

扫雷

1.0.3

制作者 Doxygen 1.9.6

1 待办事项列表	1
2 命名空间索引	3
2.1 命名空间列表	3
3 继承关系索引	5
3.1 类继承关系	5
4 类索引	7
4.1 类列表	7
5 文件索引	9
5.1 文件列表	9
6 命名空间文档	11
6.1 resource_rc 命名空间参考	11
6.1.1 函数说明	11
6.1.1.1 qCleanupResources()	11
6.1.1.2 qInitResources()	12
6.1.2 变量说明	12
6.1.2.1 else	12
6.1.2.2 qt_resource_data	12
6.1.2.3 qt_resource_name	12
6.1.2.4 qt_resource_struct	13
6.1.2.5 qt_resource_struct_v1	13
6.1.2.6 qt_resource_struct_v2	13
6.1.2.7 qt_version	13
6.1.2.8 rcc_version	13
7 类说明	15
7.1 BackgroundMusicPlayer类 参考	15
7.1.1 详细描述	16
7.1.2 构造及析构函数说明	16
7.1.2.1 BackgroundMusicPlayer()	16
7.1.3 成员函数说明	16
7.1.3.1 playNewBackgroundMusc	16
7.2 Board类 参考	17
7.2.1 详细描述	19
7.2.2 构造及析构函数说明	19
7.2.2.1 Board()	19
7.2.2.2 ~Board()	20
7.2.3 成员函数说明	20
7.2.3.1 calculateCurrentIntegral()	20
7.2.3.2 dealClickChunk	20

7.2.3.3 dealDoubleClickChunk	21
7.2.3.4 dealSignalExploded	22
7.2.3.5 detect()	22
7.2.3.6 gameOver()	23
7.2.3.7 getBombNum()	24
7.2.3.8 getColNum()	25
7.2.3.9 getGameTimer()	25
7.2.3.10 getRowNum()	25
7.2.3.11 inBoard() [1/2]	26
7.2.3.12 inBoard() [2/2]	26
7.2.3.13 initSurroundBomb()	27
7.2.3.14 setBombs()	27
7.2.3.15 setIsFirstClick()	28
7.2.3.16 signalGameOver	29
7.2.3.17 signalMove	29
7.2.3.18 signalPlayNewBackGroundMusic	29
7.2.3.19 signalUpLoadHistory	30
7.2.3.20 upLoadHistory()	30
7.2.4 友元及相关函数文档	30
7.2.4.1 DenotationMod	30
7.2.5 类成员变量说明	30
7.2.5.1 bombNum	31
7.2.5.2 chunks	31
7.2.5.3 colNum	31
7.2.5.4 flagBombNum	31
7.2.5.5 GameMod	31
7.2.5.6 gameTimer	31
7.2.5.7 isFirstClick	31
7.2.5.8 minedNum	31
7.2.5.9 rowNum	32
7.2.5.10 selfCurrentIntegral	32
7.2.5.11 SurroundDirection	32
7.2.5.12 SurroundDirectionNum	32
7.3 ChooseByDirection类 参考	32
7.3.1 详细描述	33
7.3.2 构造及析构函数说明	33
7.3.2.1 ChooseByDirection()	33
7.3.3 成员函数说明	34
7.3.3.1 addItems()	34
7.3.3.2 getCurrentItems()	34
7.3.3.3 setItems()	34
7.4 Chunk类 参考	35

7.4.1 详细描述	36
7.4.2 成员枚举类型说明	36
7.4.2.1 MineState	36
7.4.2.2 MineType	37
7.4.2.3 RIGHT_KEY_MENU	37
7.4.3 构造及析构函数说明	37
7.4.3.1 Chunk()	37
7.4.4 成员函数说明	38
7.4.4.1 floatByDoubleClick()	38
7.4.4.2 getChunkSize()	38
7.4.4.3 getMineState()	38
7.4.4.4 getMineType()	39
7.4.4.5 getPix()	39
7.4.4.6 getRowCol()	39
7.4.4.7 getSurroundBomb()	40
7.4.4.8 openThenShow()	40
7.4.4.9 setMineState()	40
7.4.4.10 setMineType()	41
7.4.4.11 setPix()	41
7.4.4.12 setRowCol()	41
7.4.4.13 setSurroundBomb()	42
7.4.4.14 showBomb()	42
7.4.4.15 signalClickChunk	43
7.4.4.16 signalDoubleClickChunk	43
7.4.4.17 signalExploded	43
7.4.4.18 signalFlagBombChanged	43
7.5 DenotationMod类 参考	44
7.5.1 详细描述	46
7.5.2 构造及析构函数说明	46
7.5.2.1 DenotationMod()	46
7.6 Leaderboard类 参考	47
7.6.1 详细描述	48
7.6.2 构造及析构函数说明	48
7.6.2.1 Leaderboard()	48
7.6.3 成员函数说明	48
7.6.3.1 dealMainSocketNewRecvMessage	48
7.6.4 友元及相关函数文档	49
7.6.4.1 Packet< Leaderboard >	49
7.7 MainWindow类 参考	49
7.7.1 详细描述	50
7.7.2 构造及析构函数说明	50
7.7.2.1 MainWindow()	50

7.7.2.2 ~MainWindow()	51
7.7.3 成员函数说明	51
7.7.3.1 signalMainSocketNewRecvMessage	51
7.7.4 友元及相关函数文档	51
7.7.4.1 Packet< MainWindow >	51
7.8 MessageTips类 参考	52
7.8.1 详细描述	53
7.8.2 构造及析构造函数说明	53
7.8.2.1 MessageTips()	53
7.8.2.2 ~MessageTips()	54
7.8.3 成员函数说明	54
7.8.3.1 getBackgroundColor()	54
7.8.3.2 setFrameColor()	54
7.8.3.3 setFrameSize()	55
7.8.3.4 getOpacityValue()	55
7.8.3.5 getShowTime()	55
7.8.3.6 getTextColor()	55
7.8.3.7 getTextSize()	56
7.8.3.8 paintEvent()	56
7.8.3.9 setBackgroundColor()	56
7.8.3.10 setCloseTimeSpeed()	56
7.8.3.11 setFrameColor()	57
7.8.3.12 setFrameSize()	57
7.8.3.13 setOpacityValue()	57
7.8.3.14 setShowTime()	58
7.8.3.15 setTextColor()	58
7.8.3.16 setTextSize()	59
7.9 NetBoard类 参考	59
7.9.1 详细描述	62
7.9.2 构造及析构造函数说明	62
7.9.2.1 NetBoard()	62
7.9.3 成员函数说明	63
7.9.3.1 dealAntiGameOver	63
7.9.3.2 dealClickChunk	63
7.9.3.3 dealMainSocketNewRecvMessage	64
7.9.3.4 dealMatchResponse	65
7.9.3.5 dealNetInitState	66
7.9.3.6 dealUpdateAntiIntegral	66
7.9.3.7 gameOver()	66
7.9.3.8 queryNewMatch()	67
7.9.3.9 sendIntegralToServer()	68
7.9.4 友元及相关函数文档	69

7.9.4.1 Packet< NetBoard >	69
7.10 NetDenotationMod类 参考	69
7.10.1 详细描述	73
7.10.2 构造及析构函数说明	73
7.10.2.1 NetDenotationMod()	73
7.10.3 成员函数说明	74
7.10.3.1 dealMatchResponse	74
7.10.3.2 dealSignalExploded	74
7.10.4 友元及相关函数文档	75
7.10.4.1 Packet< NetDenotationMod >	75
7.11 Packet< T > 模板类 参考	75
7.11.1 详细描述	76
7.11.2 构造及析构函数说明	76
7.11.2.1 Packet()	76
7.11.3 成员函数说明	76
7.11.3.1 formatMes() [1/2]	76
7.11.3.2 formatMes() [2/2]	76
7.11.3.3 installClassFunctionEvent()	77
7.11.3.4 pushMessage()	77
7.12 Chunk::RowCol结构体 参考	77
7.12.1 构造及析构函数说明	78
7.12.1.1 RowCol()	78
7.12.2 友元及相关函数文档	78
7.12.2.1 operator<	78
7.12.3 类成员变量说明	78
7.12.3.1 col	78
7.12.3.2 row	79
7.13 SettingWindow类 参考	79
7.13.1 详细描述	80
7.13.2 构造及析构函数说明	80
7.13.2.1 SettingWindow()	80
7.13.3 成员函数说明	80
7.13.3.1 reloadApp()	80
8 文件说明	81
8.1 C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp 文件参考	81
8.2 C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h 文件参考	81
8.3 BackgroundMusicPlayer.h	82
8.4 C:/Users/SJ/Desktop/扫雷/客户端源码/Base.h 文件参考	83
8.4.1 宏定义说明	84
8.4.1.1 dendl	84
8.4.1.2 dout	84

8.4.2 函数说明	84
8.4.2.1 inRange()	84
8.4.2.2 inRect()	85
8.5 Base.h	85
8.6 C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp 文件参考	85
8.7 C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h 文件参考	86
8.8 Board.h	87
8.9 C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.cpp 文件参考	88
8.10 C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h 文件参考	88
8.11 ChooseByDirection.h	89
8.12 C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp 文件参考	89
8.13 C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h 文件参考	90
8.14 Chunk.h	91
8.15 C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp 文件参考	92
8.16 C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h 文件参考	93
8.17 DenotationMod.h	93
8.18 C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp 文件参考	94
8.19 C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h 文件参考	94
8.20 Leaderboard.h	95
8.21 C:/Users/SJ/Desktop/扫雷/客户端源码/main.cpp 文件参考	96
8.21.1 函数说明	96
8.21.1.1 main()	97
8.21.1.2 myMessageOutput()	97
8.22 C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp 文件参考	97
8.23 C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.h 文件参考	98
8.24 MainWindow.h	99
8.25 C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp 文件参考	100
8.26 C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h 文件参考	100
8.27 MessageTips.h	101
8.28 C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp 文件参考	102
8.29 C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h 文件参考	103
8.30 NetBoard.h	104
8.31 C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.cpp 文件参考	104
8.32 C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h 文件参考	105
8.33 NetDenotationMod.h	106
8.34 C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.cpp 文件参考	106
8.34.1 宏定义说明	107
8.34.1.1 dendl	107
8.34.1.2 dout	107
8.35 C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.h 文件参考	108
8.36 Packet.h	109
8.37 C:/Users/SJ/Desktop/扫雷/客户端源码/resource_rc.py 文件参考	109

8.38 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.cpp 文件参考	110
8.39 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h 文件参考	110
8.40 SettingWindow.h	111
Index	113

Chapter 1

待办事项列表

类 `MessageTips`

添加move功能 改成更好看的样式

Chapter 2

命名空间索引

2.1 命名空间列表

这里列出了所有命名空间定义,附带简要说明:

resource_rc	11
---------------------------------------	----

Chapter 3

继承关系索引

3.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Ui::Board	
Board	17
DenotationMod	44
NetBoard	59
NetDenotationMod	69
Ui::Leaderboard	
Leaderboard	47
Ui::MainWindow	
MainWindow	49
Packet< T >	75
Packet< Leaderboard >	75
Packet< MainWindow >	75
Packet< NetBoard >	75
Packet< NetDenotationMod >	75
QLabel	
ChooseByDirection	32
Chunk	35
QMainWindow	
Board	17
Leaderboard	47
MainWindow	49
SettingWindow	79
QObject	
BackgroundMusicPlayer	15
QWidget	
MessageTips	52
Chunk::RowCol	77
Ui::SettingWindow	
SettingWindow	79

Chapter 4

类索引

4.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明：

BackgroundMusicPlayer	
The BackgroundMusicPlayer class 这个类设置了退出自动删除，不必也不应该使用智能指针 该类是多线程类,将其moveToThread后通过信号使用它	15
Board	
The Board class 经典模式的游戏类	17
ChooseByDirection	
The ChooseByDirection class 自定义的根据方向键切换模式的控件,支持鼠标点击，会自动获取焦点 T 为可选选项的数据类型	32
Chunk	
The Chunk class 单个块对象	35
DenotationMod	
The DenotationMod class 爆炸模式，触雷不会死，但点击次数有限	44
Leaderboard	
The Leaderboard class 显示历史战绩	47
MainWindow	
The MainWindow class 主窗口对象	49
MessageTips	
The MessageTips class 实现自动消失的消息框,由于时间原因,此代码借鉴于csdn 必须使用指针类型或者指定父对象 @my doing 添加了对qt6的兼容 添加关闭时自动删除，释放资源,避免内存泄露	52
NetBoard	
The NetBoard class 经典模式的网络对战	59
NetDenotationMod	
The NetDenotationMod class 爆炸模式网络对战	69
Packet< T >	
用于socket协议的信息封装和解包，可以绑定信息-回调函数,Packet.cpp和Packet.h都得放在头文件中(-I Packet.cpp Packet.h) 如果要绑定私有行为，应该将Packet<T>声明为友元 T为parent对应的类名,installClassFunctionEvent 会在触发时调用parent的成员函数 所有要绑定的函数都应该以void为返回值,QStringList为参数	75
Chunk::RowCol	77
SettingWindow	
The SettingWindow class 设置界面	79

Chapter 5

文件索引

5.1 文件列表

这里列出了所有文件，并附带简要说明:

C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp	81
C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h	81
C:/Users/SJ/Desktop/扫雷/客户端源码/Base.h	83
C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp	85
C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h	86
C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.cpp	88
C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h	88
C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp	89
C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h	90
C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp	92
C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h	93
C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp	94
C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h	94
C:/Users/SJ/Desktop/扫雷/客户端源码/main.cpp	96
C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp	97
C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.h	98
C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp	102
C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h	103
C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.cpp	104
C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h	105
C:/Users/SJ/Desktop/扫雷/客户端源码/resource.rc.py	109
C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.cpp	110
C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h	110
C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp	100
C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h	100
C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.cpp	106
C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.h	108

Chapter 6

命名空间文档

6.1 resource_rc 命名空间参考

函数

- def `qInitResources()`
- def `qCleanupResources()`

变量

- b `qt_resource_data`
- b `qt_resource_name`
- b `qt_resource_struct_v1`
- b `qt_resource_struct_v2`
- list `qt_version` = [int(v) for v in QtCore.qVersion().split('.')]
- int `rcc_version` = 1
- b `qt_resource_struct` = `qt_resource_struct_v1`
- `else` :

6.1.1 函数说明

6.1.1.1 qCleanupResources()

```
def resource_rc.qCleanupResources ( )
```

函数调用图:



6.1.1.2 qInitResources()

```
def resource_rc.qInitResources ( )
```

这是这个函数的调用关系图:



