

mkRPG

Generated by Doxygen 1.7.6.1

Mon Nov 14 2016 20:58:17

Contents

1	Todo List	1
2	Deprecated List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	13
6.1	File List	13
7	Namespace Documentation	15
7.1	src::parsing::map_parser Namespace Reference	15
7.1.1	Detailed Description	15
7.1.2	Function Documentation	15
7.1.2.1	get_size	15
7.1.2.2	map_parser	15
7.1.2.3	parse_cell	15
8	Class Documentation	17
8.1	src.path.Archi Class Reference	17
8.1.1	Detailed Description	17
8.1.2	Member Function Documentation	17

8.1.2.1	get_src_dir	17
8.1.2.2	get_src_file	18
8.1.2.3	get_static_dir	18
8.1.2.4	get_static_file	18
8.1.2.5	get_xml_dir	18
8.1.2.6	get_xml_file	18
8.1.2.7	list_files	18
8.2	src.backgroundLayer.BackgroundLayer Class Reference	18
8.3	src.world.BaseObject Class Reference	19
8.4	BColor Class Reference	20
8.4.1	Detailed Description	20
8.4.2	Constructor & Destructor Documentation	21
8.4.2.1	BColor	21
8.4.2.2	BColor	21
8.4.3	Member Function Documentation	21
8.4.3.1	color	21
8.4.3.2	colorChanged	21
8.4.3.3	name	21
8.4.3.4	nameChanged	21
8.4.3.5	setColor	22
8.4.3.6	setColorQuiet	22
8.4.3.7	setName	22
8.4.3.8	setNameQuiet	22
8.4.4	Property Documentation	22
8.4.4.1	color	22
8.4.4.2	name	23
8.5	BDock Class Reference	23
8.5.1	Detailed Description	24
8.6	BDocksZone Class Reference	24
8.6.1	Member Enumeration Documentation	25
8.6.1.1	ScrollBarMode	25
8.7	BDockWidget Class Reference	25
8.7.1	Detailed Description	26
8.8	BinaryStateMachine Class Reference	26

8.8.1 Detailed Description	27
8.9 BLayout Class Reference	27
8.10 src.world.Cell Class Reference	27
8.11 Cell Class Reference	28
8.11.1 Detailed Description	28
8.12 CellDock Class Reference	28
8.13 CellType Class Reference	29
8.13.1 Detailed Description	29
8.14 CellTypeListModel Class Reference	30
8.14.1 Detailed Description	30
8.15 CellTypesDock Class Reference	30
8.16 src.character.Character Class Reference	31
8.17 src.chunk.Chunk Class Reference	31
8.18 src.cache.ChunkCache Class Reference	32
8.19 CCoords Class Reference	33
8.19.1 Detailed Description	33
8.20 src.client.Client Class Reference	33
8.21 src.world.ClientObject Class Reference	34
8.22 Editor Class Reference	34
8.22.1 Detailed Description	35
8.23 src.world.Entity Class Reference	35
8.24 Game Class Reference	35
8.24.1 Detailed Description	36
8.24.2 Member Function Documentation	36
8.24.2.1 newIdent	36
8.25 GameObject Class Reference	36
8.25.1 Detailed Description	37
8.25.2 Constructor & Destructor Documentation	38
8.25.2.1 GameObject	38
8.25.3 Member Function Documentation	38
8.25.3.1 flags	38
8.25.3.2 getFlag	38
8.25.3.3 getParam	39
8.25.3.4 hasFlag	39

8.25.3.5	hasParam	39
8.25.3.6	ident	39
8.25.3.7	init	39
8.25.3.8	isValid	40
8.25.3.9	lastChildrenEdition	40
8.25.3.10	lastEdition	40
8.25.3.11	lastInternalEdition	40
8.25.3.12	params	40
8.25.3.13	setFlag	41
8.25.3.14	setParam	41
8.25.3.15	touch	41
8.26	src.cache.GlobalCache Class Reference	41
8.27	Image Class Reference	42
8.27.1	Detailed Description	42
8.27.2	Member Function Documentation	42
8.27.2.1	isValid	42
8.28	src.cache.ImageCache Class Reference	43
8.29	src.interactions.Interaction Class Reference	43
8.30	src.printWorld.Interface Class Reference	44
8.31	Intertie Class Reference	44
8.31.1	Detailed Description	45
8.32	src.layer.Layer Class Reference	45
8.33	src.world.Map Class Reference	45
8.33.1	Member Function Documentation	46
8.33.1.1	fill	46
8.34	Map Class Reference	46
8.34.1	Detailed Description	47
8.35	MapDock Class Reference	47
8.36	MapPainter Class Reference	47
8.36.1	Detailed Description	48
8.36.2	Member Enumeration Documentation	49
8.36.2.1	Element	49
8.36.3	Constructor & Destructor Documentation	49
8.36.3.1	MapPainter	49

8.36.3.2	MapPainter	49
8.36.4	Member Function Documentation	50
8.36.4.1	cooToPt	50
8.36.4.2	cooToPxl	50
8.36.4.3	hasHighlightedCell	50
8.36.4.4	highlightedCell	50
8.36.4.5	indToPt	50
8.36.4.6	isCell	50
8.36.4.7	mapSizeChanged	50
8.36.4.8	move	51
8.36.4.9	paint	51
8.36.4.10	preSelectedCellColor	51
8.36.4.11	ptToCoo	51
8.36.4.12	ptToPxl	51
8.36.4.13	pxlToCoo	51
8.36.4.14	pxlToPt	51
8.36.4.15	render	52
8.36.4.16	resize	52
8.36.4.17	resize	52
8.36.4.18	scale	52
8.36.4.19	selectedCellColor	52
8.36.4.20	setHighlightedCell	52
8.36.4.21	setHighlightedCell	53
8.36.4.22	setMap	53
8.36.4.23	setPaintedElement	53
8.36.4.24	setPaintedElements	53
8.36.4.25	setPreSelectedCellColor	53
8.36.4.26	setScale	53
8.36.4.27	setScaleDomain	54
8.36.4.28	setSelectedCellColor	54
8.36.4.29	setViewCenter	54
8.36.4.30	setViewCenter	54
8.36.4.31	setViewCenterQuiet	54
8.36.4.32	size	54

8.36.4.33	viewCenter	55
8.36.4.34	viewCenterChanged	55
8.36.4.35	virtualSize	55
8.36.4.36	zoom	55
8.37	MapsEditor Class Reference	55
8.37.1	Detailed Description	56
8.38	MapsListModel Class Reference	56
8.38.1	Detailed Description	56
8.39	MapViewer Class Reference	57
8.39.1	Detailed Description	57
8.39.2	Member Enumeration Documentation	57
8.39.2.1	SelectionMode	57
8.40	src.map.MapViewer Class Reference	58
8.40.1	Member Function Documentation	59
8.40.1.1	neighbors_chunk	59
8.41	src.network.NetworkClient Class Reference	59
8.42	src.networkudp.NetworkClient Class Reference	59
8.43	src.network.NetworkServer Class Reference	60
8.44	src.networkudp.NetworkServer Class Reference	61
8.45	NewGame Class Reference	61
8.46	Object Class Reference	62
8.46.1	Detailed Description	62
8.47	ObjectEditor Class Reference	62
8.48	ObjectFlagTableModel Class Reference	62
8.49	ObjectParamTableModel Class Reference	63
8.50	src.world.ObjectType Class Reference	63
8.51	Options Struct Reference	64
8.51.1	Detailed Description	64
8.51.2	Member Function Documentation	65
8.51.2.1	isAdjustable	65
8.51.2.2	load	65
8.51.2.3	options	66
8.51.2.4	reinitialise	66
8.51.2.5	save	66

8.51.2.6	setAdjustable	66
8.51.2.7	setDefault	66
8.52	src.orders.Order Class Reference	67
8.53	src.orders.OrderDispatcher Class Reference	67
8.53.1	Detailed Description	68
8.53.2	Member Function Documentation	68
8.53.2.1	treat	68
8.54	ParamItemDelegate Class Reference	68
8.55	src.tools.Perf Class Reference	68
8.55.1	Detailed Description	69
8.55.2	Member Function Documentation	69
8.55.2.1	show	69
8.55.2.2	tic	69
8.55.2.3	toc	69
8.56	PtCoords Class Reference	69
8.56.1	Detailed Description	70
8.57	PxCoords Class Reference	70
8.57.1	Detailed Description	70
8.58	RlCoords Class Reference	70
8.58.1	Detailed Description	71
8.59	src.cache.ScaledCache Class Reference	71
8.60	SelectionDock Class Reference	72
8.61	src.server.Server Class Reference	72
8.62	src.network.ServerConnection Class Reference	73
8.63	src.world.ServerObject Class Reference	73
8.64	TabAcces Class Reference	73
8.65	TabBar Class Reference	74
8.66	src.utils.WalkableGraph Class Reference	74
8.67	Welcome Class Reference	75
8.68	World Class Reference	75
8.68.1	Detailed Description	76
8.69	src.world.World Class Reference	76
8.70	WorldEditor Class Reference	76
8.71	src.printWorld.WorldViewer Class Reference	76

8.72	XmlHandler Class Reference	77
9	File Documentation	79
9.1	src/editor/Game/game.h File Reference	79
9.1.1	Detailed Description	79
9.2	src/editor/Game/map.h File Reference	79
9.2.1	Detailed Description	80
9.2.2	Define Documentation	80
9.2.2.1	forCells	80
9.3	src/editor/Game/mappainter.h File Reference	80
9.3.1	Detailed Description	81
9.3.2	Function Documentation	81
9.3.2.1	operator [^]	81
9.4	src/editor/Game/mapslistmodel.h File Reference	81
9.4.1	Detailed Description	82
9.5	src/editor/Game/object.h File Reference	82
9.5.1	Detailed Description	83
9.5.2	Define Documentation	84
9.5.2.1	Attr	84
9.5.2.2	AttrDef	85
9.5.2.3	AttrGetter	85
9.5.2.4	AttrSetter	86
9.5.2.5	AttrT	86
9.5.2.6	C	86
9.5.2.7	C0	87
9.5.2.8	C1	87
9.5.2.9	Flag	88
9.5.2.10	FlagGetter	88
9.5.2.11	FlagSetter	88
9.5.2.12	ObjectsMap	89
9.5.2.13	ObjectsMapC	89
9.5.2.14	Param	89
9.5.2.15	ParamGetter	90
9.5.2.16	ParamSetter	90

