, sg, pl*) ObjectsMapC(ini##body##sg, ini##body##pl, Ini##body##sg, Ini##body##pl, pref,ini)

The macro ObjectsMap define the methods necessary to manipulate a set of objects

7.2.2.4 #define ObjectsMapC(*name, names, Type, Types, pref, arg*)

Value:

```
void add##Type(Type* arg){pref##Types[arg->ident()] = arg; Editing;} \
    void remove##Type(Type* arg){if(pref##Types.contains(arg->ident()))pref##
      Types.remove(arg->ident()); Editing;} \
    inline Type* name(int id) const{return pref##Types.contains(id) ? pref##
      Types[id] : nullptr;} \
    inline QList<Type*> names() const{return pref##Types.values();} \
private: \
    QMap<int, Type*> pref##Types; \
public:
```

Internal used only

7.2.2.5 #define Param(*name, Name*)

Value:

```
Getter(name) \
    inline void set##Name(int name##Value){ParamDef(name, name##Value);}
```

The macro Param defines the getter and setter functions for an object's parameter.

7.2.2.6 #define ParamDef(*name, value*) params[#name] = value; Editing

The macro ParamDef is a convenient way to modify a parameter

7.2.2.7 #define ParamObj(*name, Name, pref*)

Value:

```
Name* name() const{return pref##Name;} \
    void set##Name(Name* name##Obj){pref##Name = name##Obj; \
        ParamDef(name,name##Obj ? name##Obj->ident() : 0);} \
private: \
    Name* pref##Name; \
public:
```

The macro ParamObj defines the getter and setter functions for a member object of an Object.