6.1.2 变量说明

6.1.2.1 else

```
resource_rc.else :
```

6.1.2.2 qt_resource_data

```
b resource_rc.qt_resource_data
```

6.1.2.3 qt_resource_name

```
b resource_rc.qt_resource_name
```

初始值:

```

00001 = b"\
00002 \x00\x03\
00003 \x00\x00\x70\x37\
00004 \x00\x69\
00005 \x00\x6d\x00\x67\
00006 \x00\x08\
00007 \x0c\xa6\xc3\x95\
00008 \x00\x52\
00009 \x00\x65\x00\x73\x00\x6f\x00\x75\x00\x72\x00\x63\x00\x65\
00010 \x00\x0d\
00011 \x0a\xb7\x65\x47\
00012 \x00\x73\
00013 \x00\x61\x00\x66\x00\x65\x00\x43\x00\x68\x00\x75\x00\x6e\x00\x6b\x00\x2e\x00\x70\x00\x6e\x00\x67\
00014 \x00\x0f\
00015 \x0e\x3d\xd6\xa7\
00016 \x00\x75\
00017 \x00\x6e\x00\x4d\x00\x69\x00\x6e\x00\x65\x00\x64\x00\x42\x00\x6f\x00\x6d\x00\x62\x00\x2e\x00\x70\x00\x6e\x00\x67\
00018 \x00\x10\
00019 \x02\x89\x10\x27\
00020 \x00\x75\
00021 \x00\x6e\x00\x6b\x00\x6e\x00\x6f\x00\x77\x00\x6e\x00\x43\x00\x68\x00\x75\x00\x6e\x00\x6b\x00\x2e\x00\x70\x00\x6e\x00\x67\
00022 \x00\x10\
00023 \x04\x9b\x1b\xa7\
00024 \x00\x65\
00025 \x00\x78\x00\x70\x00\x6c\x00\x6f\x00\x64\x00\x65\x00\x64\x00\x42\x00\x6f\x00\x6d\x00\x62\x00\x2e\x00\x70\x00\x6e\x00\x67\
00026 "
```

6.1.2.4 qt_resource_struct

```
b resource_rc.qt_resource_struct = qt_resource_struct_v1
```

6.1.2.5 qt_resource_struct_v1

```
b resource_rc.qt_resource_struct_v1
```

初始值:

```
00001 = b"\
00002 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x01\
00003 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x02\
00004 \x00\x00\x00\x0c\x00\x02\x00\x00\x00\x01\x00\x00\x00\x03\
00005 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x04\x00\x00\x00\x04\
00006 \x00\x00\x00\x66\x00\x00\x00\x00\x00\x01\x00\x00\x1b\xb3\
00007 \x00\x00\x00\x8c\x00\x00\x00\x00\x00\x01\x00\x00\x4a\xa6\
00008 \x00\x00\x00\x22\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\
00009 \x00\x00\x00\x42\x00\x00\x00\x00\x00\x01\x00\x00\x09\xf1\
00010 "
```

6.1.2.6 qt_resource_struct_v2

```
b resource_rc.qt_resource_struct_v2
```

初始值:

```
00001 = b"\
00002 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x01\
00003 \x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
00004 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x02\
00005 \x00\x00\x00\x00\x00\x00\x00\x00\x00\
00006 \x00\x00\x00\x0c\x00\x02\x00\x00\x00\x01\x00\x00\x00\x03\
00007 \x00\x00\x00\x00\x00\x00\x00\x00\x00\
00008 \x00\x00\x00\x00\x00\x02\x00\x00\x00\x04\x00\x00\x00\x04\
00009 \x00\x00\x00\x00\x00\x00\x00\x00\x00\
00010 \x00\x00\x00\x66\x00\x00\x00\x00\x00\x01\x00\x00\x1b\xb3\
00011 \x00\x00\x01\x87\x4a\x54\xe6\xaf\
00012 \x00\x00\x00\x8c\x00\x00\x00\x00\x00\x01\x00\x00\x4a\xa6\
00013 \x00\x00\x01\x87\x7e\x8a\xcd\x1e\
00014 \x00\x00\x00\x22\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\
00015 \x00\x00\x01\x87\x7e\x89\x10\x87\
00016 \x00\x00\x00\x42\x00\x00\x00\x00\x00\x01\x00\x00\x09\xf1\
00017 \x00\x00\x01\x87\x7e\x8a\x24\x97\
00018 "
```

6.1.2.7 qt_version

```
list resource_rc.qt_version = [int(v) for v in QtCore.qVersion().split('.')]
```

6.1.2.8 rcc_version

```
int resource_rc.rcc_version = 1
```


Chapter 7

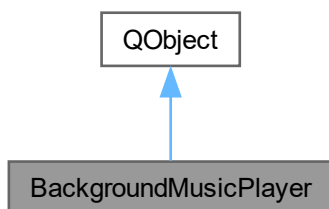
类说明

7.1 BackgroundMusicPlayer类 参考

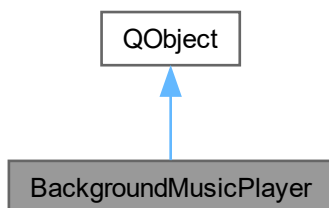
The `BackgroundMusicPlayer` class 这个类设置了退出自动删除，不必也不应该使用智能指针 该类是多线程类,将其`moveToThread`后通过信号使用它

```
#include <BackgroundMusicPlayer.h>
```

类 `BackgroundMusicPlayer` 继承关系图:



`BackgroundMusicPlayer` 的协作图:



Public 槽

- virtual void [playNewBackgroundMusc](#) (QString path)
[BackgroundMusicPlayer::playNewBackgroundMusc](#) 播放新的音乐

Public 成员函数

- [BackgroundMusicPlayer](#) (QObject *parent=nullptr)
[BackgroundMusicPlayer::BackgroundMusicPlayer](#)

7.1.1 详细描述

The [BackgroundMusicPlayer](#) class 这个类设置了退出自动删除，不必也不应该使用智能指针 该类是多线程类,将其moveToThread后通过信号使用它

7.1.2 构造及析构函数说明

7.1.2.1 BackgroundMusicPlayer()

```
BackgroundMusicPlayer::BackgroundMusicPlayer (
    QObject * parent = nullptr ) [explicit]
```

[BackgroundMusicPlayer::BackgroundMusicPlayer](#)

参数

parent	
--------	--

7.1.3 成员函数说明

7.1.3.1 playNewBackgroundMusc

```
void BackgroundMusicPlayer::playNewBackgroundMusc (
    QString path ) [virtual], [slot]
```

[BackgroundMusicPlayer::playNewBackgroundMusc](#) 播放新的音乐

参数

path	文件地址
------	------

该类的文档由以下文件生成:

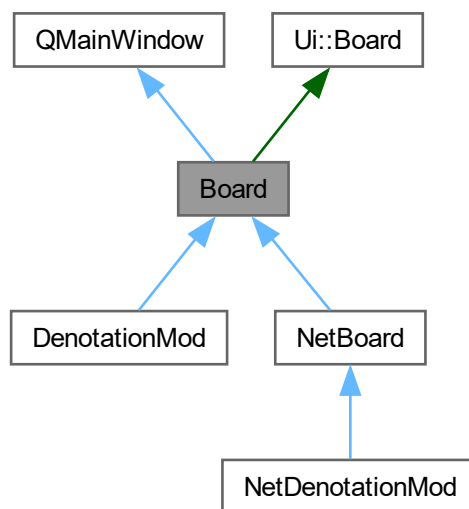
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp

7.2 Board类 参考

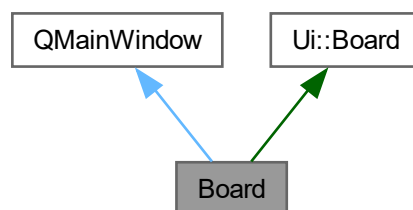
The `Board` class 经典模式的游戏类

```
#include <Board.h>
```

类 `Board` 继承关系图:



`Board` 的协作图:



信号

- void `signalGameOver` ()
- void `signalPlayNewBackGroundMusic` (QString path)
- void `signalMove` ()
- void `signalUpLoadHistory` (QString gameMod, QString `rowNum`, QString `colNum`, QString `bombNum`, QString integral)

Public 成员函数

- `Board` (qint32 `rowNum`, qint32 `colNum`, qint32 `bombNum`, QWidget *parent=nullptr, QString `GameMod`="Classic")
`Board::Board`
- `~Board` ()
`Board::~~Board`
- qint32 `getRowNum` () const
`Board::getRowNum`
- qint32 `getColNum` () const
`Board::getColNum`
- qint32 `getBombNum` () const
`Board::getBombNum`
- QPointer< QTimer > `getGameTimer` () const
`Board::getGameTimer` 获得游戏计时器对象
- void `setIsFirstClick` (bool newIsFirstClick)
`Board::setIsFirstClick`

Protected 槽

- virtual void `dealSignalExploded` ()
`Board::dealSignalExploded` 触雷事件槽函数
- virtual void `dealClickChunk` (Chunk::RowCol rc)
`Board::dealClickChunk` 单击事件槽函数
- virtual void `dealDoubleClickChunk` (Chunk::RowCol rc, bool system=false)
`Board::dealDoubleClickChunk` 双击事件槽函数

Protected 成员函数

- void `setBombs` (Chunk::RowCol firstClickedRC)
`Board::setBombs` 初始化雷区,若有物理引擎将会调用其来生成随机数
- virtual void `detect` (Chunk::RowCol rc)
`Board::detect` 递归探索周围的雷区
- void `initSurroundBomb` ()
`Board::initSurroundBomb` 每个chunk初始化周围的雷数
- bool `inBoard` (Chunk::RowCol rc)
`Board::inBoard` 判断是否在棋盘内部
- bool `inBoard` (qint32 row, qint32 col)
`Board::inBoard` 重载
- virtual void `gameOver` (QString loseOrWin)
`Board::gameOver` 游戏结束槽函数
- virtual void `upLoadHistory` ()
`Board::upLoadHistory` 上传历史记录到服务器
- virtual qint32 `calculateCurrentIntegral` ()
`Board::calculateCurrentIntegral` 计算积分, 正相关于探索率, 负相关于游戏时长

Protected 属性

- const QString [GameMod](#)
- QVector< QVector< QPointer< [Chunk](#) > > > [chunks](#)
- const qint32 [rowNum](#)
- const qint32 [colNum](#)
- const qint32 [bombNum](#)
- qint32 [flagBombNum](#) =0
- qint32 [minedNum](#)
- QPointer< QTimer > [gameTimer](#)
- bool [isFirstClick](#)
- qint32 [selfCurrentIntegral](#) =0

静态 Protected 属性

- static const qint32 [SurroundDirectionNum](#) = 8
- static constexpr qint32 [SurroundDirection](#) [[SurroundDirectionNum](#)][2]

友元

- class [DenotationMod](#)

7.2.1 详细描述

The [Board](#) class 经典模式的游戏类

7.2.2 构造及析构函数说明

7.2.2.1 Board()

```
Board::Board (
    qint32 rowNum,
    qint32 colNum,
    qint32 bombNum,
    QWidget * parent = nullptr,
    QString GameMod = "Classic" )
```

[Board::Board](#)

参数

parent	
--------	--

[Board::Board](#)

参数

rowNum	行数
colNum	列数
bombNum	雷数
parent	

7.2.2.2 ~Board()

Board::~~Board ()

Board::~~Board

7.2.3 成员函数说明

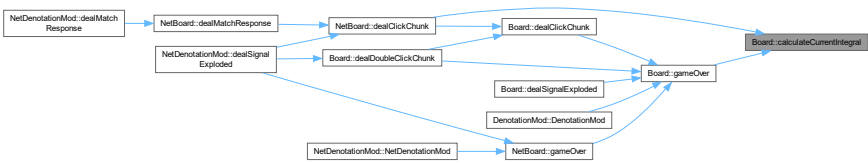
7.2.3.1 calculateCurrentIntegral()

qint32 Board::calculateCurrentIntegral () [protected], [virtual]

Board::calculateCurrentIntegral 计算积分，正相关于探索率，负相关于游戏时长

返回

这是这个函数的调用关系图:



7.2.3.2 dealClickChunk

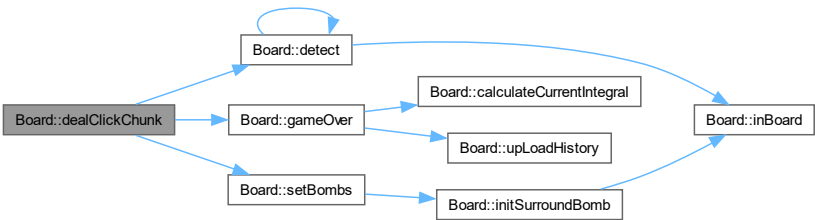
void Board::dealClickChunk (
 Chunk::RowCol rc) [protected], [virtual], [slot]

Board::dealClickChunk 单击事件槽函数

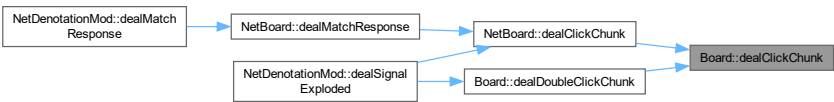
参数

rc	
----	--

函数调用图:



这是这个函数的调用关系图:



7.2.3.3 dealDoubleClickChunk

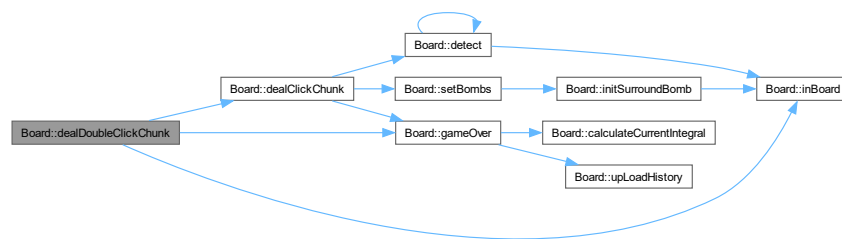
```
void Board::dealDoubleClickChunk (
    Chunk::RowCol rc,
    bool system = false ) [protected], [virtual], [slot]
```

Board::dealDoubleClickChunk 双击事件槽函数

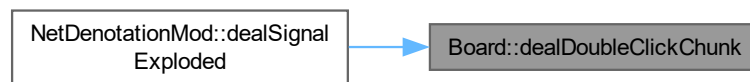
参数

rc	点击处坐标
system	是否是系统调用的

函数调用图:



这是这个函数的调用关系图:



7.2.3.4 dealSignalExploded

```
void Board::dealSignalExploded ( ) [protected], [virtual], [slot]
```

[Board::dealSignalExploded](#) 触雷事件槽函数

函数调用图:



7.2.3.5 detect()

```
void Board::detect (
    Chunk::RowCol rc ) [protected], [virtual]
```

[Board::detect](#) 递归探索周围的雷区

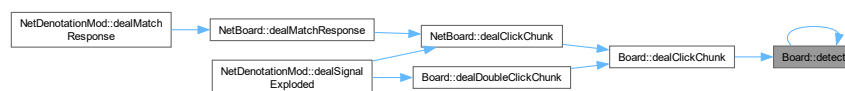
参数

rc	
----	--

函数调用图:



这是这个函数的调用关系图:



7.2.3.6 gameOver()

```
void Board::gameOver (
    QString loseOrWin ) [protected], [virtual]
```

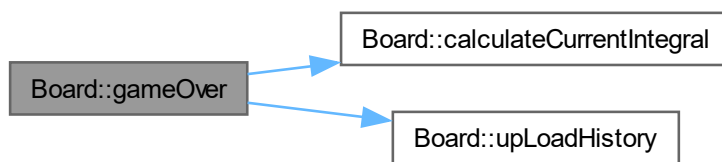
[Board::gameOver](#) 游戏结束槽函数

参数

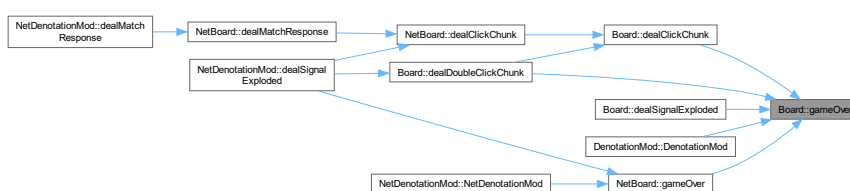
loseOrWin	输赢?
-----------	-----

被 [NetBoard](#) 重载.