9.5.2.17	SetFlag	90
9.5.2.18	SetParam	91
9.6	src/editor/Game/xmlhandler.h File Reference	91
9.6.1	Detailed Description	92
9.7	src/editor/GUI/options.h File Reference	92
9.7.1	Detailed Description	93
9.7.2	Define Documentation	93
9.7.2.1	Default	93
9.7.2.2	DefaultF	93
9.7.2.3	DIR	93
9.7.2.4	MAP	93
9.7.2.5	WIN	93
9.8	src/editor/GUI/Tabs/bcolor.h File Reference	93
9.8.1	Detailed Description	94
9.9	src/editor/GUI/Tabs/Docks/bdock.h File Reference	94
9.9.1	Detailed Description	94

Chapter 1

Todo List

Class `GameObject`

Chapter 2

Deprecated List

Member [ObjectsMap](#) (pref, ini, Ini, body, sg, pl)

Member [ObjectsMapC](#) (name, names, Type, Types, pref, arg)

Chapter 3

Namespace Index

3.1 Namespace List

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

[src::parsing::map_parser](#) 15

Chapter 4

Class Index

4.1 Class Hierarchy

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

src.path.Archi	17
src.backgroundLayer.BackgroundLayer	18
src.world.BaseObject	19
src.world.ClientObject	34
src.world.ServerObject	73
BColor	20
BDock	23
BDocksZone	24
BDockWidget	25
CellDock	28
CellTypesDock	30
MapDock	47
SelectionDock	72
BinaryStateMachine	26
BLayout	27
src.world.Cell	27
CellTypeListModel	30
src.character.Character	31
src.chunk.Chunk	31
CICoords	33
src.client.Client	33
Editor	34
src.world.Entity	35
GameObject	36
Cell	28
CellType	29
Game	35
Image	42

Map	46
Object	62
World	75
src.cache.GlobalCache	41
src.cache.ScaledCache	71
src.cache.ChunkCache	32
src.cache.ImageCache	43
src.interactions.Interaction	43
src.printWorld.Interface	44
src.printWorld.WorldViewer	76
Intertie	44
src.layer.Layer	45
src.world.Map	45
MapPainter	47
MapsEditor	55
MapsListModel	56
MapView	57
src.map.MapViewer	58
src.network.NetworkClient	59
src.networkudp.NetworkClient	59
src.network.NetworkServer	60
src.networkudp.NetworkServer	61
NewGame	61
ObjectEditor	62
ObjectFlagTableModel	62
ObjectParamTableModel	63
src.world.ObjectType	63
Options	64
src.orders.Order	67
src.orders.OrderDispatcher	67
ParamItemDelegate	68
src.tools.Perf	68
PtCoords	69
PxCoords	70
RlCoords	70
src.server.Server	72
src.network.ServerConnection	73
TabAcces	73
TabBar	74
src.utils.WalkableGraph	74
Welcome	75
src.world.World	76
WorldEditor	76
XmlHandler	77

Chapter 5

Class Index

5.1 Class List

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

src.path.Archi	17
src.backgroundLayer.BackgroundLayer	18
src.world.BaseObject	19
BColor	
Simple frame that offers color selection	20
BDock	
Container for widget to display in a BDocksZone	23
BDocksZone	24
BDockWidget	
Base class for game-related docks	25
BinaryStateMachine	
Simple QStateMachine with two states	26
BLayout	27
src.world.Cell	27
Cell	
The Cell class	28
CellDock	28
CellType	
The CellType class	29
CellTypeListModel	
The CellTypeListModel class	30
CellTypesDock	30
src.character.Character	31
src.chunk.Chunk	31
src.cache.ChunkCache	32
CICoords	
Describe positions with cell coordinates	33
src.client.Client	33
src.world.ClientObject	34

Editor	
Main window of the Editor	34
src.world.Entity	35
Game	
Gather the differents parts needed to describe a game	35
GameObject	
Base class for every part of games	36
src.cache.GlobalCache	41
Image	
Stores an external file in a QImage, and gives each image ressources a unique identifier	42
src.cache.ImageCache	43
src.interactions.Interaction	43
src.printWorld.Interface	44
Intertie	
Provide int that move smoothly from their value to an objective	44
src.layer.Layer	45
src.world.Map	45
Map	
The Map class	46
MapDock	47
MapPainter	
That can paint a Map using a QPainter	47
MapsEditor	
Tab offering map editing facilities	55
MapsListModel	
Presentation class for the Qt Model-View framework	56
MapView	
Widget to display and edit a Map using a MapPainter	57
src.map.MapViewer	58
src.network.NetworkClient	59
src.networkudp.NetworkClient	59
src.network.NetworkServer	60
src.networkudp.NetworkServer	61
NewGame	61
Object	
The Object class	62
ObjectEditor	62
ObjectFlagTableModel	62
ObjectParamTableModel	63
src.world.ObjectType	63
Options	
Session-independant options and preferences	64
src.orders.Order	67
src.orders.OrderDispatcher	67
ParamItemDelegate	68
src.tools.Perf	68
PtCoords	
Describe positions with virtual point coordinates	69

PxCoords	
Describe positions with real pixel coordinates	70
RICoords	
Describe positions with relative coordinates	70
src.cache.ScaledCache	71
SelectionDock	72
src.server.Server	72
src.network.ServerConnection	73
src.world.ServerObject	73
TabAcces	73
TabBar	74
src.utils.WalkableGraph	74
Welcome	75
World	
The World class	75
src.world.World	76
WorldEditor	76
src.printWorld.WorldViewer	76
XmlHandler	77

Chapter 6

File Index

6.1 File List

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

src/editor/Game/ game.h	
Definition of the Game and World classes	79
src/editor/Game/ map.h	
Definition of the Map , Cell and CellType classes	79
src/editor/Game/ mappainter.h	
Definition of the MapPainter class and other related classes to render maps	80
src/editor/Game/ mapslistmodel.h	
Definition of Model/View presentation classes	81
src/editor/Game/ object.h	
Definition of the base class GameObject , and some inherited classes	82
src/editor/Game/ xmlhandler.h	
Definition of the XmlHandler class and other related classes to read XML game's files	91
src/editor/GUI/ editor.h	??
src/editor/GUI/ newgame.h	??
src/editor/GUI/ options.h	
Definition of the Options class, and the constants that are used in this class	92
src/editor/GUI/ tabacces.h	??
src/editor/GUI/ tabbar.h	??
src/editor/GUI/Tabs/ bcolor.h	
Definition of the BColor class	93
src/editor/GUI/Tabs/ celldock.h	??
src/editor/GUI/Tabs/ celltypesdock.h	??
src/editor/GUI/Tabs/ mapdock.h	??
src/editor/GUI/Tabs/ mapseditor.h	??
src/editor/GUI/Tabs/ mapviewer.h	??
src/editor/GUI/Tabs/ objecteditor.h	??

src/editor/GUI/Tabs/ selectiondock.h	??
src/editor/GUI/Tabs/ welcome.h	??
src/editor/GUI/Tabs/ worldeditor.h	??
src/editor/GUI/Tabs/Docks/ bdock.h	
Definition of the BDock class	94
src/editor/GUI/Tabs/Docks/ bdockszone.h	??
src/editor/GUI/Tabs/Docks/ bdockwidget.h	??
src/editor/GUI/Tabs/Docks/ intertie.h	??

Chapter 7

Namespace Documentation

7.1 `src::parsing::map_parser` Namespace Reference

Functions

- def [parse_cell](#)
- def [map_parser](#)
- def [get_size](#)
- def **gen_map**

7.1.1 Detailed Description

This module handles xml parsing for maps description files.

7.1.2 Function Documentation

7.1.2.1 `def src.parsing.map_parser.get_size (tree)`

Gets the size of the map.

7.1.2.2 `def src.parsing.map_parser.map_parser (map_xml)`

The main parser for the map xml file.

7.1.2.3 `def src.parsing.map_parser.parse_cell (cell_object)`

Parses a CellType attribute.

Chapter 8

Class Documentation

8.1 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`

8.1.1 Detailed Description

This class manages the architecture 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

8.1.2 Member Function Documentation

8.1.2.1 `def src.path.Archi.get_src_dir (self, dir_path)`

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

8.1.2.2 `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.

8.1.2.3 `def src.path.Archi.get_static_dir (self, dir_path)`

Gets the given dir_path with respect to the static folder.

8.1.2.4 `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

8.1.2.5 `def src.path.Archi.get_xml_dir (self, dir_path)`

Gets the given dir_path with respect to the xml folder.

8.1.2.6 `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.

8.1.2.7 `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

8.2 src.backgroundLayer.BackgroundLayer Class Reference

Inherits Layer.

Public Member Functions

- `def __init__`
- `def init_layer`
- `def render`

- def **collision_test**
- def **click_update**
- def **get_grid_info**
- def **zoom**

Public Attributes

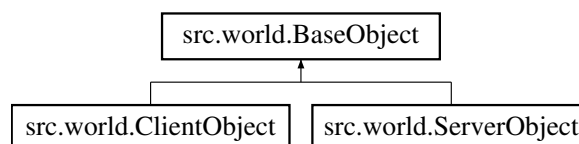
- **cell_ids**
- **grid_cell_ids**
- **g_width**
- **g_height**
- **image**
- **rect**
- **mouse_iso**
- **selected_cell**

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

- src/backgroundLayer.py

8.3 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

8.4 BColor Class Reference

The [BColor](#) class is a simple frame that offers color selection.

```
#include <bcolor.h>
```

Public Slots

- void [setColor](#) (const QColor &c)
- void [setColorQuiet](#) (const QColor &c)
- void [setName](#) (const QString &s)
- void [setNameQuiet](#) (const QString &s)

Signals

- void [colorChanged](#) (const QColor &)
- void [nameChanged](#) (const QString &)

Public Member Functions

- [BColor](#) (QWidget *parent=0)
- [BColor](#) (QColor c, QWidget *parent=0)
- const QString & [name](#) () const
- const QColor & [color](#) () const

Properties

- QColor [color](#)
- QString [name](#)

8.4.1 Detailed Description

The [BColor](#) class is a simple frame that offers color selection.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 BColor::BColor (QWidget * *parent* = 0) [explicit]

Constructs a new [BColor](#) object, with white as current color.

8.4.2.2 BColor::BColor (QColor *c*, QWidget * *parent* = 0) [explicit]

Constructs a new [BColor](#) object and sets the color to *c*.

8.4.3 Member Function Documentation

8.4.3.1 const QColor& BColor::color () const

Returns the current color of the selector.

See also

[setColor](#), [setColorQuiet](#), [colorChanged](#)

8.4.3.2 void BColor::colorChanged (const QColor &) [signal]

This signal is emitted when the color change, both when the user edit it or when [setColor](#) is called.

See also

[color](#), [setColorQuiet](#)

8.4.3.3 const QString& BColor::name () const

Returns the name of the selector.

See also

[setName](#), [setNameQuiet](#), [nameChanged](#)

8.4.3.4 void BColor::nameChanged (const QString &) [signal]

This signal is emitted when the name change, when [setColor](#) is called.

See also

[name](#), [setNameQuiet](#)

8.4.3.5 void BColor::setColor (const QColor & c) [slot]

Sets the current color.

The signal [colorChanged](#) is emitted.

See also

[setColorQuiet](#), [color](#)

8.4.3.6 void BColor::setColorQuiet (const QColor & c) [slot]

Sets the current color.

The signal [colorChanged](#) is not emitted.

See also

[setColor](#), [color](#)

8.4.3.7 void BColor::setName (const QString & s) [slot]

Sets the name of the selector.

The signal [nameChanged](#) is emitted.

See also

[setNameQuiet](#), [name](#)

8.4.3.8 void BColor::setNameQuiet (const QString & s) [slot]

Sets the name of the selector.

The signal [nameChanged](#) is not emitted.

See also

[setName](#), [name](#)

8.4.4 Property Documentation

8.4.4.1 const QColor & BColor::color [read, write]

The current color that is displayed by the widget

See also

[setColor](#), [setColorQuiet](#), [colorChanged](#).

8.4.4.2 const QString & BColor::name [read, write]

The name that is shown as title for the color chooser dialog used for user color definition purpose.

see also [setName](#), [setNameQuiet](#), and [nameChanged](#).

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

- [src/editor/GUI/Tabs/bcolor.h](#)
- [src/editor/GUI/Tabs/bcolor.cpp](#)

8.5 BDock Class Reference

The [BDock](#) class is the container for widget to display in a [BDocksZone](#).

```
#include <bdock.h>
```

Public Slots

- void **setTitle** (QString s)
- void **setUnfold** (bool v)

Signals

- void **mouseClick** (int i, const QPoint &p)
- void **mouseMove** (int i, const QPoint &p)
- void **mouseRelease** (int i, const QPoint &p)
- void **movementFinished** (int i)

Public Member Functions

- **BDock** (QString title, [BDockWidget](#) *dock, QWidget *parent=0)
- bool **unfold** () const
- int **currentSize** () const
- void **setCurrentSize** (int t)
- void **setIndex** (int i)
- int **index** () const
- void **moveTo** (int i, bool inert=true)
- void **setLength** (int l)

Properties

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

8.5.1 Detailed Description

The [BDock](#) class is the container for widget to display in a [BDocksZone](#).

A [BDock](#) is composed of a title and a [QScrollArea](#) in which a [BDockWidget](#) is displayed.

This container is movable within the [BDocksZone](#) it belongs to, and it can be hide.

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](#)

8.6 BDocksZone Class Reference

Public Types

- enum [ScrollBarMode](#) { [AlwaysVisible](#), [Adjustable](#), [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**

8.6.1 Member Enumeration Documentation

8.6.1.1 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 useless

Adjustable 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:

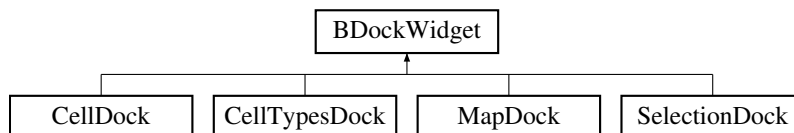
- `src/editor/GUI/Tabs/Docks/bdockszone.h`
- `src/editor/GUI/Tabs/Docks/bdockszone.cpp`

8.7 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**

8.7.1 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

8.8 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

8.8.1 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`

8.9 BLayout Class Reference

Signals

- void **sizeChanged** (int)
- void **showPoint** (int, 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`

8.10 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:

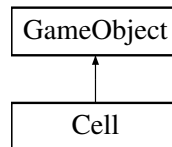
- `src/world.py`

8.11 Cell Class Reference

The [Cell](#) class.

```
#include <map.h>
```

Inheritance diagram for Cell:



Public Member Functions

- **Cell** ([Game](#) *g=nullptr, [GameObject](#) *parent=nullptr)
- bool **isSelected** () const
- void **setSelected** (bool s=true)
- void **invertSelected** ()
- void **addSelection** ()
- bool **isPreSelected** () const
- void **confirmPreSelection** (bool add=true)
- void **clearPreSelection** ()

Public Attributes

- ObjectListD(o, O, bject,, s, [Object](#)) private int **nbSel**
- bool **selectMod**

8.11.1 Detailed Description

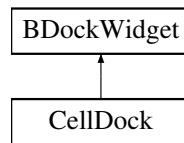
The [Cell](#) class.

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

- src/editor/Game/[map.h](#)
- src/editor/Game/map.cpp

8.12 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:

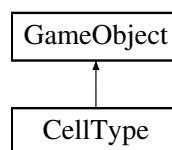
- src/editor/GUI/Tabs/celldock.h
- src/editor/GUI/Tabs/celldock.cpp

8.13 CellType Class Reference

The [CellType](#) class.

```
#include <map.h>
```

Inheritance diagram for CellType:



Public Member Functions

- **CellType** ([Game](#) *g, [GameObject](#) *parent)

8.13.1 Detailed Description

The [CellType](#) class.

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

- [src/editor/Game/map.h](#)
- [src/editor/Game/map.cpp](#)

8.14 CellTypeListModel Class Reference

The [CellTypeListModel](#) class.

```
#include <mapslistmodel.h>
```

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

8.14.1 Detailed Description

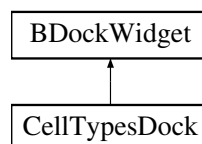
The [CellTypeListModel](#) class.

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

- [src/editor/Game/mapslistmodel.h](#)
- [src/editor/Game/mapslistmodel.cpp](#)

8.15 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

8.16 src.character.Character Class Reference

Public Member Functions

- def `__init__`
- def `render`
- def `update`
- def `update_skin`
- def `zoom`
- def `set_path`
- def `make_skin`
- def `get_cell_pos_by_index`
- def `move`

Public Attributes

- `skin`
- `name`
- `action`
- `scale`
- `orientation`
- `image`
- `current_image`
- `game_frame_count`
- `anim_frame_count`
- `current_cell`
- `path`
- `pos_offset`

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

- src/character.py

8.17 src.chunk.Chunk Class Reference

Public Member Functions

- def `__init__`

- def **init_chunk**
- def **render**
- def **scale_chunk**
- def **update**
- def **click_trigger**
- def **set_state**
- def **get_state**

Public Attributes

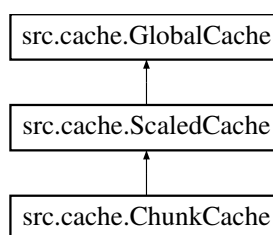
- **index**
- **cells**
- **g_width**
- **g_height**
- **scale**
- **width**
- **height**
- **pos**
- **rect**
- **layers**
- **image**

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

- `src/chunk.py`

8.18 `src.cache.ChunkCache` Class Reference

Inheritance diagram for `src.cache.ChunkCache`:



Public Member Functions

- def **init_chunk**
- def **init_elts**
- def **init_chunks**
- def **get_chunk**
- def **add_scaled**

Static Public Attributes

- dictionary **cache** = {}

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

- `src/cache.py`

8.19 CCoords Class Reference

The [CCoords](#) class describe positions with cell coordinates.

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

Public Member Functions

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

8.19.1 Detailed Description

The [CCoords](#) 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

[RCoords](#), [PtCoords](#), [PxCoords](#)

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

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

8.20 src.client.Client Class Reference

Public Member Functions

- def **__init__**
- def **__del__**
- def **run**
- def **frame_counter**
- def **update_view**
- def **get_conf_file**
- def **get_conf**
- def **init_cache**
- def **handleOrder**

Public Attributes

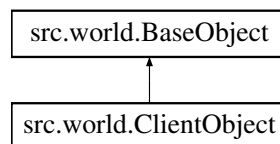
- **net**
- **screen_size**
- **screen**
- **world**
- **interface**
- **interactions**
- **perso**
- **orderDispatcher**
- **background**
- **conf**

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

- `src/client.py`

8.21 `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`

8.22 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)

8.22.1 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

8.23 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:

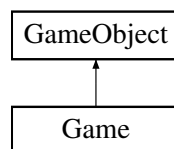
- src/world.py

8.24 Game Class Reference

The [Game](#) class gather the differents parts needed to describe a game.

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

Inheritance diagram for Game:



Public Member Functions

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

8.24.1 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)

8.24.2 Member Function Documentation

8.24.2.1 `int Game::newIdent () [inline]`

Returns a new unused identifiers

Note

It should only be used by [GameObject](#) methods [GameObject::init](#) and [GameObject::GameObject](#).