函数调用图:



这是这个函数的调用关系图:



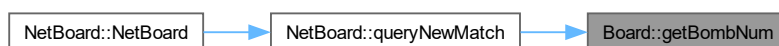
7.2.3.7 getBombNum()

```
qint32 Board::getBombNum ( ) const
```

`Board::getBombNum`

返回

这是这个函数的调用关系图:



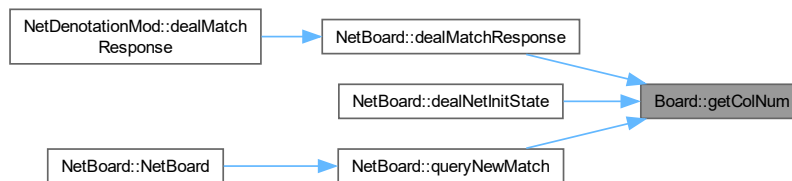
7.2.3.8 getColNum()

```
qint32 Board::getColNum ( ) const
```

[Board::getColNum](#)

返回

这是这个函数的调用关系图:



7.2.3.9 getGameTimer()

```
QPointer< QTimer > Board::getGameTimer ( ) const
```

[Board::getGameTimer](#) 获得游戏计时器对象

返回

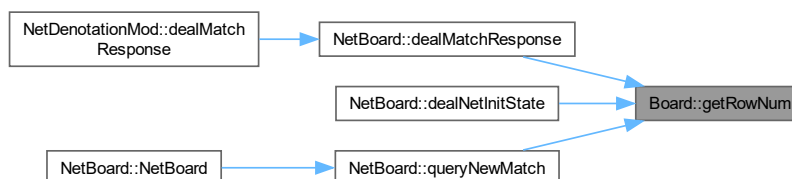
7.2.3.10 getRowNum()

```
qint32 Board::getRowNum ( ) const
```

[Board::getRowNum](#)

返回

这是这个函数的调用关系图:



返回

函数调用图:

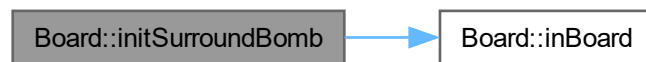


7.2.3.13 initSurroundBomb()

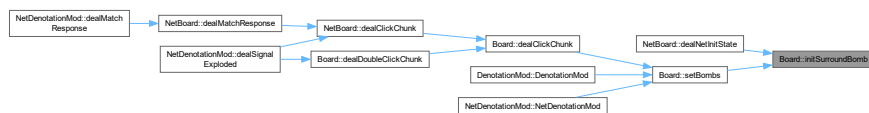
```
void Board::initSurroundBomb ( ) [protected]
```

Board::initSurroundBomb 每个chunk初始化周围的雷数

函数调用图:



这是这个函数的调用关系图:



7.2.3.14 setBombs()

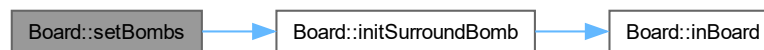
```
void Board::setBombs (
    Chunk::RowCol firstClickedRC ) [protected]
```

Board::setBombs 初始化雷区,若有物理引擎将会调用其来生成随机数

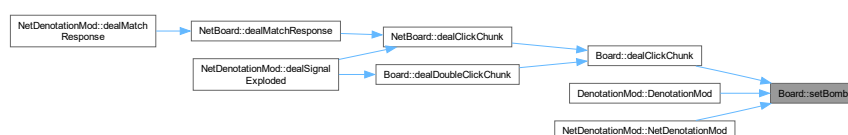
参数

firstClickedRC	第一次点击的坐标
----------------	----------

函数调用图:



这是这个函数的调用关系图:



7.2.3.15 setIsFirstClick()

```
void Board::setIsFirstClick (
    bool newIsFirstClick )
```

[Board::setIsFirstClick](#)

参数

newIsFirstClick	是否是第一次点击
-----------------	----------

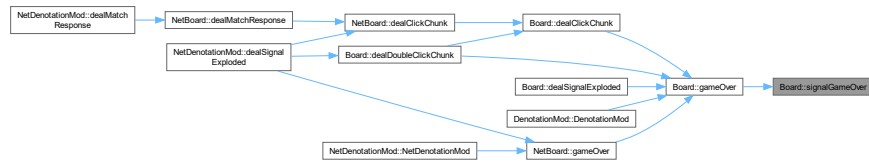
这是这个函数的调用关系图:



7.2.3.16 signalGameOver

```
void Board::signalGameOver ( ) [signal]
```

这是这个函数的调用关系图:



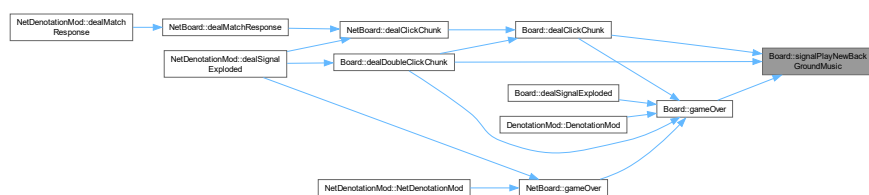
7.2.3.17 signalMove

```
void Board::signalMove ( ) [signal]
```

7.2.3.18 signalPlayNewBackGroundMusic

```
void Board::signalPlayNewBackGroundMusic (
    QString path ) [signal]
```

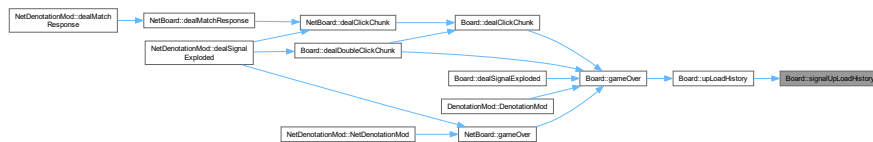
这是这个函数的调用关系图:



7.2.3.19 signalUploadHistory

```
void Board::signalUploadHistory (
    QString gameMod,
    QString rowNum,
    QString colNum,
    QString bombNum,
    QString integral ) [signal]
```

这是这个函数的调用关系图:

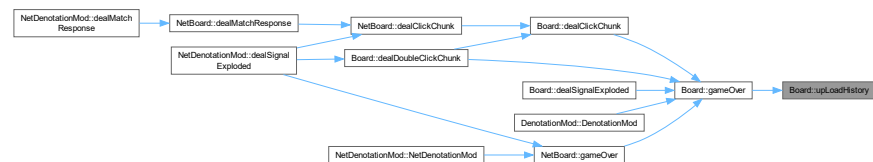


7.2.3.20 uploadHistory()

```
void Board::uploadHistory ( ) [protected], [virtual]
```

[Board::uploadHistory](#) 上传历史记录到服务器

这是这个函数的调用关系图:



7.2.4 友元及相关函数文档

7.2.4.1 DenotationMod

```
friend class DenotationMod [friend]
```

7.2.5 类成员变量说明

7.2.5.1 bombNum

```
const qint32 Board::bombNum [protected]
```

7.2.5.2 chunks

```
QVector<QVector<QPointer<Chunk> > > Board::chunks [protected]
```

7.2.5.3 colNum

```
const qint32 Board::colNum [protected]
```

7.2.5.4 flagBombNum

```
qint32 Board::flagBombNum =0 [protected]
```

7.2.5.5 GameMod

```
const QString Board::GameMod [protected]
```

7.2.5.6 gameTimer

```
QPointer<QTimer> Board::gameTimer [protected]
```

7.2.5.7 isFirstClick

```
bool Board::isFirstClick [protected]
```

7.2.5.8 minedNum

```
qint32 Board::minedNum [protected]
```

7.2.5.9 rowNum

```
const qint32 Board::rowNum [protected]
```

7.2.5.10 selfCurrentIntegral

```
qint32 Board::selfCurrentIntegral =0 [protected]
```

7.2.5.11 SurroundDirection

```
constexpr qint32 Board::SurroundDirection[SurroundDirectionNum][2] [static], [constexpr],  
[protected]
```

初始值:

```
= {  
    {1,0},{0,1},{-1,0},{0,-1},  
    {1,1},{1,-1},{-1,1},{-1,-1}}
```

7.2.5.12 SurroundDirectionNum

```
const qint32 Board::SurroundDirectionNum = 8 [static], [protected]
```

该类的文档由以下文件生成:

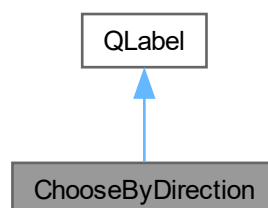
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp

7.3 ChooseByDirection类 参考

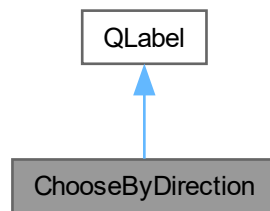
The [ChooseByDirection](#) class 自定义的根据方向键切换模式的控件,支持鼠标点击,会自动获取焦点 T 为可选选项的数据类型

```
#include <ChooseByDirection.h>
```

类 ChooseByDirection 继承关系图:



ChooseByDirection 的协作图:



Public 成员函数

- `ChooseByDirection` (`QWidget *parent`)
`ChooseByDirection::ChooseByDirection`
- virtual void `setItems` (const `QVector< QString >` &newItems)
`ChooseByDirection::setItems` 更新items
- virtual void `addItems` (const `QString` &newItem)
`ChooseByDirection::addItems` 添加item
- virtual const `QString` `getCurrentItems` () const
`ChooseByDirection::getCurrentItems` 返回当前item

7.3.1 详细描述

The `ChooseByDirection` class 自定义的根据方向键切换模式的控件,支持鼠标点击, 会自动获取焦点 `T` 为可选选项的数据类型

7.3.2 构造及析构函数说明

7.3.2.1 ChooseByDirection()

```
ChooseByDirection::ChooseByDirection (  
    QWidget * parent )
```

`ChooseByDirection::ChooseByDirection`

参数

parent	
--------	--

7.3.3 成员函数说明

7.3.3.1 addItem()

```
void ChooseByDirection::addItem (
    const QString & newItem ) [virtual]
```

[ChooseByDirection::addItem](#) 添加item

参数

newItem	
---------	--

7.3.3.2 getCurrentItems()

```
const QString ChooseByDirection::getCurrentItems ( ) const [virtual]
```

[ChooseByDirection::getCurrentItems](#) 返回当前item

返回

7.3.3.3 setItems()

```
void ChooseByDirection::setItems (
    const QVector< QString > & newItems ) [virtual]
```

[ChooseByDirection::setItems](#) 更新items

参数

newItems	
----------	--

该类的文档由以下文件生成:

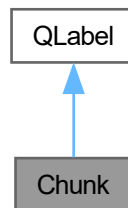
- C:/Users/SJ/Desktop/扫雷/客户端源码/[ChooseByDirection.h](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/[ChooseByDirection.cpp](#)

7.4 Chunk类 参考

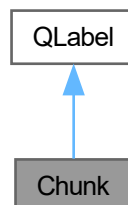
The `Chunk` class 单个块对象

```
#include <Chunk.h>
```

类 `Chunk` 继承关系图:



`Chunk` 的协作图:



类

- struct `RowCol`

Public 类型

- enum class `MineType` { `Bomb` , `NotBomb` }
- enum class `MineState` { `UnMined` , `Mined` , `FlagBomb` , `FlagQuestion` }
- enum class `RIGHT_KEY_MENU` { `FlagBomb` , `FlagQuestion` }

信号

- void `signalExploded ()`
- void `signalClickChunk (Chunk::RowCol rc)`
- void `signalDoubleClickChunk (Chunk::RowCol rc, bool system=false)`
- void `signalFlagBombChanged (qint32 changedNum)`

只有点击不爆炸后才会发出该信号

Public 成员函数

- `Chunk (QWidget *parent=nullptr)`
`Chunk::Chunk`
- virtual void `setRowCol (qint32 row, qint32 col)`
`Chunk::setRowCol` 设置坐标
- const qint32 `getChunkSize ()`
- virtual void `setMineType (const MineType mt)`
`Chunk::setMineType` 设置类型
- virtual `MineType getMineType () const`
`Chunk::getMineType` 返回类型
- virtual qint32 `getSurroundBomb () const`
`Chunk::getSurroundBomb`
- virtual void `setSurroundBomb (qint32 newSurroundBomb)`
`Chunk::setSurroundBomb`
- virtual const `RowCol getRowCol () const`
`Chunk::getRowCol` 返回坐标
- virtual `MineState getMineState () const`
`Chunk::getMineState` 返回探索状态
- virtual void `setMineState (MineState newMineState)`
`Chunk::setMineState`
- virtual void `floatByDoubleClick ()`
`Chunk::floatByDoubleClick` 浮动效果
- virtual void `showBomb ()`
`Chunk::showBomb` 翻开所有雷
- virtual void `openThenShow ()`
`Chunk::openThenShow` 翻开并显示
- void `setPix (const QPixmap &newPix)`
`Chunk::setPix` 设置画布
- const QPixmap & `getPix () const`
`Chunk::getPix` 返回全局画布

7.4.1 详细描述

The `Chunk` class 单个块对象

7.4.2 成员枚举类型说明

7.4.2.1 MineState

```
enum class Chunk::MineState [strong]
```

枚举值

UnMined	
Mined	
FlagBomb	
FlagQuestion	

7.4.2.2 MineType

```
enum class Chunk::MineType [strong]
```

枚举值

Bomb	
NotBomb	

7.4.2.3 RIGHT_KEY_MENU

```
enum class Chunk::RIGHT_KEY_MENU [strong]
```

枚举值

FlagBomb	
FlagQuestion	

7.4.3 构造及析构造函数说明

7.4.3.1 Chunk()

```
Chunk::Chunk (
    QWidget * parent = nullptr ) [explicit]
```

Chunk::Chunk

参数

parent	
--------	--

7.4.4 成员函数说明

7.4.4.1 floatByDoubleClick()

```
void Chunk::floatByDoubleClick ( ) [virtual]
```

[Chunk::floatByDoubleClick](#) 浮动效果

7.4.4.2 getChunkSize()

```
const qint32 Chunk::getChunkSize ( )
```

7.4.4.3 getMineState()

```
Chunk::MineState Chunk::getMineState ( ) const [virtual]
```

[Chunk::getMineState](#) 返回探索状态

返回

这是这个函数的调用关系图:



7.4.4.4 getMineType()

```
Chunk::MineType Chunk::getMineType ( ) const [virtual]
```

Chunk::getMineType 返回类型

返回

这是这个函数的调用关系图:



7.4.4.5 getPix()

```
const QPixmap & Chunk::getPix ( ) const
```

Chunk::getPix 返回全局画布

返回

7.4.4.6 getRowCol()

```
const Chunk::RowCol Chunk::getRowCol ( ) const [virtual]
```

Chunk::getRowCol 返回坐标

返回

7.4.4.7 getSurroundBomb()

```
qint32 Chunk::getSurroundBomb ( ) const [virtual]
```

[Chunk::getSurroundBomb](#)

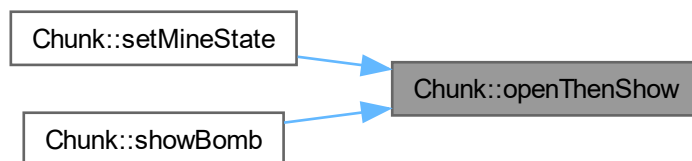
返回

7.4.4.8 openThenShow()

```
void Chunk::openThenShow ( ) [virtual]
```

[Chunk::openThenShow](#) 翻开并显示

这是这个函数的调用关系图:



7.4.4.9 setMineState()

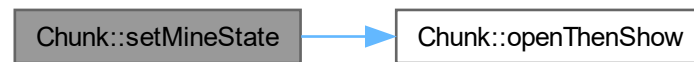
```
void Chunk::setMineState (   
    MineState newMineState ) [virtual]
```

[Chunk::setMineState](#)

参数

<code>newMineState</code>	
---------------------------	--

函数调用图:



7.4.4.10 setMineType()

```
void Chunk::setMineType (
    const MineType mt ) [virtual]
```

[Chunk::setMineType](#) 设置类型

参数

mt	
----	--

7.4.4.11 setPix()

```
void Chunk::setPix (
    const QPixmap & newPix )
```

[Chunk::setPix](#) 设置画布

参数

newPix	
--------	--

7.4.4.12 setRowCol()

```
void Chunk::setRowCol (
    quint32 row,
    quint32 col ) [virtual]
```

[Chunk::setRowCol](#) 设置坐标

参数

row	行坐标
col	列坐标

7.4.4.13 setSurroundBomb()

```
void Chunk::setSurroundBomb (
    qint32 newSurroundBomb ) [virtual]
```

[Chunk::setSurroundBomb](#)

参数

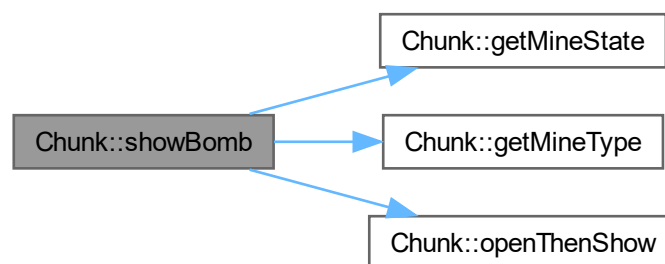
newSurroundBomb	
-----------------	--

7.4.4.14 showBomb()

```
void Chunk::showBomb ( ) [virtual]
```

[Chunk::showBomb](#) 翻开所有雷

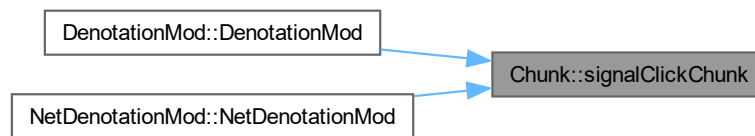
函数调用图:



7.4.4.15 signalClickChunk

```
void Chunk::signalClickChunk (
    Chunk::RowCol rc ) [signal]
```

这是这个函数的调用关系图:



7.4.4.16 signalDoubleClickChunk

```
void Chunk::signalDoubleClickChunk (
    Chunk::RowCol rc,
    bool system = false ) [signal]
```

7.4.4.17 signalExploded

```
void Chunk::signalExploded ( ) [signal]
```

7.4.4.18 signalFlagBombChanged

```
void Chunk::signalFlagBombChanged (
    quint32 changedNum ) [signal]
```

只有点击不爆炸后才会发出该信号

该类的文档由以下文件生成:

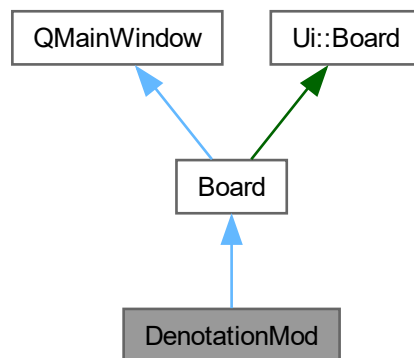
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp

7.5 DenotationMod类 参考

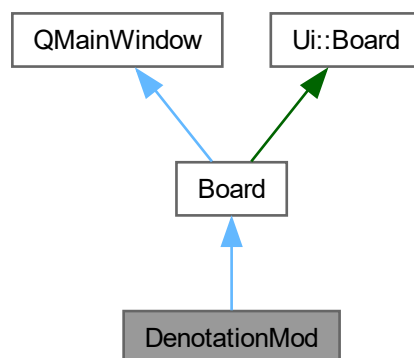
The `DenotationMod` class 爆炸模式，触雷不会死，但点击次数有限

```
#include <DenotationMod.h>
```

类 `DenotationMod` 继承关系图:



`DenotationMod` 的协作图:



Public 成员函数

- `DenotationMod` (`qint32 rowNum`, `qint32 colNum`, `qint32 bombNum`, `QWidget *parent=nullptr`, `QString GameMod="Denotation"`)
`DenotationMod::DenotationMod`

Public 成员函数 继承自 Board

- Board (qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent=nullptr, QString GameMod="Classic")
Board::Board
- ~Board ()
Board::~~Board
- qint32 getRowNum () const
Board::getRowNum
- qint32 getColNum () const
Board::getColNum
- qint32 getBombNum () const
Board::getBombNum
- QPointer< QTimer > getGameTimer () const
Board::getGameTimer 获得游戏计时器对象
- void setIsFirstClick (bool newIsFirstClick)
Board::setIsFirstClick

额外继承的成员函数

信号 继承自 Board

- void signalGameOver ()
- void signalPlayNewBackGroundMusic (QString path)
- void signalMove ()
- void signalUpLoadHistory (QString gameMod, QString rowNum, QString colNum, QString bombNum, QString integral)

Protected 槽 继承自 Board

- virtual void dealSignalExploded ()
Board::dealSignalExploded 触雷事件槽函数
- virtual void dealClickChunk (Chunk::RowCol rc)
Board::dealClickChunk 单击事件槽函数
- virtual void dealDoubleClickChunk (Chunk::RowCol rc, bool system=false)
Board::dealDoubleClickChunk 双击事件槽函数

Protected 成员函数 继承自 Board

- void setBombs (Chunk::RowCol firstClickedRC)
Board::setBombs 初始化雷区,若有物理引擎将会调用其来生成随机数
- virtual void detect (Chunk::RowCol rc)
Board::detect 递归探索周围的雷区
- void initSurroundBomb ()
Board::initSurroundBomb 每个chunk初始化周围的雷数
- bool inBoard (Chunk::RowCol rc)
Board::inBoard 判断是否在棋盘内部
- bool inBoard (qint32 row, qint32 col)
Board::inBoard 重载
- virtual void gameOver (QString loseOrWin)
Board::gameOver 游戏结束槽函数
- virtual void upLoadHistory ()
Board::upLoadHistory 上传历史记录到服务器
- virtual qint32 calculateCurrentIntegral ()
Board::calculateCurrentIntegral 计算积分, 正相关于探索率, 负相关于游戏时长

Protected 属性 继承自 [Board](#)

- const QString [GameMod](#)
- QVector< QVector< QPointer< [Chunk](#) > > > [chunks](#)
- const qint32 [rowNum](#)
- const qint32 [colNum](#)
- const qint32 [bombNum](#)
- qint32 [flagBombNum](#) =0
- qint32 [minedNum](#)
- QPointer< QTimer > [gameTimer](#)
- bool [isFirstClick](#)
- qint32 [selfCurrentIntegral](#) =0

静态 Protected 属性 继承自 [Board](#)

- static const qint32 [SurroundDirectionNum](#) = 8
- static constexpr qint32 [SurroundDirection](#) [[SurroundDirectionNum](#)][2]

7.5.1 详细描述

The [DenotationMod](#) class 爆炸模式，触雷不会死，但点击次数有限

7.5.2 构造及析构造函数说明

7.5.2.1 DenotationMod()

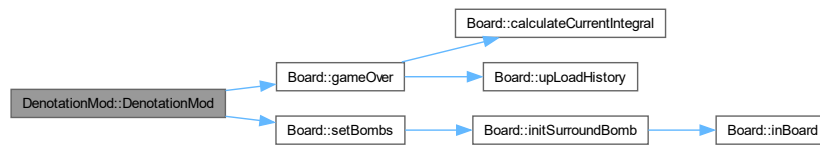
```
DenotationMod::DenotationMod (
    qint32 rowNum,
    qint32 colNum,
    qint32 bombNum,
    QWidget * parent = nullptr,
    QString GameMod = "Denotation" )
```

[DenotationMod::DenotationMod](#)

参数

rowNum	
colNum	
bombNum	
parent	
GameMod	

函数调用图:



该类的文档由以下文件生成:

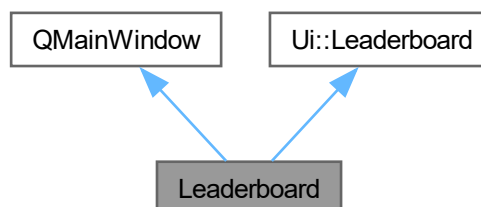
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp

7.6 Leaderboard类 参考

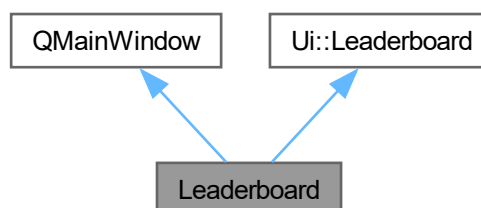
The [Leaderboard](#) class 显示历史战绩

```
#include <Leaderboard.h>
```

类 Leaderboard 继承关系图:



Leaderboard 的协作图:



Public 槽

- virtual void [dealMainSocketNewRecvMessage](#) (QByteArray mes)
[Leaderboard::dealMainSocketNewRecvMessage](#) 接受mainWindow转发的服务器传过来的消息

Public 成员函数

- [Leaderboard](#) (QTcpSocket *socket, QWidget *parent=nullptr)
[Leaderboard::Leaderboard](#)

友元

- class [Packet](#)< [Leaderboard](#) >

7.6.1 详细描述

The [Leaderboard](#) class 显示历史战绩

7.6.2 构造及析构造函数说明

7.6.2.1 Leaderboard()

```
Leaderboard::Leaderboard (  
    QTcpSocket * socket,  
    QWidget * parent = nullptr ) [explicit]
```

[Leaderboard::Leaderboard](#)

参数

socket	
parent	

7.6.3 成员函数说明

7.6.3.1 dealMainSocketNewRecvMessage

```
void Leaderboard::dealMainSocketNewRecvMessage (  
    QByteArray mes ) [virtual], [slot]
```

[Leaderboard::dealMainSocketNewRecvMessage](#) 接受mainWindow转发的服务器传过来的消息

参数

mes	
-----	--

函数调用图:



7.6.4 友元及相关函数文档

7.6.4.1 Packet< Leaderboard >

```
friend class Packet< Leaderboard > [friend]
```

该类的文档由以下文件生成:

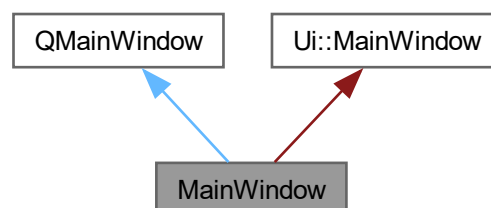
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp

7.7 MainWindow类 参考

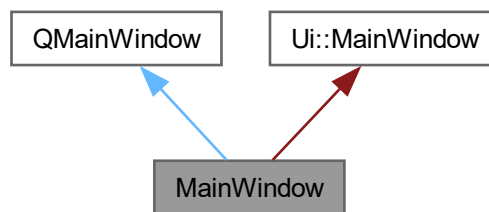
The `MainWindow` class 主窗口对象

```
#include <MainWindow.h>
```

类 `MainWindow` 继承关系图:



MainWindow 的协作图:



信号

- void [signalMainSocketNewRecvMessage](#) (QByteArray mes)

Public 成员函数

- [MainWindow](#) (QWidget *parent=nullptr)
[MainWindow::MainWindow](#)
- [~MainWindow](#) ()
[MainWindow::~~MainWindow](#)

友元

- class [Packet](#) < [MainWindow](#) >

7.7.1 详细描述

The [MainWindow](#) class 主窗口对象

7.7.2 构造及析构函数说明

7.7.2.1 MainWindow()

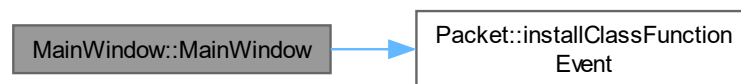
```
MainWindow::MainWindow (
    QWidget * parent = nullptr )
```

[MainWindow::MainWindow](#)

参数

parent	
--------	--

函数调用图:



7.7.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

[MainWindow::~MainWindow](#)

7.7.3 成员函数说明

7.7.3.1 signalMainSocketNewRecvMessage

```
void MainWindow::signalMainSocketNewRecvMessage (
    QByteArray mes ) [signal]
```

7.7.4 友元及相关函数文档

7.7.4.1 Packet< MainWindow >

```
friend class Packet< MainWindow > [friend]
```

该类的文档由以下文件生成:

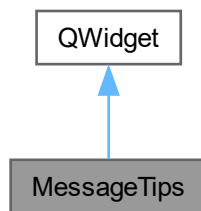
- C:/Users/SJ/Desktop/扫雷/客户端源码/[MainWindow.h](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/[MainWindow.cpp](#)

7.8 MessageTips类 参考

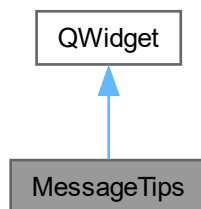
The [MessageTips](#) class 实现自动消失的消息框,由于时间原因,此代码借鉴于[csdn](#) 必须使用指针类型或者指定父对象 @my doing 添加了对qt6的兼容 添加关闭时自动删除, 释放资源,避免内存泄露

```
#include <MessageTips.h>
```

类 MessageTips 继承关系图:



MessageTips 的协作图:



Public 成员函数

- [MessageTips](#) (QString showStr="none", QWidget *parent=nullptr)
[MessageTips::MessageTips](#)
- [~MessageTips](#) ()
[MessageTips::~~MessageTips](#)
- double [getOpacityValue](#) () const
[MessageTips::getOpacityValue](#)
- void [setOpacityValue](#) (double value)
[MessageTips::setOpacityValue](#)
- qint32 [getTextSize](#) () const
[MessageTips::getTextSize](#)

- void [setTextSize](#) (int value)
[MessageTips::setTextSize](#)
- QColor [getTextColor](#) () const
[MessageTips::getTextColor](#)
- void [setTextColor](#) (const QColor &value)
[MessageTips::setTextColor](#)
- QColor [getBackgroundColor](#) () const
[MessageTips::getBackgroundColor](#)
- void [setBackgroundColor](#) (const QColor &value)
[MessageTips::setBackgroundColor](#)
- QColor [getFrameColor](#) () const
[MessageTips::getFrameColor](#)
- void [setFrameColor](#) (const QColor &value)
[MessageTips::setFrameColor](#)
- qint32 [getFrameSize](#) () const
[MessageTips::getFrameSize](#)
- void [setFrameSize](#) (int value)
[MessageTips::setFrameSize](#)
- qint32 [getShowTime](#) () const
[MessageTips::getShowTime](#)
- void [setShowTime](#) (int msec)
[MessageTips::setShowTime](#)
- void [setCloseTimeSpeed](#) (int closeTime=100, double closeSpeed=0.1)
[MessageTips::setCloseTimeSpeed](#) 设置关闭的时间和速度, speed大小限定0~1