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

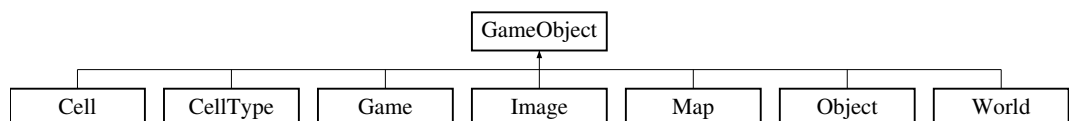
- [src/editor/Game/game.h](#)
- [src/editor/Game/game.cpp](#)

8.25 GameObject Class Reference

The [GameObject](#) class is the base class for every part of games.

```
#include <object.h>
```

Inheritance diagram for [GameObject](#):



Public Member Functions

- [GameObject](#) ([Game](#) *g=nullptr, [GameObject](#) *parent=nullptr)
- void [init](#) ([Game](#) *g, [GameObject](#) *p)
- virtual bool [isValid](#) () const
- int [ident](#) () const
- const QDateTime & [lastInternalEdition](#) () const
- const QDateTime & [lastChildrenEdition](#) () const
- const QDateTime & [lastEdition](#) () const

- int `getParam` (const QString &p) const
- void `setParam` (const QString &p, int v)
- bool `hasParam` (const QString &p) const
- QList< QString > `params` () const
- bool `getFlag` (const QString &f) const
- void `setFlag` (const QString &f, bool v)
- bool `hasFlag` (const QString &f) const
- QList< QString > `flags` () const
- void `touch` ()
- void `addReference` ()
- void `removeReference` ()
- void `setParent` (GameObject *p)

Protected Member Functions

- void `addChild` (GameObject *c)
- void `removeChild` (GameObject *c)
- void `childrenTouched` (const QDateTime &d)

Protected Attributes

- GameObject * `parent`
- QMap< int, GameObject * > `children`
- Game * `game`
- int `id`
- int `nbRef`
- QMap< QString, int > `aParams`
- QMap< QString, bool > `aFlags`
- QString `fileName`
- QDateTime `lastEdit`
- QDateTime `lastChildEdit`

8.25.1 Detailed Description

The `GameObject` class is the base class for every part of games.

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

Object edition notification mechanism

To make the edition easier, each `GameObject` contains two QDateTime values :

- The most recent edition time, which is updated by the `touch` method
- The most recent child edition time, also updated by the `touch` method

Note

If the changes that are made in the object have to be detected by display/edition widgets, the [touch](#) function should be called.

To prevent the notification chain to be broken, the existing objects should always have a parent (except for the root object). This can be achieved using the [init](#) or `setParent` method, when the parent have not been given in the constructor. (see [object.h](#) for details)

References count

Todo**8.25.2 Constructor & Destructor Documentation**

8.25.2.1 `GameObject::GameObject (Game * g = nullptr, GameObject * parent = nullptr)`

Constructs a new [GameObject](#) with parent `parent` and the reference to the game `g`.

Note

If these objects cannot be given to the constructor (case of an array of objects), the [init](#) method must be called after the creation to make the [GameObject](#) valid.

8.25.3 Member Function Documentation

8.25.3.1 `QList<QString> GameObject::flags () const [inline]`

Returns the list of the registered flags

See also

[getFlag](#), [setFlag](#), [params](#)

8.25.3.2 `bool GameObject::getFlag (const QString & f) const [inline]`

Returns the value of the `f` flag.

Note

If the requested parameter does not exists, a `false` value is returned, and the flags map stay unchanged

See also

[flags](#), [hasFlag](#), [setFlag](#), [getParam](#)

8.25.3.3 `int GameObject::getParam (const QString & p) const` `[inline]`

Returns the value of the `p` parameter.

Note

If the requested parameter does not exists, a null value is returned, and the parameters map stay unchanged

See also

[params](#), [hasParam](#), [setParam](#), [getFlag](#)

8.25.3.4 `bool GameObject::hasFlag (const QString & f) const` `[inline]`

Returns true if the flag `f` is register in the object's flags.

See also

[getFlag](#), [setFlag](#), [hasParam](#)

8.25.3.5 `bool GameObject::hasParam (const QString & p) const` `[inline]`

Returns true if the parameter `is` register in the object's parameters.

See also

[getParam](#), [setParam](#), [hasFlag](#)

8.25.3.6 `int GameObject::ident () const` `[inline]`

Returns the name wide unique identifier of the object.

See also

[init](#), [GameObject](#)

8.25.3.7 `void GameObject::init (Game * g, GameObject * p)`

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

See also

[isValid](#), [GameObject](#)

8.25.3.8 `virtual bool GameObject::isValid () const` `[inline, virtual]`

Returns true if the object has been initialised

See also

[init](#), [GameObject](#)

Reimplemented in [Image](#).

8.25.3.9 `const QDateTime& GameObject::lastChildrenEdition () const` `[inline]`

Returns the last time one of the object's children has been modified.

See also

[lastEdition](#), [lastInternalEdition](#)

8.25.3.10 `const QDateTime& GameObject::lastEdition () const` `[inline]`

Returns the last time a modification was made on the object or one of its children.

See also

[lastInternalEdition](#), [lastChildrenEdition](#)

8.25.3.11 `const QDateTime& GameObject::lastInternalEdition () const` `[inline]`

Returns the last edition time.

See also

[lastEdition](#), [lastChildrenEdition](#)

8.25.3.12 `QList<QString> GameObject::params () const` `[inline]`

Returns the list of the registered paramters

See also

[getParam](#), [setParam](#), [flags](#)

8.25.3.13 void **GameObject::setFlag** (const QString & *f*, bool *v*) [inline]

Set the value of the *f* flag.

Note

If the requested flag does not exists, it is created.

See also

[flags](#), [hasFlag](#), [getFlag](#), [setParam](#)

8.25.3.14 void **GameObject::setParam** (const QString & *p*, int *v*) [inline]

Set the value of the *p* parameter.

Note

If the requested parameter does not exists, it is created.

See also

[params](#), [hasParam](#), [getParam](#), [setFlag](#)

8.25.3.15 void **GameObject::touch** ()

Notify the object and its parent that it has been modified.

See also

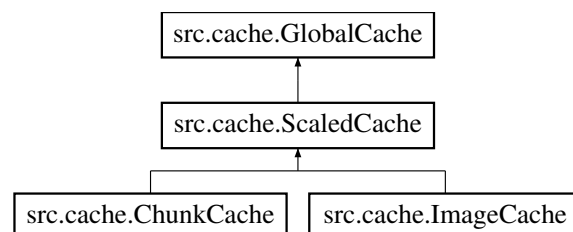
[lastInternalEdition](#), [lastChildrenEdition](#), [lastEdition](#).

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

- [src/editor/Game/object.h](#)
- [src/editor/Game/object.cpp](#)

8.26 src.cache.GlobalCache Class Reference

Inheritance diagram for src.cache.GlobalCache:



Public Member Functions

- def `__init__`
- def `set`
- def `get`
- def `clear`
- def `keys`
- def `show`

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

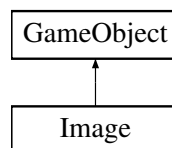
- `src/cache.py`

8.27 Image Class Reference

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

```
#include <object.h>
```

Inheritance diagram for `Image`:



Public Member Functions

- `Image (Game *g, GameObject *parent, const QString &fileName)`
- `bool isValid () const`
- `const QImage & image () const`
- `const QSize size () const`
- `void update ()`

8.27.1 Detailed Description

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

8.27.2 Member Function Documentation

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

Returns true if the object has been initialised

See also

[init](#), [GameObject](#)

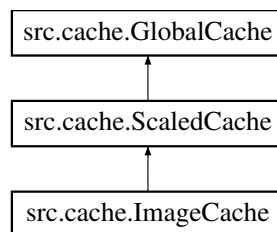
Reimplemented from [GameObject](#).

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

- [src/editor/Game/object.h](#)
- [src/editor/Game/object.cpp](#)

8.28 src.cache.ImageCache Class Reference

Inheritance diagram for src.cache.ImageCache:



Public Member Functions

- def **init_image_from_file**
- def **init_image_from_surface**
- def **get_image**
- def **init_elts**
- def **init_images**
- def **add_scaled**

Static Public Attributes

- dictionary **cache** = {}

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

- [src/cache.py](#)

8.29 src.interactions.Interaction Class Reference

Public Member Functions

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

Public Attributes

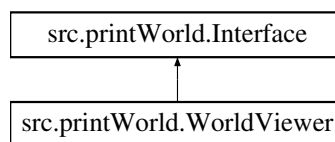
- **target**
- **type**
- **key**
- **event**

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

- `src/interactions.py`

8.30 `src.printWorld.Interface` Class Reference

Inheritance diagram for `src.printWorld.Interface`:



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

- `src/printWorld.py`

8.31 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)

8.31.1 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

8.32 src.layer.Layer Class Reference

Public Member Functions

- def **__init__**
- def **render**
- def **update**
- def **get_cell_pos**
- def **make_grid**
- def **update_cell**
- def **update_grid**
- def **zoom**

Public Attributes

- **cells**
- **scale**
- **g_width**
- **g_height**
- **size**

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

- src/layer.py

8.33 src.world.Map Class Reference

Public Member Functions

- def **__init__**
- def **fill**

Public Attributes

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

8.33.1 Member Function Documentation

8.33.1.1 `def src.world.Map.fill (self)`

Complète les cases par défaut

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

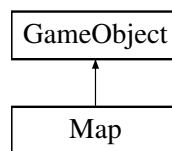
- `src/world.py`

8.34 Map Class Reference

The [Map](#) class.

```
#include <map.h>
```

Inheritance diagram for Map:



Public Member Functions

- **Map** ([Game](#) *g, [GameObject](#) *parent)
- **ParamGetter** (width) ParamGetter(height) QSize size() const
- void **setWidth** (int w)
- void **setHeight** (int h)
- void **resize** (int w, int h)
- [Cell](#) & **cell** (int i, int j) const
- [Cell](#) & **cell** (const QPoint &p) const
- void **selectAll** ()
- void **unselectAll** ()
- void **confirmPreSelection** (bool add=true)
- void **clearPreSelection** ()

8.34.1 Detailed Description

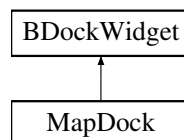
The [Map](#) class.