Protected 成员函数

- void [paintEvent](#) (QPaintEvent *event) override
[MessageTips::paintEvent](#)

7.8.1 详细描述

The [MessageTips](#) class 实现自动消失的消息框,由于时间原因,此代码借鉴于csdn 必须使用指针类型或者指定父对象 @my doing 添加了对qt6的兼容 添加关闭时自动删除, 释放资源,避免内存泄露

[待办事项](#) 添加move功能 改成更好看的样式

7.8.2 构造及析构函数说明

7.8.2.1 MessageTips()

```
MessageTips::MessageTips (
    QString showStr = "none",
    QWidget * parent = nullptr ) [explicit]
```

[MessageTips::MessageTips](#)

参数

showStr	显示的文字
parent	父对象

7.8.2.2 ~MessageTips()

```
MessageTips::~~MessageTips ( )
```

[MessageTips::~~MessageTips](#)

7.8.3 成员函数说明

7.8.3.1 getBackgroundColor()

```
QColor MessageTips::getBackgroundColor ( ) const
```

[MessageTips::getBackgroundColor](#)

返回

7.8.3.2 getFrameColor()

```
QColor MessageTips::getFrameColor ( ) const
```

[MessageTips::getFrameColor](#)

返回

7.8.3.3 getFrameSize()

```
int MessageTips::getFrameSize ( ) const
```

[MessageTips::getFrameSize](#)

返回

7.8.3.4 getOpacityValue()

```
double MessageTips::getOpacityValue ( ) const
```

[MessageTips::getOpacityValue](#)

返回

7.8.3.5 getShowTime()

```
int MessageTips::getShowTime ( ) const
```

[MessageTips::getShowTime](#)

返回

7.8.3.6 getTextColor()

```
QColor MessageTips::getTextColor ( ) const
```

[MessageTips::getTextColor](#)

返回

7.8.3.7 getTextSize()

```
int MessageTips::getTextSize ( ) const
```

[MessageTips::getTextSize](#)

返回

7.8.3.8 paintEvent()

```
void MessageTips::paintEvent (
    QPaintEvent * event ) [override], [protected]
```

[MessageTips::paintEvent](#)

参数

event	
-------	--

7.8.3.9 setBackgroundColor()

```
void MessageTips::setBackgroundColor (
    const QColor & value )
```

[MessageTips::setBackgroundColor](#)

参数

value	
-------	--

7.8.3.10 setCloseTimeSpeed()

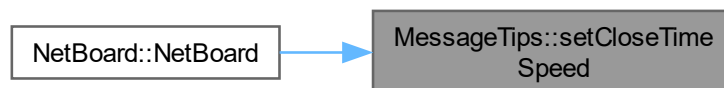
```
void MessageTips::setCloseTimeSpeed (
    int closeTime = 100,
    double closeSpeed = 0.1 )
```

[MessageTips::setCloseTimeSpeed](#) 设置关闭的时间和速度，**speed**大小限定0~1

参数

closeTime	
closeSpeed	

这是这个函数的调用关系图:



7.8.3.11 setFrameColor()

```
void MessageTips::setFrameColor (
    const QColor & value )
```

[MessageTips::setFrameColor](#)

参数

value	
-------	--

7.8.3.12 setFrameSize()

```
void MessageTips::setFrameSize (
    int value )
```

[MessageTips::setFrameSize](#)

参数

value	
-------	--

7.8.3.13 setOpacityValue()

```
void MessageTips::setOpacityValue (
```

```
double value )
```

MessageTips::setOpacityValue

参数

value	
-------	--

7.8.3.14 setShowTime()

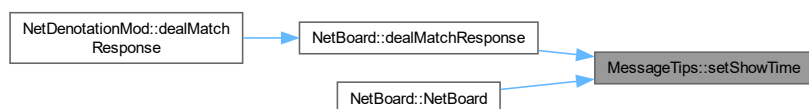
```
void MessageTips::setShowTime (
    int value )
```

MessageTips::setShowTime

参数

value	
-------	--

这是这个函数的调用关系图:



7.8.3.15 setTextColor()

```
void MessageTips::setTextColor (
    const QColor & value )
```

MessageTips::setTextColor

参数

value	
-------	--

7.8.3.16 setTextSize()

```
void MessageTips::setTextSize (
    int value )
```

[MessageTips::setTextSize](#)

参数

value	
-------	--

该类的文档由以下文件生成:

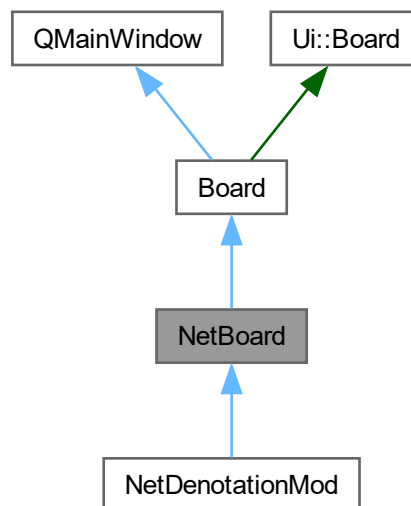
- [C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h](#)
- [C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp](#)

7.9 NetBoard类 参考

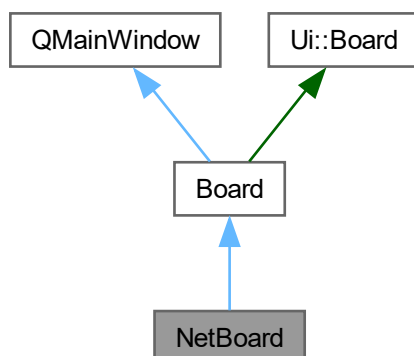
The [NetBoard](#) class 经典模式的网络对战

```
#include <NetBoard.h>
```

类 NetBoard 继承关系图:



NetBoard 的协作图:



Public 槽

- virtual void [dealMainSocketNewRecvMessage](#) (QByteArray mes)
[NetBoard::dealMainSocketNewRecvMessage](#) 接受mainWindow转发的消息
- virtual void [dealNetInitState](#) (QStringList list)
[NetBoard::dealNetInitState](#)
- virtual void [dealUpdateAntiIntegral](#) (QStringList list)
[NetBoard::dealUpdateAntiIntegral](#) 更新对手的积分
- virtual void [dealAntiGameOver](#) (QStringList list)
[NetBoard::dealAntiGameOver](#) 处理对手游戏结束事件

Public 成员函数

- [NetBoard](#) (QTcpSocket *socket, qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent=nullptr, QString GameMod="NetBoard")
[NetBoard::NetBoard](#)

Public 成员函数 继承自 Board

- [Board](#) (qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent=nullptr, QString GameMod="Classic")
[Board::Board](#)
- [~Board](#) ()
[Board::~~Board](#)
- qint32 [getRowNum](#) () const
[Board::getRowNum](#)
- qint32 [getColNum](#) () const
[Board::getColNum](#)
- qint32 [getBombNum](#) () const
[Board::getBombNum](#)
- QPointer< QTimer > [getGameTimer](#) () const
[Board::getGameTimer](#) 获得游戏计时器对象
- void [setIsFirstClick](#) (bool newIsFirstClick)
[Board::setIsFirstClick](#)

Protected 槽

- virtual void [dealMatchResponse](#) (QStringList list)
[NetBoard::dealMatchResponse](#) 处理匹配结果
- virtual void [dealClickChunk](#) (Chunk::RowCol rc) override
[NetBoard::dealClickChunk](#) 重载单击事件

Protected 槽 继承自 Board

- virtual void [dealSignalExploded](#) ()
[Board::dealSignalExploded](#) 触雷事件槽函数
- virtual void [dealClickChunk](#) (Chunk::RowCol rc)
[Board::dealClickChunk](#) 单击事件槽函数
- virtual void [dealDoubleClickChunk](#) (Chunk::RowCol rc, bool system=false)
[Board::dealDoubleClickChunk](#) 双击事件槽函数

Protected 成员函数

- virtual void [gameOver](#) (QString loseOrWin) override
[NetBoard::gameOver](#) 重载游戏结束事件。
- virtual void [sendIntegralToServer](#) (qint32 integral)
[NetBoard::sendIntegralToServer](#) 发送自己的积分到服务器
- virtual void [queryNewMatch](#) ()
[NetBoard::queryNewMatch](#) 向服务器请求新的匹配

Protected 成员函数 继承自 Board

- void [setBombs](#) (Chunk::RowCol firstClickedRC)
[Board::setBombs](#) 初始化雷区,若有物理引擎将会调用其来生成随机数
- virtual void [detect](#) (Chunk::RowCol rc)
[Board::detect](#) 递归探索周围的雷区
- void [initSurroundBomb](#) ()
[Board::initSurroundBomb](#) 每个chunk初始化周围的雷数
- bool [inBoard](#) (Chunk::RowCol rc)
[Board::inBoard](#) 判断是否在棋盘内部
- bool [inBoard](#) (qint32 row, qint32 col)
[Board::inBoard](#) 重载
- virtual void [gameOver](#) (QString loseOrWin)
[Board::gameOver](#) 游戏结束槽函数
- virtual void [uploadHistory](#) ()
[Board::uploadHistory](#) 上传历史记录到服务器
- virtual qint32 [calculateCurrentIntegral](#) ()
[Board::calculateCurrentIntegral](#) 计算积分, 正相关于探索率, 负相关于游戏时长

友元

- class [Packet](#)< [NetBoard](#) >

额外继承的成员函数

信号 继承自 [Board](#)

- void [signalGameOver](#) ()
- void [signalPlayNewBackGroundMusic](#) (QString path)
- void [signalMove](#) ()
- void [signalUploadHistory](#) (QString gameMod, QString [rowNum](#), QString [colNum](#), QString [bombNum](#), QString integral)

Protected 属性 继承自 [Board](#)

- const QString [GameMod](#)
- QVector< QVector< QPointer< [Chunk](#) > > > [chunks](#)
- const qint32 [rowNum](#)
- const qint32 [colNum](#)
- const qint32 [bombNum](#)
- qint32 [flagBombNum](#) =0
- qint32 [minedNum](#)
- QPointer< QTimer > [gameTimer](#)
- bool [isFirstClick](#)
- qint32 [selfCurrentIntegral](#) =0

静态 Protected 属性 继承自 [Board](#)

- static const qint32 [SurroundDirectionNum](#) = 8
- static constexpr qint32 [SurroundDirection](#) [[SurroundDirectionNum](#)][2]

7.9.1 详细描述

The [NetBoard](#) class 经典模式的网络对战

7.9.2 构造及析构函数说明

7.9.2.1 NetBoard()

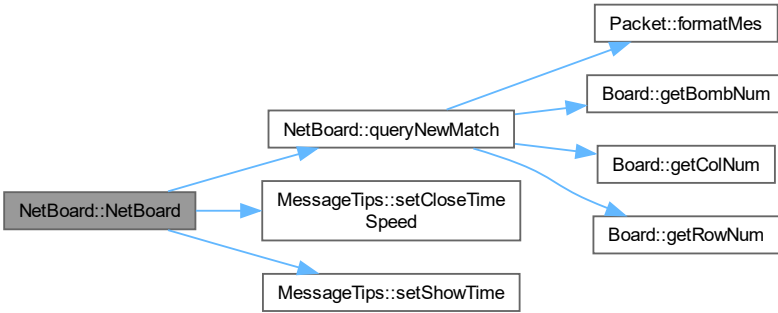
```
NetBoard::NetBoard (
    QTcpSocket * socket,
    qint32 rowNum,
    qint32 colNum,
    qint32 bombNum,
    QWidget * parent = nullptr,
    QString GameMod = "NetBoard" )
```

[NetBoard::NetBoard](#)

参数

socket	
rowNum	
colNum	
bombNum	
parent	
GameMod	

/note:暂不处理连接成功的事件,后期若要添加断线重连时可以加上,(当然，也可以让MainWindow来处理。。。以后再说) connect(this->socket,QTcpSocket::connected,); / / note:暂不处理，理由同上
connect(this->socket,&QTcpSocket::disconnected); note:此处的消息在MainWindow读取后已经被清空了
connect(this->socket,&QTcpSocket::readyRead,[&]{ QString newMes = this->socket->readAll(); dout<<newMes; this->packet.pushMessage(newMes); });函数调用图:



7.9.3 成员函数说明

7.9.3.1 dealAntiGameOver

```
void NetBoard::dealAntiGameOver (
    QStringList list ) [virtual], [slot]
```

NetBoard::dealAntiGameOver 处理对手游戏结束事件

参数

list	
------	--

7.9.3.2 dealClickChunk

```
void NetBoard::dealClickChunk (
```

```
Chunk::RowCol rc ) [override], [protected], [virtual], [slot]
```

NetBoard::dealClickChunk 重载单击事件

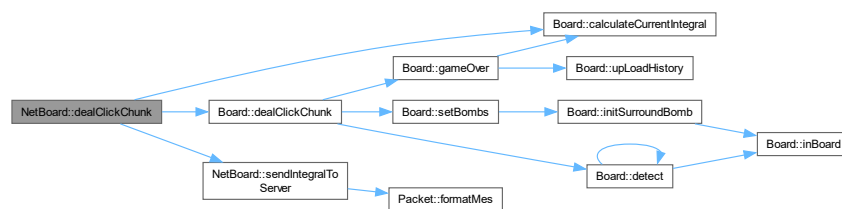
参数

rc	
----	--

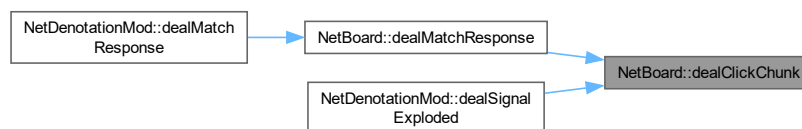
注解

:玄学改bug法,如若不这样,“后手”方无法正常init,(慢一步)

函数调用图:



这是这个函数的调用关系图:



7.9.3.3 dealMainSocketNewRecvMessage

```
void NetBoard::dealMainSocketNewRecvMessage (
    QByteArray mes ) [virtual], [slot]
```

NetBoard::dealMainSocketNewRecvMessage 接受mainWindow转发的消息

参数

mes	
-----	--

函数调用图:



7.9.3.4 dealMatchResponse

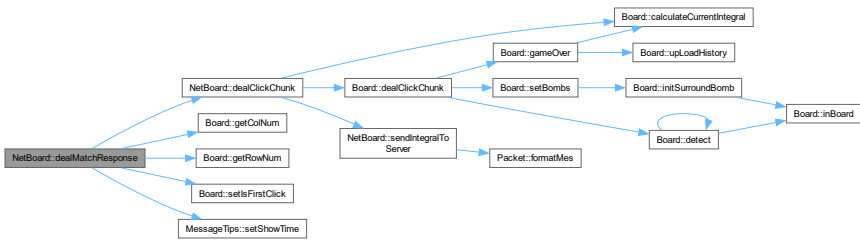
```
void NetBoard::dealMatchResponse (
    QStringList list ) [protected], [virtual], [slot]
```

NetBoard::dealMatchResponse 处理匹配结果

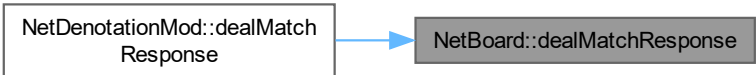
参数

list	length :2 {1/0}{email}
------	------------------------

函数调用图:



这是这个函数的调用关系图:



7.9.3.5 dealNetInitState

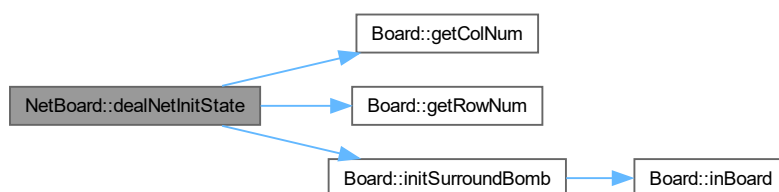
```
void NetBoard::dealNetInitState (
    QStringList list ) [virtual], [slot]
```

[NetBoard::dealNetInitState](#)

参数

list	length: {1}{12* rowNum * colNum in lastGamMod split by /}
------	---

函数调用图:



7.9.3.6 dealUpdateAntiIntegral

```
void NetBoard::dealUpdateAntiIntegral (
    QStringList list ) [virtual], [slot]
```

[NetBoard::dealUpdateAntiIntegral](#) 更新对手的积分

参数

list	length:1{}
------	------------

7.9.3.7 gameOver()

```
void NetBoard::gameOver (
    QString loseOrWin ) [override], [protected], [virtual]
```

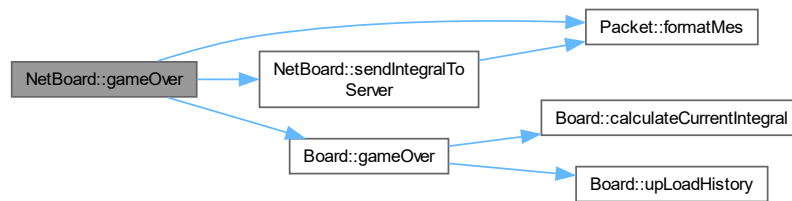
[NetBoard::gameOver](#) 重载游戏结束事件,

参数

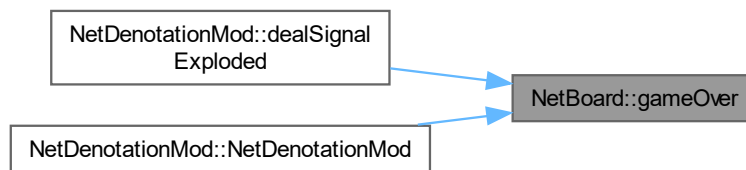
loseOrWin	
-----------	--

重载 [Board](#) .