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

- [src/editor/Game/map.h](#)
- [src/editor/Game/map.cpp](#)

8.35 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](#)

8.36 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 describes 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](#) ()
- [RICoords](#) [viewCenter](#) () const
- void [setViewCenter](#) ([RICoords](#) 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 [CICoords](#) &p)
- bool [setHighlightedCell](#) (int i, int j)
- QPoint [highlightedCell](#) () const
- bool [hasHighlightedCell](#) () const
- bool [isCell](#) (const [CICoords](#) &c) 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](#) ([CICoords](#) p) const
- [CICoords](#) [ptToCoo](#) ([PtCoords](#) p) const
- [PxCoords](#) [cooToPxl](#) ([CICoords](#) p) const
- [CICoords](#) [pxlToCoo](#) ([PxCoords](#) p) const
- [PtCoords](#) [indToPt](#) (int i, int j) const
- const QColor & [selectedCellColor](#) () const
- const QColor & [preSelectedCellColor](#) () const
- void [setSelectedCellColor](#) (const QColor &c)
- void [setPreSelectedCellColor](#) (const QColor &c)

8.36.1 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 : [RICoords](#), [CICoords](#), [PtCoords](#) and [PxCoords](#)

8.36.2 Member Enumeration Documentation

8.36.2.1 enum `MapPainter::Element`

The `Element` enum describes 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 background 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

8.36.3 Constructor & Destructor Documentation

8.36.3.1 `MapPainter::MapPainter (QObject * parent = 0)`

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

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

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

8.36.4 Member Function Documentation

8.36.4.1 `PtCoords MapPainter::cooToPt (CCoords p) const`

Converts cells indice to virtual point coordinates

8.36.4.2 `PxCoords MapPainter::cooToPxl (CCoords p) const`

Convenient function equivalent to `ptToPxl(cooToPt(p))`

8.36.4.3 `bool MapPainter::hasHighlightedCell () const`

Returns true if a [cell](#) is highligthed.

See also

[highlightedCell](#), [setHighlightedCell](#)

8.36.4.4 `QPoint MapPainter::highlightedCell () const`

Returns the integer index of the [cell](#) the is highlighted.

See also

[setHighlightedCell](#), [hasHighlightedCell](#)

8.36.4.5 `PtCoords MapPainter::indToPt (int i, int j) const`

Converts to coordinates

8.36.4.6 `bool MapPainter::isCell (const CCoords & c) const`

Returns true if the coordinate `c` correspond to a [cell](#).

8.36.4.7 `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, ...).

8.36.4.8 `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](#)

8.36.4.9 `void MapPainter::paint (QPainter & p)`

Draws the map in the QPaintDevice.

See also

[render](#)

8.36.4.10 `const QColor & MapPainter::preSelectedCellColor () const`

Returns the color of the filter that is applied to pre-selected cells.

See also

[setPreSelectedCellColor](#), [selectedCellColor](#)

8.36.4.11 `CICoords MapPainter::ptToCoo (PtCoords p) const`

Converts virtual point to cell indice

8.36.4.12 `PxCoords MapPainter::ptToPxl (PtCoords p) const`

Converts virtual point to real pixel coordinates

8.36.4.13 `CICoords MapPainter::pxlToCoo (PxCoords p) const`

Convenient function equivalent to [ptToCoo](#)([pxlToPt](#)(*p*))

8.36.4.14 `PtCoords MapPainter::pxlToPt (PxCoords p) const`

Converts real pixel to virtual point coordinates

8.36.4.15 `const QImage & MapPainter::render ()`

Provides a QImage with a view of the map.

See also

[paint](#)

8.36.4.16 `void MapPainter::resize (QSize s)`

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

See also

[size](#)

8.36.4.17 `void MapPainter::resize (int wi, int he)`

This is an overload function, see [resize](#)

8.36.4.18 `double MapPainter::scale () const`

Returns the current scale of the view.

See also

[setScale](#)

8.36.4.19 `const QColor & MapPainter::selectedCellColor () const`

Returns the color of the filter that is applied to selected cells.

See also

[setSelectedCellColor](#), [preSelectedCellColor](#)

8.36.4.20 `bool MapPainter::setHighlightedCell (const CIColors & p)`

Sets the highlighted [cell](#) to the one at the [CIColors](#) p

See also

[highlightedCell](#), [hasHighlightedCell](#)

8.36.4.21 `bool MapPainter::setHighlightedCell (int i, int j)`

This is an overload function, see [setViewCenter](#).

8.36.4.22 `void MapPainter::setMap (Map * m)`

Loads the [map](#), computing the new size of the view area.

8.36.4.23 `void MapPainter::setPaintedElement (MapPainter::Element e, bool painte
=true)`

Enables or disables the render of an [element](#).

See also

[setPaintedElements](#)

8.36.4.24 `void MapPainter::setPaintedElements (Element e)`

Sets the rendered [elements](#).

See also

[setPaintedElement](#)

8.36.4.25 `void MapPainter::setPreSelectedCellColor (const QColor & c)`

Sets the color of the filter that is applied to pre-selected cells.

See also

[preSelectedCellColor](#), [setSelectedCellColor](#)

8.36.4.26 `void MapPainter::setScale (double scale)`

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

See also

[scale](#), [setScaleDomain](#)

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

Sets the valid values for the scale.

See also

[scale](#), [setScale](#)

8.36.4.28 void MapPainter::setSelectedCellColor (const QColor & c)

Sets the color of the filter that is applied to selected cells.

See also

[selectedCellColor](#), [setPreSelectedCellColor](#)

8.36.4.29 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](#)

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

This is an overload function, see [setViewCenter](#).

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

does the same as [setViewCenter](#), without emitting the signal `viewCenterChanged` to avoid event loop.

8.36.4.32 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](#)

8.36.4.33 `RICoords MapPainter::viewCenter () const`

Return the relative coordinates of the current view center.

See also

[setViewCenter](#)

8.36.4.34 `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.

8.36.4.35 `QSize MapPainter::virtualSize () const`

Computes the total size of the image of the map

8.36.4.36 `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 particular 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`

8.37 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)

8.37.1 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

8.38 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

8.38.1 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

8.39 MapViewer Class Reference

The [MapViewer](#) class provides a widget to display and edit a [Map](#) using a [MapPainter](#).

```
#include <mapviewer.h>
```

Public Types

- enum [SelectionMode](#) { [PencilSelection](#), [RectangleSelection](#), [RegionSelection](#) }
The SelectionMode enum describes the different behaviour the cell selection can have.

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** ()
- void **setSelectionMode** ([SelectionMode](#) m)
- [SelectionMode](#) **selectionMode** () const

8.39.1 Detailed Description

The [MapViewer](#) class provides a widget to display and edit a [Map](#) using a [MapPainter](#).

Several [selection modes](#) are available. Combined with the “Ctrl” and “Shift” modifiers, a colossal amount of selection possibilities is offered. See [SelectionMode](#) for more information.

8.39.2 Member Enumeration Documentation

8.39.2.1 enum MapViewer::SelectionMode

The SelectionMode enum describes the different behaviour the cell selection can have.

The selection's behaviour is based on two parameters :

- The keyboard modifiers that are pressed during selection.
- The current selection mode

If the “Ctrl” modifier is pressed, the past selected cells stay selected otherwise, they are all unselected

If the “Shift” modifier is pressed, the selection is inverted.

Three modes of selection exists :

Enumerator:

PencilSelection The [cells](#) under the cursor are selected

RectangleSelection The [cells](#) inside the rectangle defined by the clicked cell and the cell under the cursor are selected

RegionSelection The [cells](#) inside the region drawn by cursor’s moves are selected

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

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

8.40 src.map.MapViewer Class Reference

Public Member Functions

- def **__init__**
- def **load_chunks**
- def **make_walkables**
- def **zoom**
- def **move**
- def **render**
- def [neighbors_chunk](#)
- def **onscreen_chunks**
- def **update**
- def **propagate_trigger**
- def **compute_path**
- def **load_bg**

Public Attributes

- **map**
- **world**
- **scale**
- **cm_height**

- **width**
- **height**
- **walkablesGraph**
- **current_chunk**
- **pos_offset**
- **chunk_pos**
- **chunks_state**

8.40.1 Member Function Documentation

8.40.1.1 def src.map.MapViewer.neighbors_chunk (self, chunk)

Returns indexes of the neighbors of chunk and index of chunk

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

- src/map.py

8.41 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

8.42 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

8.43 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

8.44 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

8.45 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:

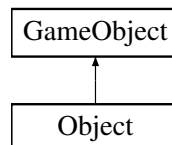
- src/editor/GUI/newgame.h
- src/editor/GUI/newgame.cpp

8.46 Object Class Reference

The [Object](#) class.

```
#include <object.h>
```

Inheritance diagram for Object:



Public Member Functions

- **Object** ([Game](#) *g, [GameObject](#) *parent)

8.46.1 Detailed Description

The [Object](#) class.