函数调用图:



这是这个函数的调用关系图:

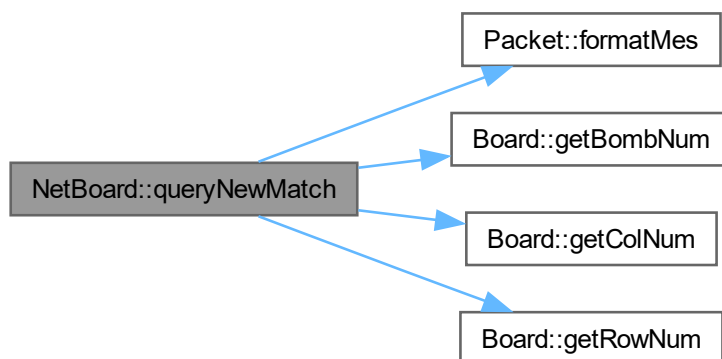


7.9.3.8 queryNewMatch()

```
void NetBoard::queryNewMatch ( ) [protected], [virtual]
```

[NetBoard::queryNewMatch](#) 向服务器请求新的匹配

函数调用图:



这是这个函数的调用关系图:



7.9.3.9 sendIntegralToServer()

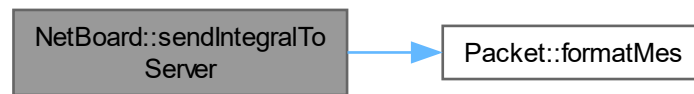
```
void NetBoard::sendIntegralToServer (
    qint32 integral ) [protected], [virtual]
```

[NetBoard::sendIntegralToServer](#) 发送自己的积分到服务器

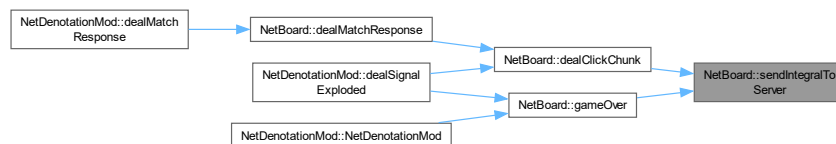
参数

integral	
----------	--

函数调用图:



这是这个函数的调用关系图:



7.9.4 友元及相关函数文档

7.9.4.1 Packet< NetBoard >

```
friend class Packet< NetBoard > [friend]
```

该类的文档由以下文件生成:

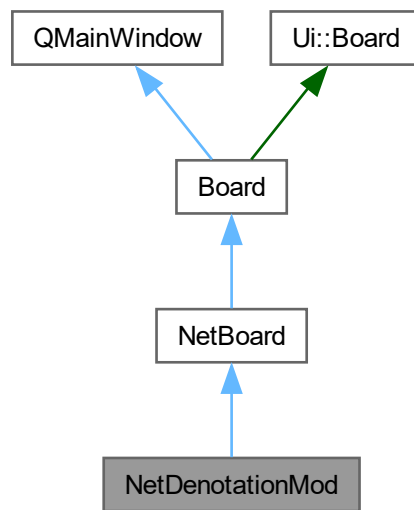
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp

7.10 NetDenotationMod类 参考

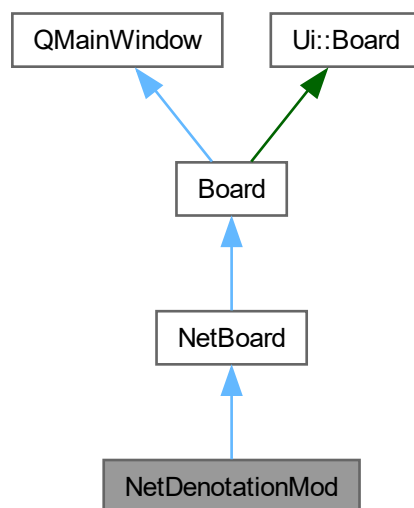
The `NetDenotationMod` class 爆炸模式网络对战

```
#include <NetDenotationMod.h>
```

类 NetDenotationMod 继承关系图:



NetDenotationMod 的协作图:



Public 成员函数

- [NetDenotationMod](#) (QTcpSocket *socket, qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent=nullptr, QString GameMod="NetDenotation")
[NetDenotationMod::NetDenotationMod](#)

Public 成员函数 继承自 [NetBoard](#)

- [NetBoard](#) (QTcpSocket *socket, qint32 [rowNum](#), qint32 [colNum](#), qint32 [bombNum](#), QWidget *parent=nullptr, QString [GameMod](#)="NetBoard")
[NetBoard::NetBoard](#)

Public 成员函数 继承自 [Board](#)

- [Board](#) (qint32 [rowNum](#), qint32 [colNum](#), qint32 [bombNum](#), QWidget *parent=nullptr, QString [GameMod](#)="Classic")
[Board::Board](#)
- [~Board](#) ()
[Board::~~Board](#)
- qint32 [getRowNum](#) () const
[Board::getRowNum](#)
- qint32 [getColNum](#) () const
[Board::getColNum](#)
- qint32 [getBombNum](#) () const
[Board::getBombNum](#)
- QPointer< QTimer > [getGameTimer](#) () const
[Board::getGameTimer](#) 获得游戏计时器对象
- void [setIsFirstClick](#) (bool [newIsFirstClick](#))
[Board::setIsFirstClick](#)

Protected 槽

- virtual void [dealSignalExploded](#) () override
[NetDenotationMod::dealSignalExploded](#)
- virtual void [dealMatchResponse](#) (QStringList list) override
[NetDenotationMod::dealMatchResponse](#)

Protected 槽 继承自 [NetBoard](#)

- virtual void [dealMatchResponse](#) (QStringList list)
[NetBoard::dealMatchResponse](#) 处理匹配结果
- virtual void [dealClickChunk](#) ([Chunk::RowCol](#) rc) override
[NetBoard::dealClickChunk](#) 重载单击事件

Protected 槽 继承自 [Board](#)

- virtual void [dealSignalExploded](#) ()
[Board::dealSignalExploded](#) 触雷事件槽函数
- virtual void [dealClickChunk](#) ([Chunk::RowCol](#) rc)
[Board::dealClickChunk](#) 单击事件槽函数
- virtual void [dealDoubleClickChunk](#) ([Chunk::RowCol](#) rc, bool system=false)
[Board::dealDoubleClickChunk](#) 双击事件槽函数

友元

- class [Packet](#)< [NetDenotationMod](#) >

额外继承的成员函数

Public 槽 继承自 [NetBoard](#)

- virtual void [dealMainSocketNewRecvMessage](#) (QByteArray mes)
[NetBoard::dealMainSocketNewRecvMessage](#) 接受mainWindow转发的消息
- virtual void [dealNetInitState](#) (QStringList list)
[NetBoard::dealNetInitState](#)
- virtual void [dealUpdateAntiIntegral](#) (QStringList list)
[NetBoard::dealUpdateAntiIntegral](#) 更新对手的积分
- virtual void [dealAntiGameOver](#) (QStringList list)
[NetBoard::dealAntiGameOver](#) 处理对手游戏结束事件

信号 继承自 [Board](#)

- void [signalGameOver](#) ()
- void [signalPlayNewBackGroundMusic](#) (QString path)
- void [signalMove](#) ()
- void [signalUpLoadHistory](#) (QString gameMod, QString [rowNum](#), QString [colNum](#), QString [bombNum](#), QString integral)

Protected 成员函数 继承自 [NetBoard](#)

- virtual void [gameOver](#) (QString loseOrWin) override
[NetBoard::gameOver](#) 重载游戏结束事件,
- virtual void [sendIntegralToServer](#) (qint32 integral)
[NetBoard::sendIntegralToServer](#) 发送自己的积分到服务器
- virtual void [queryNewMatch](#) ()
[NetBoard::queryNewMatch](#) 向服务器请求新的匹配

Protected 成员函数 继承自 [Board](#)

- void [setBombs](#) (Chunk::RowCol firstClickedRC)
[Board::setBombs](#) 初始化雷区,若有物理引擎将会调用其来生成随机数
- virtual void [detect](#) (Chunk::RowCol rc)
[Board::detect](#) 递归探索周围的雷区
- void [initSurroundBomb](#) ()
[Board::initSurroundBomb](#) 每个chunk初始化周围的雷数
- bool [inBoard](#) (Chunk::RowCol rc)
[Board::inBoard](#) 判断是否在棋盘内部
- bool [inBoard](#) (qint32 row, qint32 col)
[Board::inBoard](#) 重载
- virtual void [gameOver](#) (QString loseOrWin)
[Board::gameOver](#) 游戏结束槽函数
- virtual void [upLoadHistory](#) ()
[Board::upLoadHistory](#) 上传历史记录到服务器
- virtual qint32 [calculateCurrentIntegral](#) ()
[Board::calculateCurrentIntegral](#) 计算积分, 正相关于探索率, 负相关于游戏时长

Protected 属性 继承自 [Board](#)

- const QString [GameMod](#)
- QVector< QVector< QPointer< [Chunk](#) > > > [chunks](#)
- const qint32 [rowNum](#)
- const qint32 [colNum](#)
- const qint32 [bombNum](#)
- qint32 [flagBombNum](#) =0
- qint32 [minedNum](#)
- QPointer< QTimer > [gameTimer](#)
- bool [isFirstClick](#)
- qint32 [selfCurrentIntegral](#) =0

静态 Protected 属性 继承自 [Board](#)

- static const qint32 [SurroundDirectionNum](#) = 8
- static constexpr qint32 [SurroundDirection](#) [[SurroundDirectionNum](#)][2]

7.10.1 详细描述

The [NetDenotationMod](#) class 爆炸模式网络对战

7.10.2 构造及析构造函数说明

7.10.2.1 NetDenotationMod()

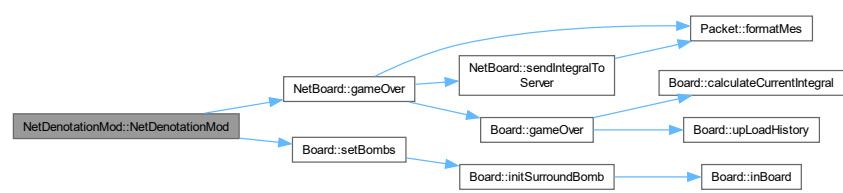
```
NetDenotationMod::NetDenotationMod (
    QTcpSocket * socket,
    qint32 rowNum,
    qint32 colNum,
    qint32 bombNum,
    QWidget * parent = nullptr,
    QString GameMod = "NetDenotation" )
```

[NetDenotationMod::NetDenotationMod](#)

参数

socket	
rowNum	
colNum	
bombNum	
parent	
GameMod	

函数调用图:



7.10.3 成员函数说明

7.10.3.1 dealMatchResponse

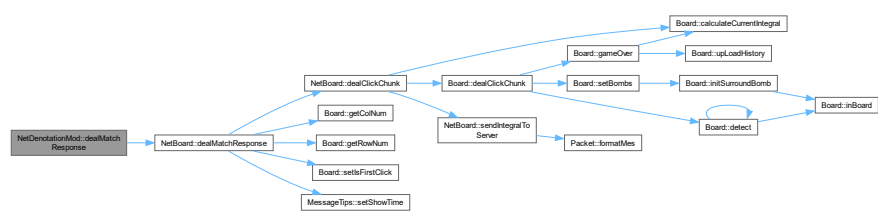
```
void NetDenotationMod::dealMatchResponse (
    QStringList list ) [override], [protected], [virtual], [slot]
```

NetDenotationMod::dealMatchResponse

参数

list	
------	--

函数调用图:

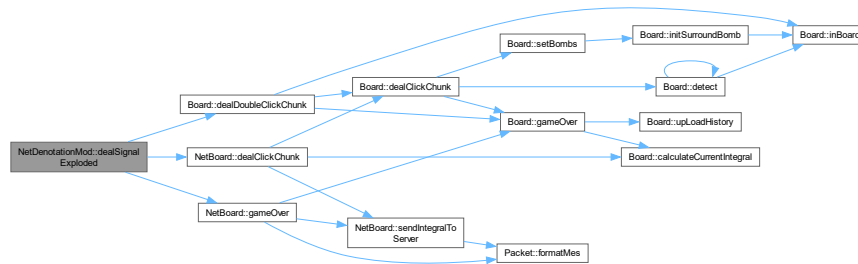


7.10.3.2 dealSignalExploded

```
void NetDenotationMod::dealSignalExploded ( ) [override], [protected], [virtual], [slot]
```

NetDenotationMod::dealSignalExploded

函数调用图:



7.10.4 友元及相关函数文档

7.10.4.1 Packet< NetDenotationMod >

```
friend class Packet< NetDenotationMod > [friend]
```

该类的文档由以下文件生成:

- C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.cpp

7.11 Packet< T > 模板类 参考

用于socket协议的信息封装和解包，可以绑定信息-回调函数,Packet.cpp和Packet.h都得放在头文件中(-l Packet.cpp Packet.h) 如果要绑定私有行为，应该将Packet<T>声明为友元 T为parent对应的类名,installClassFunctionEvent 会在触发时调用parent的成员函数 所有要绑定的函数都应该以void为返回值,QStringList为参数

```
#include <Packet.h>
```

Public 成员函数

- Packet (T *parent)
- virtual void pushMessage (QString newMes)
- virtual QString formatMes (QStringList newMesList)
- virtual QString formatMes (QString newMes)
- virtual void installClassFunctionEvent (QString funcName, qint32 parameterNum, void(T::*callBack)(QStringList))

7.11.1 详细描述

```
template<typename T>
class Packet< T >
```

用于socket协议的信息封装和解包，可以绑定信息-回调函数,Packet.cpp和Packet.h都得放在头文件中(-I Packet.cpp Packet.h) 如果要绑定私有行为，应该将Packet<T>声明为友元 T为parent对应的类名,installClassFunctionEvent 会在触发时调用parent的成员函数 所有要绑定的函数都应该以void为返回值,QStringList为参数

7.11.2 构造及析构函数说明

7.11.2.1 Packet()

```
template<typename T >
Packet< T >::Packet (
    T * parent )
```

7.11.3 成员函数说明

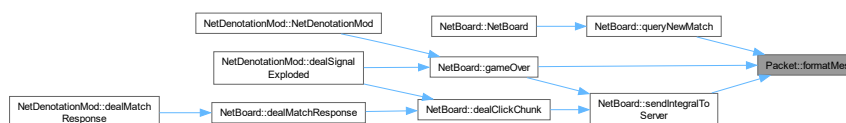
7.11.3.1 formatMes() [1/2]

```
template<typename T >
QString Packet< T >::formatMes (
    QString newMes ) [virtual]
```

7.11.3.2 formatMes() [2/2]

```
template<typename T >
QString Packet< T >::formatMes (
    QStringList newMesList ) [virtual]
```

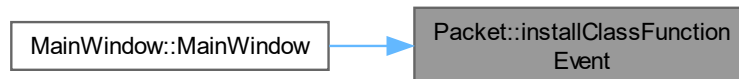
这是这个函数的调用关系图:



7.11.3.3 installClassFunctionEvent()

```
template<typename T >
void Packet< T >::installClassFunctionEvent (
    QString funcName,
    quint32 parameterNum,
    void(T::*)(QStringList) callBack ) [virtual]
```

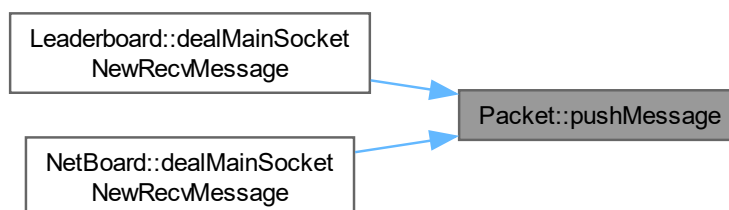
这是这个函数的调用关系图:



7.11.3.4 pushMessage()

```
template<typename T >
void Packet< T >::pushMessage (
    QString newMes ) [virtual]
```

这是这个函数的调用关系图:



该类的文档由以下文件生成:

- C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/[Packet.h](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/[Packet.cpp](#)

7.12 Chunk::RowCol结构体 参考

```
#include <Chunk.h>
```

Public 成员函数

- [RowCol](#) (qint32 [row](#)=0, qint32 [col](#)=0)

Public 属性

- qint32 [row](#)
- qint32 [col](#)

友元

- bool [operator<](#) (const [RowCol](#) &l, const [RowCol](#) &r)

7.12.1 构造及析构造函数说明

7.12.1.1 RowCol()

```
Chunk::RowCol::RowCol (
    qint32 row = 0,
    qint32 col = 0 ) [inline]
```

7.12.2 友元及相关函数文档

7.12.2.1 operator<

```
bool operator< (
    const RowCol & l,
    const RowCol & r ) [friend]
```

7.12.3 类成员变量说明

7.12.3.1 col

```
qint32 Chunk::RowCol::col
```


7.12.3.2 row

```
qint32 Chunk::RowCol::row
```

该结构体的文档由以下文件生成:

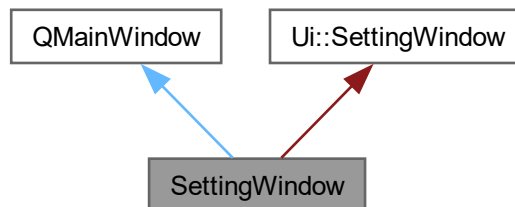
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h

7.13 SettingWindow类 参考

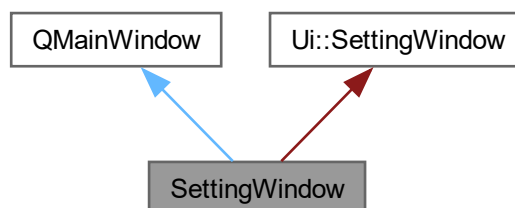
The [SettingWindow](#) class 设置界面

```
#include <SettingWindow.h>
```

类 SettingWindow 继承关系图:



SettingWindow 的协作图:



Public 成员函数

- [SettingWindow](#) (QWidget *parent=nullptr)
[SettingWindow::SettingWindow](#)
- virtual void [reloadApp](#) ()
[SettingWindow::reloadApp](#) 重启程序

7.13.1 详细描述

The [SettingWindow](#) class 设置界面

7.13.2 构造及析构函数说明

7.13.2.1 SettingWindow()

```
SettingWindow::SettingWindow (
    QWidget * parent = nullptr ) [explicit]
```

[SettingWindow::SettingWindow](#)

参数

parent	
--------	--

7.13.3 成员函数说明

7.13.3.1 reloadApp()

```
void SettingWindow::reloadApp ( ) [virtual]
```

[SettingWindow::reloadApp](#) 重启程序

该类的文档由以下文件生成:

- C:/Users/SJ/Desktop/扫雷/客户端源码/[SettingWindow.h](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/[SettingWindow.cpp](#)

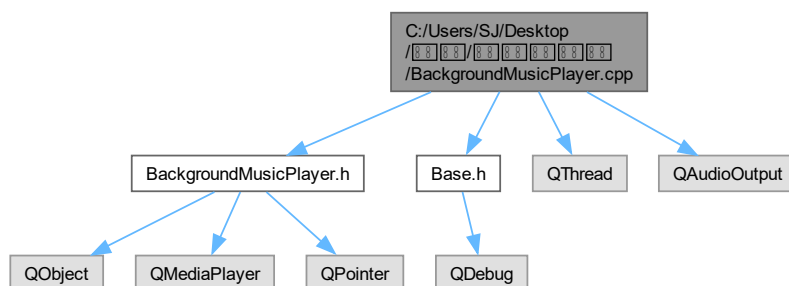
Chapter 8

文件说明

8.1 C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp 文件参考

```
#include "BackgroundMusicPlayer.h"  
#include "Base.h"  
#include <QThread>  
#include <QAudioOutput>
```

BackgroundMusicPlayer.cpp 的引用(Include)关系图:

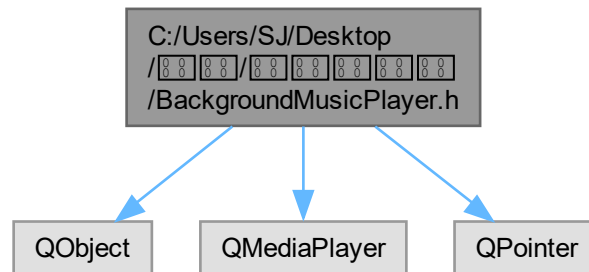


8.2 C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h 文件参考

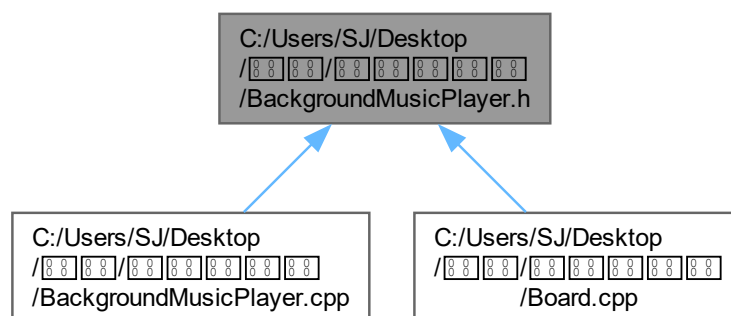
```
#include <QObject>  
#include <QMediaPlayer>
```

```
#include <QPointer>
```

BackgroundMusicPlayer.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `BackgroundMusicPlayer`

The `BackgroundMusicPlayer` class 这个类设置了退出自动删除，不必也不应该使用智能指针 该类是多线程类,将其moveToThread后通过信号使用它

8.3 BackgroundMusicPlayer.h

[浏览该文件的文档.](#)

```

00001 #pragma once
00002
00003 #include <QObject>
00004 #include<QMediaPlayer>
00005 #include<QPointer>
00011 class BackgroundMusicPlayer : public QObject
00012 {
  
```

```

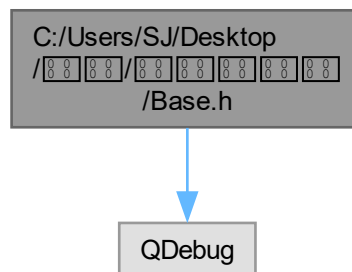
00013     Q_OBJECT
00014 public:
00015     explicit BackgroundMusicPlayer(QObject *parent = nullptr);
00016 private:
00017     QPointer<QMediaPlayer> player;
00018     QPointer<QAudioOutput> audioOutput;
00019 signals:
00020 public slots:
00021     virtual void playNewBackgroundMusc(QString path);
00022 };
00023
00024

```

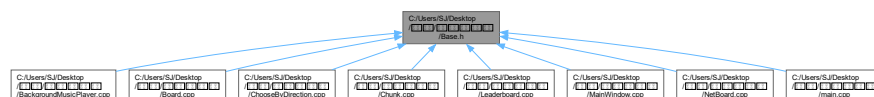
8.4 C:/Users/SJ/Desktop/扫雷/客户端源码/Base.h 文件参考

```
#include <QDebug>
```

Base.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



宏定义

- `#define dout qDebug()<<"["<<__LINE__<<","<<__FUNCTION__<<","<<__FILE__<<"]"`
摘要 调试宏
- `#define endl Qt::endl`

函数

- `template<typename T>`
`bool inRange (T l, T r, T p)`
检查是否在闭区间内部
- `template<typename T>`
`bool inRect (T topx, T topy, T width, T height, T px, T py)`
判断(px,py)是否在矩形(topx,topy,width,height)

8.4.1 宏定义说明

8.4.1.1 endl

```
#define endl Qt::endl
```

8.4.1.2 dout

```
#define dout qDebug() << "[" << __LINE__ << ", " << __FUNCTION__ << ", " << __FILE__ << "]" "
```

摘要 调试宏

@FileName [Base.h](#)

8.4.2 函数说明

8.4.2.1 inRange()

```
template<typename T >  
bool inRange (  
    T l,  
    T r,  
    T p ) [inline]
```

检查是否在闭区间内部

参数

l	左区间
r	右区间
p	点

返回

8.4.2.2 inRect()

```
template<typename T >
bool inRect (
    T topx,
    T topy,
    T width,
    T height,
    T px,
    T py ) [inline]
```

判断(px,py)是否在矩形(topx,topy,width,height)

参数

topx	矩形左上顶点横坐标
topy	矩形左上顶点纵坐标
width	矩形宽度
height	
px	
py	

返回

若在其内部返回true,否则返回false

8.5 Base.h

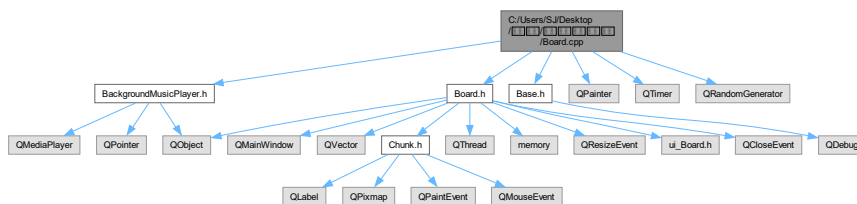
[浏览该文件的文档.](#)

```
00001 #pragma once
00002 #include<QDebug>
00008 #define dout qDebug() << "[" << __LINE__ << ", " << __FUNCTION__ << ", " << __FILE__ << "]" //< debug out (line,function
    name,file)
00009 #define endl Qt::endl //< debug endl
00018 template<typename T>
00019 inline bool inRange(T l,T r,T p)
00020 {
00021     return l <= p && p <= r;
00022 }
00033 template<typename T>
00034 inline bool inRect(T topx,T topy,T width,T height,T px,T py)
00035 {
00036     return inRange<T>(topx,width+topx,px) && inRange<T>(topy,height+topy,py);
00037 }
```

8.6 C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp 文件参考

```
#include "Board.h"
#include "Base.h"
#include <QPainter>
#include <QTimer>
#include <QRandomGenerator>
```

#include "BackgroundMusicPlayer.h"
Board.cpp 的引用(Include)关系图:



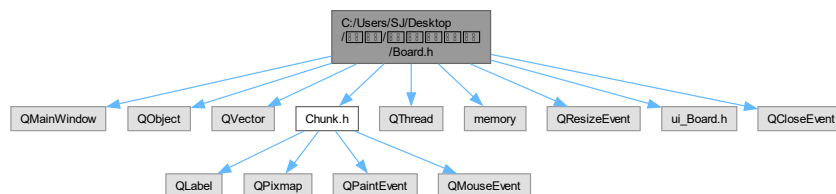
8.7 C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h 文件参考

```

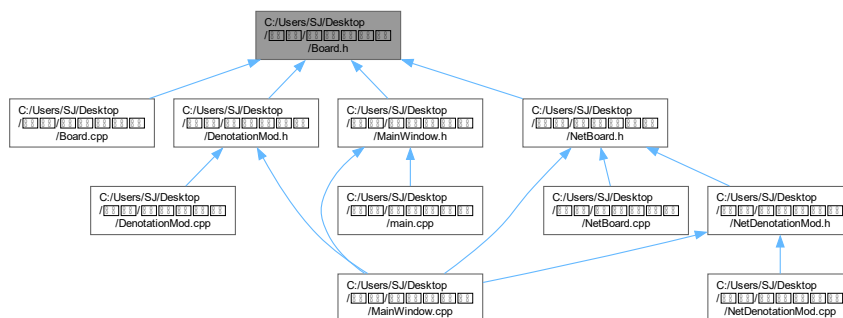
#include <QMainWindow>
#include <QObject>
#include <QVector>
#include "Chunk.h"
#include <QThread>
#include <memory>
#include <QResizeEvent>
#include "ui_Board.h"
#include <QCloseEvent>

```

Board.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class [Board](#)