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

- src/editor/Game/[object.h](#)
- src/editor/Game/object.cpp

8.47 ObjectEditor Class Reference

Public Member Functions

- **ObjectEditor** (QWidget *parent=0)
- void **setGame** ([Game](#) *g)

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

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

8.48 ObjectFlagTableModel Class Reference

Public Member Functions

- **ObjectFlagTableModel** ([GameObject](#) *obj, QObject *parent=0)

- int **rowCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- int **columnCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- QVariant **data** (const QModelIndex &index, int role) const Q_DECL_OVERRIDE
- Qt::ItemFlags **flags** (const QModelIndex &index) const
- bool **setData** (const QModelIndex &index, const QVariant &value, int role)
- QVariant **headerData** (int section, Qt::Orientation orientation, int role) const

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

- src/editor/Game/[mapslistmodel.h](#)
- src/editor/Game/mapslistmodel.cpp

8.49 ObjectParamTableModel Class Reference

Public Member Functions

- **ObjectParamTableModel** ([GameObject](#) *obj, QObject *parent=0)
- int **rowCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- int **columnCount** (const QModelIndex &parent) const Q_DECL_OVERRIDE
- QVariant **data** (const QModelIndex &index, int role) const Q_DECL_OVERRIDE
- Qt::ItemFlags **flags** (const QModelIndex &index) const
- bool **setData** (const QModelIndex &index, const QVariant &value, int role)
- QVariant **headerData** (int section, Qt::Orientation orientation, int role) const

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

- src/editor/Game/[mapslistmodel.h](#)
- src/editor/Game/mapslistmodel.cpp

8.50 src.world.ObjectType Class Reference

Public Member Functions

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

Public Attributes

- **type**

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

- src/world.py

8.51 Options Struct Reference

The [Options](#) class provides session-independent options and preferences.

```
#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)
- `template<class T >`
`void setDefault` (QString group, QString opt, T val)
- `bool isAdjustable` (QString group, QString opt, bool adjust=true)
- `void setAdjustable` (QString group, QString opt, bool adjust)
- `void reinitialise` (QString group="")

Static Public Member Functions

- static `Options & options` ()

8.51.1 Detailed Description

The [Options](#) class provides session-independent options and preferences.

Features

The [Options](#) class aims at storing global options, that are available at any place in the entire application. The preferences are permanently stored and remain between the separate sessions and windows.

Two sorts of options exist :

- The adjustable ones : the value of the option change when [save](#) is called.
- The non-adjustable ones : the value of the option doesn't change is [save](#) is called, the option must be modified with .

The sort of option can be set with the [setAdjustable](#) function.

Design

The [Options](#) class is designed following the `__Singleton` design pattern__. The constructor is thus private, and the only [Options](#) instance is created at the first call of [options](#).

QSetting is used internaly, see Qt's documentation for details about the storing mechanisms.

Reading and writting existing options

To read or write options, the [Options](#) instance must be retrieved, using the [options](#) function, then the [load](#) and [save](#) functions can be called.

Adding options

To add a new option, it is only needed to add a default hard coded value, using the [Default](#) and [DefaultF](#) macros in the [Options](#) constructor.

It is strongly advice to use macro to define new options group (see [WIN](#), for an example).

Note

To use [Options](#) with custom types (other than “C++” standard), the defining header of the type must be included at the top of the [options.h](#) file, in order to be used in the default value declaration.

Warning

Pointer objects are not supported, and the result of the use of the [Options](#) class with such values is undefined.

See also

[options.h](#)

8.51.2 Member Function Documentation

8.51.2.1 `bool Options::isAdjustable (QString group, QString opt, bool adjust = true)`

Returns “true” if the option defined by its group and name is adjustable, “false” elsewhere.

See also

[setAdjustable](#)

8.51.2.2 `template<class T> T Options::load (QString group, QString opt) [inline]`

Reads an option defined by its group and name.

Note

The template argument must be precised since it can't be deduced from arguments' types.

Warning

If the option type and the reading type mismatch, an default null value is returned.

See also

[save](#)

8.51.2.3 Options & Options::options () [static]

Returns the unique [Options](#) instance.

8.51.2.4 void Options::reinitialise (QString group = " ")

Clear all options from the group. If “group == “”, all entries are deleted.

8.51.2.5 template<class T > void Options::save (QString group, QString opt, T val) [inline]

Writes the new value of the options defined by its group and name, if the option is adjustable. See [Options](#) for details about options types.

Note

The template argument can be omitted since it would be deduced from the value argument.

See also

[setDefault](#), [load](#)

8.51.2.6 void Options::setAdjustable (QString group, QString opt, bool adjust)

Sets if the option defined by its group and name is adjustable.

See also

[isAdjustable](#)

8.51.2.7 template<class T > void Options::setDefault (QString group, QString opt, T val) [inline]

Writes the new value of the options defined by its group and name, whatever the option type is. See [Options](#) for details about options types.

Note

The template argument can be omitted since it would be deduced from the value argument.

See also

[save](#), [load](#)

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

- src/editor/GUI/[options.h](#)
- src/editor/GUI/options.cpp

8.52 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

8.53 src.orders.OrderDispatcher Class Reference

Public Member Functions

- def **__init__**
- def [treat](#)

Public Attributes

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

8.53.1 Detailed Description

pour diminuer la redondance de code client/serveur

8.53.2 Member Function Documentation

8.53.2.1 `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`

8.54 ParamItemDelegate Class Reference

Public Member Functions

- **ParamItemDelegate** (QObject *parent=nullptr)
- QWidget * **createEditor** (QWidget *parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- void **setEditorData** (QWidget *editor, const QModelIndex &index) const
- void **updateEditorGeometry** (QWidget *editor, const QStyleOptionViewItem &option, const QModelIndex &index) const
- void **setModelData** (QWidget *editor, QAbstractItemModel *model, const QModelIndex &index) const

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

- `src/editor/GUI/Tabs/objecteditor.h`
- `src/editor/GUI/Tabs/objecteditor.cpp`

8.55 src.tools.Perf Class Reference

Public Member Functions

- `def __init__`
- `def tic`
- `def toc`
- `def show`

Public Attributes

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

8.55.1 Detailed Description

Calcule les performances d'un morceau de code

8.55.2 Member Function Documentation

8.55.2.1 `def src.tools.Perf.show (self)`

Affiche le rapport

8.55.2.2 `def src.tools.Perf.tic (self)`

À lancer avant la fonction

8.55.2.3 `def src.tools.Perf.toc (self)`

À lancer après la fonction

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

- `src/tools.py`

8.56 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)

8.56.1 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

[RCoords](#), [CCoords](#), [PxCoords](#)

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

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

8.57 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)

8.57.1 Detailed Description

The [PxCoords](#) class describe positions with real pixel coordinates.

Theses coordinates describe the pixel position.

See also

[RCoords](#), [CCoords](#), [PtCoords](#)

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

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

8.58 RCoords Class Reference

The [RCoords](#) class describe positions with relative coordinates.

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

Public Member Functions

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

8.58.1 Detailed Description

The [RICoords](#) class describe positions with relative coordinates.

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

See also

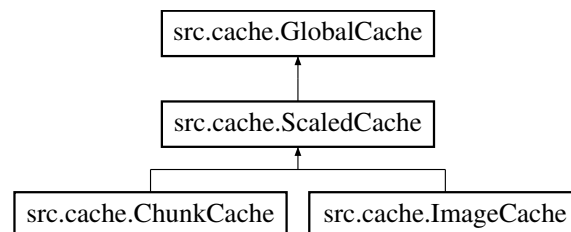
[CICoords](#) [PtCoords](#), [PxCoords](#)

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

- src/editor/Game/[mappainter.h](#)

8.59 src.cache.ScaledCache Class Reference

Inheritance diagram for src.cache.ScaledCache:



Public Member Functions

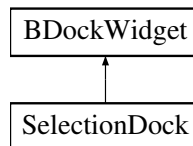
- def **add_scaled**
- def **remove**
- def **get_elt**
- def **free_cache**

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

- src/cache.py

8.60 SelectionDock Class Reference

Inheritance diagram for SelectionDock:



Public Member Functions

- **SelectionDock** ([MapView](#) *mv, QWidget *parent=0)

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

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

8.61 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

8.62 src.network.ServerConnection Class Reference

Inherits Thread.

Public Member Functions

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

Public Attributes

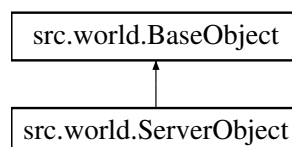
- **soc**
- **handle**

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

- src/network.py

8.63 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

8.64 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

8.65 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

8.66 src.utils.WalkableGraph Class Reference

Public Member Functions

- def **__init__**
- def **get_neighbors**
- def **dist**
- def **get_path**

Public Attributes

- **walkables**

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

- `src/utls.py`

8.67 Welcome Class Reference**Public Member Functions**

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

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

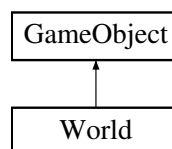
- `src/editor/GUI/Tabs/welcome.h`
- `src/editor/GUI/Tabs/welcome.cpp`

8.68 World Class Reference

The [World](#) class.

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

Inheritance diagram for World:

**Public Member Functions**

- **World** ([Game](#) *g, [GameObject](#) *parent)
- **ObjectListD** (m, M, ap,, s, [Map](#)) ObjectListD(o

Public Attributes

- **O**
- **bject**
- **s**

8.68.1 Detailed Description

The [World](#) class.

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

- [src/editor/Game/game.h](#)
- [src/editor/Game/game.cpp](#)

8.69 [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](#)

8.70 [WorldEditor](#) Class Reference

Public Member Functions

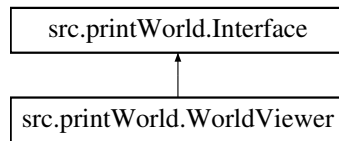
- **WorldEditor** (`QWidget *parent=0`)
- void **setGame** ([Game](#) *g)

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

- [src/editor/GUI/Tabs/worldeditor.h](#)
- [src/editor/GUI/Tabs/worldeditor.cpp](#)

8.71 [src.printWorld.WorldViewer](#) Class Reference

Inheritance diagram for [src.printWorld.WorldViewer](#):



Public Member Functions

- def **__init__**
- def **get_event**
- def **end**
- def **update**
- def **render**
- def **move**
- def **zoom**
- def **move_char**
- def **propagate_trigger**

Public Attributes

- **screen_size**
- **current_map**
- **main_char**
- **characters**

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

- `src/printWorld.py`

8.72 XmlHandler Class Reference

Public Member Functions

- **XmlHandler** ([Game](#) *g)
- bool **startElement** (const QString &, const QString &localName, const QString &, const QDomAttributes &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 9

File Documentation

9.1 src/editor/Game/game.h File Reference

Definition of the [Game](#) and [World](#) classes.

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

Classes

- class [World](#)
The [World](#) class.
- class [Game](#)
The [Game](#) class gather the differents parts needed to describe a game.

9.1.1 Detailed Description

Definition of the [Game](#) and [World](#) classes.

9.2 src/editor/Game/map.h File Reference

Definition of the [Map](#), [Cell](#) and [CellType](#) classes.

```
#include "object.h"
```

Classes

- class [CellType](#)
The [CellType](#) class.
- class [Cell](#)

The [Cell](#) class.

- class [Map](#)

The [Map](#) class.

Defines

- #define [forCells](#)(i) int nbCell = width()*height(); for(int i(0); i<nbCell; ++i)

9.2.1 Detailed Description

Definition of the [Map](#), [Cell](#) and [CellType](#) classes.

9.2.2 Define Documentation

9.2.2.1 #define [forCells](#)(i) int nbCell = width()*height(); for(int i(0); i<nbCell; ++i)

Usefull macro to set up a for on the cells

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

Definition of the [MapPainter](#) class and other related classes to render [maps](#).

```
#include "map.h" #include "GUI/options.h"
```

Classes

- class [RICoords](#)
The [RICoords](#) class describe positions with relative coordinates.
- class [CICoords](#)
The [CICoords](#) 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 subtraction operation.

9.3.1 Detailed Description

Definition of the [MapPainter](#) class and other related classes 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.

Author

Baptiste Pauget

9.3.2 Function Documentation

9.3.2.1 [MapPainter::Element operator^](#) ([MapPainter::Element](#) a, [MapPainter::Element](#) b) `[inline]`

The operator ^ is the flag subtraction operation.

Warning

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

9.4 src/editor/Game/mapslistmodel.h File Reference

Definition of Model/View presentation classes.

```
#include <QAbstractListModel>    #include <QAbstractTableModel>
#include "game.h" #include "mappainter.h"
```

Classes

- class [MapsListModel](#)
The [MapsListModel](#) class provides a presentation class for the Qt Model-View framework.
- class [CellTypeListModel](#)

The [CellTypeListModel](#) class.

- class [ObjectParamTableModel](#)
- class [ObjectFlagTableModel](#)

9.4.1 Detailed Description

Definition of Model/View presentation classes.

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

Definition of the base class [GameObject](#), and some inherited classes.

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

Classes

- class [GameObject](#)

The [GameObject](#) 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 resources a unique identifier.

- class [Object](#)

The [Object](#) class.

Defines

- #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)
- #define [ObjectListDef](#)(Objects, Type) private: QMap<int, Type*> a##Objects; public:
- #define [ObjectListAdd](#)([Object](#), Objects, Type) void add##[Object](#)(Type* new##[Object](#)){a##Objects[new##[Object](#)->ident()] = new##[Object](#); touch();}
- #define [ObjectListTake](#)([Object](#), Objects, Type) Type* take##[Object](#)(int id){touch(); return a##Objects.take(id);}
- #define [ObjectListGetter](#)(object, Objects, Type) inline Type* object(int id) const{return a##Objects.value(id, nullptr);}
- #define [ObjectListValues](#)(objects, Objects, Type) inline QList<Type*> objects() const{return a##Objects.values();}
- #define [ObjectListGetters](#)(object, [Object](#), objects, Objects, Type) ObjectList-Getter(object, Objects, Type) ObjectListValues(objects, Objects, Type)
- #define [ObjectListModifiers](#)([Object](#), Objects, Type) ObjectListAdd([Object](#), -Objects, Type) ObjectListTake([Object](#), Objects, Type)

- #define **ObjectList**(object, [Object](#), objects, Objects, Type) ObjectListDef(-Objects,Type) ObjectListGetters(object,[Object](#),objects,Objects,Type) ObjectListModifiers([Object](#), Objects, Type)
- #define **ObjectListD**(init, Init, body, sg, pl, Type) ObjectList(init##body##sg,-Init##body##sg,init##body##pl,Init##body##pl,Type)
- #define **C**(Macro, init, Init, body,...) Macro(init##body, Init##body, ## __VA_ARGS__)
- #define **C0**(Macro, init, Init, body) Macro(init##body, Init##body)
- #define **C1**(Macro, init, Init, body, arg) Macro(init##body, Init##body, arg)
- #define **SetFlag**(flag, value) aFlags[#flag] = value
- #define **FlagGetter**(flag, [Flag](#)) inline bool is##[Flag](#)() const{return aFlags[#flag];}
- #define **FlagSetter**(flag, [Flag](#)) inline void set##[Flag](#)(bool flag){[SetFlag](#)(flag,flag); touch();}
- #define **Flag**(flag, [Flag](#)) [FlagGetter](#)(flag, [Flag](#)) [FlagSetter](#)(flag, [Flag](#))
- #define **SetParam**(param, value) aParams[#param] = value
- #define **ParamGetter**(param) inline int param() const{return aParams[#param];}
- #define **ParamSetter**(param, [Param](#)) inline void set##[Param](#)(int param##Value){[SetParam](#)(param,param##Value); touch();}
- #define **Param**(param, [Param](#)) [ParamGetter](#)(param) [ParamSetter](#)(param, [Param](#))
- #define **AttrGetter**(attr, [Attr](#), Type) inline Type* attr() const{return a##[Attr](#);}
- #define **AttrFree**([Attr](#)) if(a##[Attr](#)) a##[Attr](#)->removeReference();
- #define **AttrLink**([Attr](#)) if(a##[Attr](#)) a##[Attr](#)->addReference();
- #define **AttrSetter**(attr, [Attr](#), Type) inline void set##[Attr](#)(Type* new##[Attr](#)){AttrFree([Attr](#)); a##[Attr](#) = new##[Attr](#); AttrLink([Attr](#)); touch();}
- #define **AttrDef**([Attr](#), Type) private: Type* a##[Attr](#) = nullptr; public:
- #define **Attr**(attr, [Attr](#), Type) [AttrDef](#)([Attr](#), Type) [AttrGetter](#)(attr,[Attr](#),Type) [AttrSetter](#)(attr, [Attr](#), Type)
- #define **AttrT**(type, Type) [Attr](#)(type, Type, Type)

9.5.1 Detailed Description

Definition of the base class [GameObject](#), and some inherited classes. ## The objects structure

Objects destructors

The Macro System

To add conveniently attributes and flags to [GameObject](#) subclassed objects, a set of macro is provided.

Name conventions

For a attribute named `attr`, the following conventions are observed :

- `attr()` is the getter method
- `setAttr()` is the setter method
- `aAttr` is the name of the attribut (if any)

A specific convention is applied for flags (boolean attributes) :

- isAttr() is the getter method

Macros

To define a new attribute, a global macro can be used in the class declaration. The provided basic implementations keep the object edition synchronization.

If a cleverer process is needed, custom getter or setter can be implemented, and the getter and setter macros can be used separately to define the obvious methods

****Provided macros****

Attribute Type	Complete declaration	Getter	Setter
Flags (bool)	<code>Flag</code>	<code>FlagGetter</code>	<code>FlagSetter</code>
Parameters (int)	<code>Param</code>	<code>ParamGetter</code>	<code>ParamSetter</code>
GameObject based Attributes	<code>Attr</code>	<code>AttrGetter</code>	<code>AttrSetter</code>

****The case of attributes****

An additionnal `AttrT` macro is provided, that deduce a default name from the type.

Name tools

To make the definition easier and avoid the name repetition that is introduced by the name convention, a `C` macro is provided to construct the names with lower and upper initial letter from theses letter and the end of the name.

Author

Baptiste Pauget

9.5.2 Define Documentation

9.5.2.1 `#define Attr(attr, Attr, Type) AttrDef(Attr, Type) AttrGetter(attr,Attr,Type) AttrSetter(attr, Attr, Type)`

The `Attr` macro defines a new `<aAttr>` named attribute of type `Type`, with its generic getter and setter methods.

With respect to the [name convention](#), this macro needs the parameter's name with lower and upper initial letter case.

Example

```
{.cpp}
Attr(parent,Parent, GameObject)
-->
private:
    GameObject *aParent;
public:
    inline GameObject* parent() const{return aParent;}
    inline void setParent(GameObject* parentObject){aParent = parentObject
; touch();}
```


See also

[AttrT](#), [AttrDef](#), [AttrSetter](#), [AttrGetter](#), [C](#)

9.5.2.2 `#define AttrDef(Attr, Type) private: Type* a##Attr = nullptr; public:`

The `AttrDef` macro defines a private attribute name `<aAttr>`.

Note

To avoid redefinition error, no attribute or method name `<aAttr>` must exist.

Warning

This macro is designed to be used in a public part of the class. Please note that inserting this macro in a private or protected part will change the visibility of the next declaration to public.

Example

```
{.cpp}
Attr(Parent,GameObject)
-->
private:
    GameObject *aParent;
public:
```

See also

[Attr](#)

9.5.2.3 `#define AttrGetter(attr, Attr, Type) inline Type* attr() const{return a##Attr;}`

The `AttrGetter` macro defines a generic getter method for the attribute named `attr` of type `Type`.

With respect to the [name convention](#), this macro needs the attribute's name with lower and upper initial letter case.

Example

```
{.cpp}
AttrGetter(parent,Parent,GameObject)
--> inline GameObject* parent() const{return aParent;}
```

See also

[Attr](#), [AttrSetter](#), [C](#)

9.5.2.4 `#define AttrSetter(attr, Attr, Type) inline void set##Attr(Type* new##Attr){AttrFree(Attr); a##Attr = new##Attr; AttrLink(Attr); touch();}`

The `AttrSetter` macro defines a generic setter method for the attribute named `attr` of type `Type`.

With respect to the [name convention](#), this macro needs the attribute's name with lower and upper initial letter case.

Example

```
{.cpp}
AttrSetter(parent,Parent,GameObject)
--> inline void setParent(GameObject* &parentObject){aParent =
    parentObject; touch();}
```

See also

[Attr](#), [AttrGetter](#), [C](#)

9.5.2.5 `#define AttrT(type, Type) Attr(type, Type, Type)`

The `AttrT` macro defines a new attribute of type `Type`, named after the type name, with its generic getter and setter methods.

With respect to the [name convention](#), this macro needs the parameter's type with lower and upper initial letter case.

Example

```
{.cpp}
AttrT(cellType,CellType)
-->
private:
    CellType *aCellType;
public:
    inline CellType* cellType() const{return aCellType;}
    inline void setCellType(CellType* cellTypeObject){aCellType =
        cellTypeObject; touch();}
```

See also

[Attr](#), [AttrDef](#), [AttrSetter](#), [AttrGetter](#), [C](#)

9.5.2.6 `#define C(Macro, init, Init, body, ...) Macro(init##body, Init##body, ##__VA_ARGS__)`

The `C` macro calls the `Macro` argument with argument tokens formed by the concatenation of `init` and `body`, and `Init` and `body`.

This enables to call a macro with the same argument, with the initial letter in lower and upper case.

A custom number of arguments can be added after the `body` one.

Note

This use of variadic arguments follow the [gcc specification](#), but can be not supported by some compilers.

As some IDE does not fully support variadic macro expansion, the [C0](#) and [C1](#) macros can be used to avoid some inconvenience due to uncomplete code understanding.

Example

```
{.cpp}
C(Param,w,W,idth)
    --> Param(width, Width)

C(Attr, p,P,arent, GameObject)
    --> Attr(parent, Parent, GameObject)
```

See also

[object.h](#)

9.5.2.7 #define C0(Macro, init, Init, body) Macro(init##body, Init##body)

The C0 macro is equivalent to the [C](#) macro, with no additional argument.

This macro is provided to avoid the use of variadic arguments that are currently not totally supported by some IDE. **Example**

```
{.cpp}
C(Flag, v,V,isible)
    --> Flag(visible, Visible)
```

See also

[C1](#)

9.5.2.8 #define C1(Macro, init, Init, body, arg) Macro(init##body, Init##body, arg)

The C1 macro is equivalent to the [C](#) macro, with one additional argument.

This macro is provided to avoid the use of variadic arguments that are currently not totally supported by some IDE. **Example**

```
{.cpp}
C(Attr, p,P,arent, GameObject)
    --> Attr(parent, Parent, GameObject)
```

See also

[C0](#)

9.5.2.9 `#define Flag(flag, Flag) FlagGetter(flag, Flag) FlagSetter(flag, Flag)`

The `Flag` macro defines generic getter and setter methods for the flag named `flag`.

With respect to the [name convention](#), this macro needs the flag's name with lower and upper initial letter case.

Example

```
{.cpp}
Flag(visible,Visible)
--> inline bool isVisible() const{return aFlags["visible"];}
    inline void setVisible(bool visible){aFlags["visible"] = visible;
    touch() }
```

See also

[FlagGetter](#), [FlagSetter](#), [C](#)

9.5.2.10 `#define FlagGetter(flag, Flag) inline bool is##Flag() const{return aFlags[#flag];}`

The `FlagGetter` macro defines a generic getter method for the flag named `flag`.

With respect to the [name convention](#), this macro needs the flag's name with lower and upper initial letter case.

Warning

The default getter method does not check whether the `flag` named flag really exist. To avoid runtime access error, it is strongly advice to initialize the flag in the object's constructor, using the setter method or the [SetFlag](#) macro.

Example

```
{.cpp}
FlagGetter(visible,Visible)
--> inline bool isVisible() const{return aFlags["visible"];}
```

See also

[Flag](#), [FlagSetter](#), [C](#)

9.5.2.11 `#define FlagSetter(flag, Flag) inline void set##Flag(bool flag){SetFlag(flag,flag); touch();}`

The `FlagSetter` macro defines a generic setter method for the flag named `flag`.

With respect to the [name convention](#), this macro needs the flag's name with lower and upper initial letter case.

Example

```
{.cpp}
FlagSetter(visible,Visible)
--> inline void setVisible(bool visible){aFlags["visible"] = visible;
touch() }
```

See also

[Flag](#), [FlagGetter](#), [C](#)

9.5.2.12 `#define ObjectsMap(pref, ini, Ini, body, sg, pl) ObjectsMapC(ini##body##sg, ini##body##pl, Ini##body##sg, Ini##body##pl, pref,ini)`

Deprecated

9.5.2.13 `#define ObjectsMapC(name, names, Type, Types, pref, arg)`

Value:

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

Deprecated

9.5.2.14 `#define Param(param, Param) ParamGetter(param) ParamSetter(param, Param)`

The Param macro defines generic getter and setter methods for the parameter named param.

With respect to the [name convention](#), this macro needs the parameter's name with lower and upper initial letter case.

Example

```
{.cpp}
Param(width,Width)
--> inline int width() const{return aParams["width"];}
    inline void setWidth(int widthValue){aParams["width"] = widthValue;
touch();}
```

See also

[ParamGetter](#), [ParamSetter](#), [C](#)

9.5.2.15 `#define ParamGetter(param) inline int param() const{return aParams[#param];}`

The ParamGetter macro defines a generic getter method for the parameter named `param`.

Warning

The default getter method does not check whether the `param` named parameter really exist. To avoid runtime access error, it is strongly advice to initialize the parameter in the object's constructor, using the setter method or the [SetParam](#) macro.

Example

```
{.cpp}
ParamGetter(width)
--> inline int width() const{return aParams["width"]};}
```

See also

[Param](#), [ParamSetter](#)

9.5.2.16 `#define ParamSetter(param, Param) inline void set##Param(int param##Value){SetParam(param,param##Value); touch();}`

The ParamSetter macro defines a generic setter method for the parameter named `param`.

With respect to the [name convention](#), this macro needs the parameter's name with lower and upper initial letter case.

Example

```
{.cpp}
ParamSetter(width,Width)
--> inline void setWidth(bool widthValue){aParams["width"] = widthValue;
touch();}
```

See also

[Param](#), [ParamGetter](#), [C](#)

9.5.2.17 `#define SetFlag(flag, value) aFlags[#flag] = value`

Convenient macro to set a flag directly.

This is usefull in custom setters, to avoid call loops.

Warning

The [touch](#) function isn't called. After this macro use, the [object](#) is no longer synchronised.

Example

```
{.cpp}
SetFlag(visible, false)
--> aFlags["visible"] = false
```

See also

[Flag](#), [FlagSetter](#), [object.h](#)

9.5.2.18 #define SetParam(param, value) aParams[#param] = value

Convenient macro to set a param directly.

This is usefull in custom setters, to avoid call loops.

Warning

The [touch](#) function isn't called. After this macro use, the [object](#) is no longer synchronised.

Example

```
{.cpp}
SetParam(width, 42)
--> aParams["width"] = 42
```

See also

[Param](#), [ParamSetter](#), [object.h](#)

9.6 src/editor/Game/xmlhandler.h File Reference

Definition og the [XmlHandler](#) class and other related classes to read XML game's files.

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

Classes

- class [XmlHandler](#)

Typedefs

- typedef std::pair< QString, FileContent > **Asso**

Enumerations

- enum **FileContent** { **FCUnknown**, **FCGame**, **FCRessources**, **FCWorld**, **FCMap**, **FCEntity**, **FCObject** }

Functions

- const QMap< QString, FileContent > **overHead** ({Asso("Game", FCGame), - Asso("Ressources", FCRessources), Asso("World", FCWorld), Asso("Map", FCMap), Asso("Entity", FCEntity), Asso("Object", FCObject)})

9.6.1 Detailed Description

Definition of the [XmlHandler](#) class and other related classes to read XML game's files.

9.7 src/editor/GUI/options.h File Reference

Definition of the [Options](#) class, and the constants that are used in this class.

```
#include <QSettings> #include <QDir> #include <QSize> ×
#include <QPoint> #include <QColor>
```

Classes

- struct [Options](#)

The [Options](#) class provides session-independent options and preferences.

Defines

- #define [WIN](#) "Window"
- #define [DIR](#) "Directories"
- #define [MAP](#) "MapsEditor"
- #define [DefaultF](#)(group, opt, val) defaultValues[group][opt] = QPair<QVariant, bool>(val, false)
- #define [Default](#)(group, opt, val) defaultValues[group][opt] = QPair<QVariant, bool>(val, true)

Variables

- const QString **ADAPT** = "Adjustable"
- const QString **VAL** = "Value"

9.7.1 Detailed Description

Definition of the [Options](#) class, and the constants that are used in this class. The headers of types which are used in the application must be include here. See [Options](#) for details.

Author

Baptiste Pauget

9.7.2 Define Documentation

9.7.2.1 `#define Default(group, opt, val) defaultValues[group][opt] = QPair<QVariant, bool>(val, true)`

This macro defines a new adaptati option identified by its group and name.

9.7.2.2 `#define DefaultF(group, opt, val) defaultValues[group][opt] = QPair<QVariant, bool>(val, false)`

This macro defines a new unadaptati option identified by its group and name.

9.7.2.3 `#define DIR "Directories"`

Group of paths options.

9.7.2.4 `#define MAP "MapsEditor"`

Group of [MapsEditor](#) related options.

9.7.2.5 `#define WIN "Window"`

Group of window related options.

9.8 src/editor/GUI/Tabs/bcolor.h File Reference

Definition of the [BColor](#) class.

```
#include <QtWidgets>
```

Classes

- class [BColor](#)

The [BColor](#) class is a simple frame that offers color selection.

9.8.1 Detailed Description

Definition of the [BColor](#) class.

Author

Baptiste Pauget

9.9 src/editor/GUI/Tabs/Docks/bdock.h File Reference

Definition of the [BDock](#) class.

```
#include <QtWidgets> #include "intertie.h" #include "bdockwidget.-  
h"
```

Classes

- class [BDock](#)

The [BDock](#) class is the container for widget to display in a [BDocksZone](#).

9.9.1 Detailed Description

Definition of the [BDock](#) class.