The [Board](#) class 经典模式的游戏类

8.8 Board.h

[浏览该文件的文档.](#)

```
00001 #pragma once
00002
00003 #include <QMainWindow>
00004 #include <QObject>
00005 #include <QVector>
00006 #include "Chunk.h"
00007 #include <QThread>
00008 #include <memory>
00009 #include <QResizeEvent>
00010 #include "ui_Board.h"
00011 #include <QCloseEvent>
00012 class DenotationMod;
00013 class Board : public QMainWindow, protected Ui::Board
00014 {
00015     Q_OBJECT
00016     friend class DenotationMod;
00017 public:
00018     Board(qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent = nullptr, QString GameMod =
00019 "Classic");
00020     ~Board();
00021     qint32 getRowNum() const;
00022     qint32 getColNum() const;
00023     qint32 getBombNum() const;
00024     QPointer<QTimer> getGameTimer() const;
00025     void setIsFirstClick(bool newIsFirstClick);
00026 protected:
00027     const QString GameMod;
00028     QVector<QVector<QPointer<Chunk>>> chunks; //<
00029     const qint32 rowNum; //<
00030     const qint32 colNum; //<
00031     const qint32 bombNum; //<
00032     qint32 flagBombNum=0;
00033     qint32 minedNum;
00034     const static qint32 SurroundDirectionNum = 8; //<
00035     constexpr static qint32 SurroundDirection[SurroundDirectionNum][2] = {
00036         {1,0},{0,1},{-1,0},{0,-1},
00037         {1,1},{1,-1},{-1,1},{-1,-1}}; //<
00038     QPointer<QTimer> gameTimer;
00039     bool isFirstClick; //<
00040     qint32 selfCurrentIntegral=0;
00041 private:
00042     std::unique_ptr<QPainter> painter();
00043     // QPixmap* pix; //<
00044     std::unique_ptr<QPixmap> pix;
00045     QPointer<QThread> backgroundPlayerThread;
00046 private:
00047     void init();
00048 protected:
00049     void setBombs(Chunk::RowCol firstClickedRC);
00050     virtual void detect(Chunk::RowCol rc);
00051     void initSurroundBomb();
00052     inline bool inBoard(Chunk::RowCol rc); //<
00053     inline bool inBoard(qint32 row, qint32 col); //<
00054     virtual void gameOver(QString loseOrWin); //<
00055     virtual void uploadHistory();
00056     virtual qint32 calculateCurrentIntegral();
00057 private slots:
00058     virtual void paintEvent(QPaintEvent* e) override;
00059     virtual void resizeEvent(QResizeEvent* e) override;
00060     virtual void closeEvent(QCloseEvent* e) override;
00061 protected slots:
00062     // 自定义事件槽
00063     virtual void dealSignalExploded(); //<
00064     virtual void dealClickChunk(Chunk::RowCol rc); //<
```

```

00076     virtual void dealDoubleClickChunk(Chunk::RowCol rc,bool system=false);//<
00077 signals:
00078     void signalGameOver();
00079     void signalPlayNewBackGroundMusic(QString path);
00080     void signalMove();
00081     void signalUpLoadHistory(QString gameMod,QString rowNum,QString colNum,QString bombNum,QString
integral);
00082 };
00083

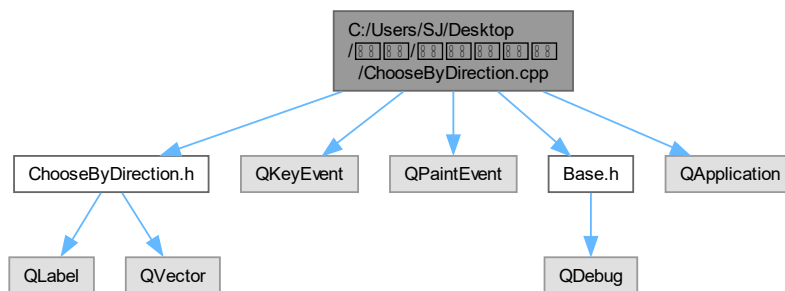
```

8.9 C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.cpp 文件参考

```

#include "ChooseByDirection.h"
#include <QKeyEvent>
#include <QPaintEvent>
#include <Base.h>
#include <QApplication>
ChooseByDirection.cpp 的引用(Include)关系图:

```

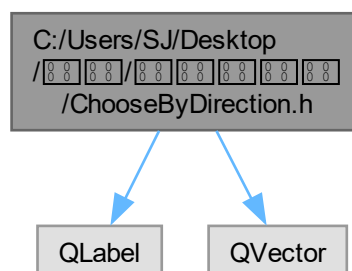


8.10 C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h 文件参考

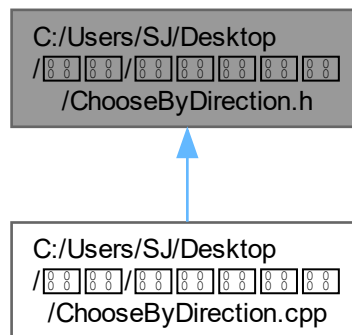
```

#include <QLabel>
#include <QVector>
ChooseByDirection.h 的引用(Include)关系图:

```



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `ChooseByDirection`

The `ChooseByDirection` class 自定义的根据方向键切换模式的控件,支持鼠标点击,会自动获取焦点 T 为可选选项的数据类型

8.11 ChooseByDirection.h

[浏览该文件的文档.](#)

```

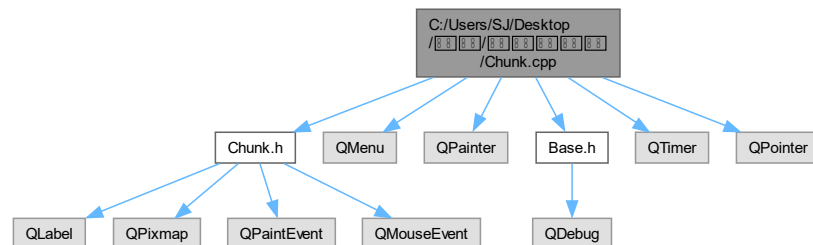
00001 #pragma once
00002
00003 #include <QLabel>
00004 #include<QVector>
00010 class ChooseByDirection : public QLabel
00011 {
00012     Q_OBJECT
00013 public:
00014     ChooseByDirection(QWidget* parent);
00015 public:
00016     virtual void setItems(const QVector<QString> &newItems);
00017     virtual void addItem(const QString& newItem);
00018     virtual const QString getCurrentItems()const;
00019 private:
00020     QVector<QString> items;//
00021     qint32 indexForItems=0;
00022 private slots:
00023     virtual void keyPressEvent(QKeyEvent* e)override;
00024     virtual void paintEvent(QPaintEvent*e)override;
00025     virtual void focusInEvent(QFocusEvent *e)override;
00026     virtual void focusOutEvent(QFocusEvent *e)override;
00027     virtual bool eventFilter(QObject *watched,QEvent *e)override;
00028     virtual void mousePressEvent(QMouseEvent* e)override;
00029     virtual void enterEvent(QEnterEvent *e)override;
00030     // virtual void leaveEvent(QEvent *e)override;//do nothing now
00031 };
00032
  
```

8.12 C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp 文件参考

```

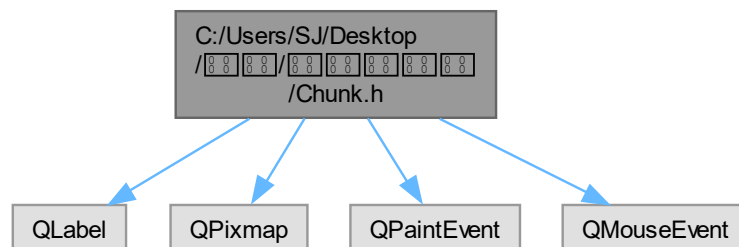
#include "Chunk.h"
#include <QMenu>
  
```

```
#include <QPainter>
#include "Base.h"
#include <QTimer>
#include <QPointer>
Chunk.cpp 的引用(Include)关系图:
```

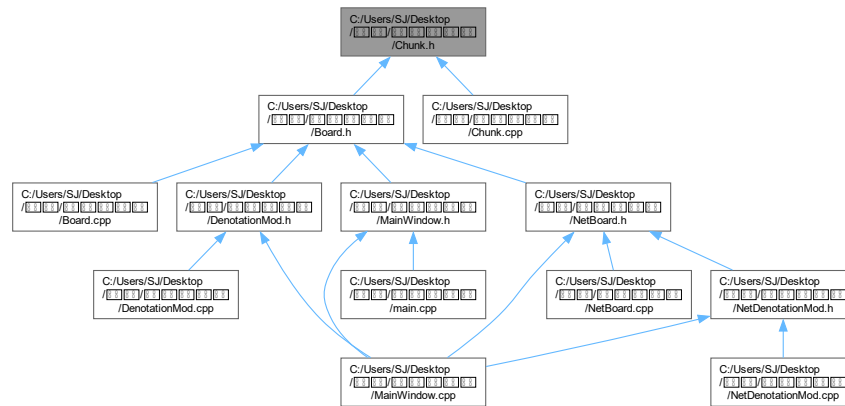


8.13 C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h 文件参考

```
#include <QLabel>
#include <QPixmap>
#include <QPaintEvent>
#include <QMouseEvent>
Chunk.h 的引用(Include)关系图:
```



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `Chunk`
The `Chunk` class 单个块对象
- struct `Chunk::RowCol`

8.14 Chunk.h

浏览该文件的文档.

```

00001 #pragma once
00002 #include <QLabel>
00003 #include<QPixmap>
00004 #include<QPaintEvent>
00005 #include<QMouseEvent>
00010 class Chunk : public QLabel
00011 {
00012     Q_OBJECT
00013 public:
00017     struct RowCol
00018     {
00019         qint32 row;
00020         qint32 col;
00021         RowCol(qint32 row = 0,qint32 col = 0):row(row),col(col) {};
00022         friend bool operator<(const RowCol& l,const RowCol& r)
00023         {
00024             if(l.row == r.row)
00025             {
00026                 return l.col < r.col;
00027             }
00028             return l.row < r.row;
00029         }
00030         // friend bool operator==(const RowCol& l,const RowCol& r)
00031         // {
00032         //     return (r.row == l.row) && (l.col == r.col);
00033         // }
00034     };
00035     enum class MineType{Bomb,NotBomb};//<
00036     enum class MineState{UnMined, Mined, FlagBomb, FlagQuestion};//<
00037     enum class RIGHT_KEY_MENU{FlagBomb,FlagQuestion};
00038 public:
00039     explicit Chunk(QWidget *parent = nullptr);
00040     // Chunk& operator=(const Chunk& h);
00041     // Chunk(const Chunk& h);
00042 public:
00043     virtual void setRowCol(qint32 row,qint32 col);
00044     const qint32 getChunkSize();
00045     virtual void setMineType(const MineType mt);
00046     virtual MineType getMineType()const;
  
```

```

00047     virtual qint32 getSurroundBomb() const;
00048     virtual void setSurroundBomb(qint32 newSurroundBomb);
00049     virtual const RowCol getRowCol() const;
00050     virtual MineState getMineState() const;
00051     virtual void setMineState(MineState newMineState);
00052     virtual void floatByDoubleClick();
00053
00054     // void setClickable(bool newClickable);
00055     virtual void showBomb();
00056     virtual void openThenShow();
00057     void setPix(const QPixmap &newPix);
00058
00059     const QPixmap &getPix() const;
00060
00061 private:
00062     virtual void drawSurroundBombNum(qint32 num);
00063     [[deprecated]] virtual void initRightKeyMenu();
00064     virtual void onTaskBoxContextMenuEvent();
00065 private:
00066     virtual void paintEvent(QPaintEvent* e) override;
00067     virtual void mousePressEvent(QMouseEvent* e) override;
00068     virtual void mouseDoubleClickEvent(QMouseEvent* e) override;
00069 private:
00070     const static qint32 DirectorNum = 4; //<
00071 private:
00072     QPixmap pix; //<全局画布,
00073     QSize pixSize{500,500};
00074     RowCol rowCol; //<
00075     MineType mineType = MineType::NotBomb; //<
00076     MineState mineState = MineState::UnMined; //<
00077     qint32 surroundBomb = 0; //<
00078     const qint32 director[DirectorNum][2] = {{1,0},{0,1},{-1,0},{0,-1}}; //<
00079     bool clickable = true;
00080 signals:
00081     void signalExploded(); //<
00082     void signalClickChunk(Chunk::RowCol rc); //<
00083     void signalDoubleClickChunk(Chunk::RowCol rc, bool system=false); //<
00085     void signalFlagBombChanged(qint32 changedNum);
00086 };

```

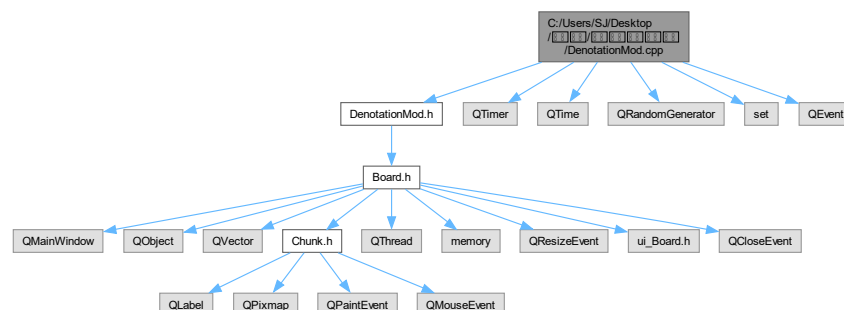
8.15 C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp 文件参考

```

#include "DenotationMod.h"
#include <QTimer>
#include <QTime>
#include <QRandomGenerator>
#include <set>
#include <QEvent>

```

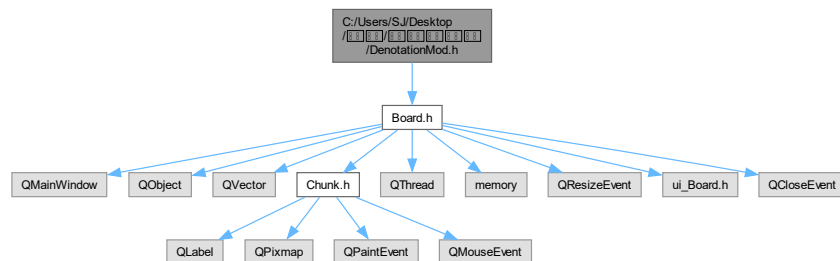
DenotationMod.cpp 的引用(Include)关系图:



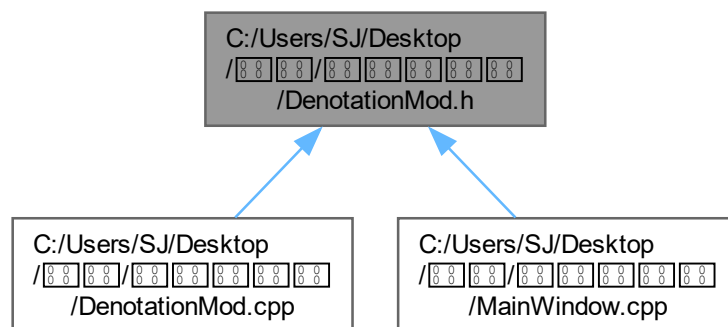
8.16 C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h 文件参考

```
#include "Board.h"
```

DenotationMod.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class [DenotationMod](#)

The [DenotationMod](#) class 爆炸模式，触雷不会死，但点击次数有限

8.17 DenotationMod.h

[浏览该文件的文档.](#)

```

00001 #pragma once
00002
00003 #include "Board.h"
00004 // #include "Chunk.h"
00009 class DenotationMod : public Board
00010 {
00011     Q_OBJECT
00012 public:
  
```

```

00013     DenotationMod(qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent = nullptr,QString GameMod
    = "Denotation");
00014 private:
00015     virtual void randomOpenNotBombChunk(qint32 num);
00016 private slots:
00017     virtual void dealSignalExploded() override;
00018 //     virtual void upLoadHistory() override;
00019 private:
00020     virtual bool eventFilter(QObject *watched,QEvent *e) override;
00021 private:
00022 //     const QString GameMod = "Denotation";
00023     qint32 moveNum;
00024     qint32 remainBombNum;
00025 };
00026

```

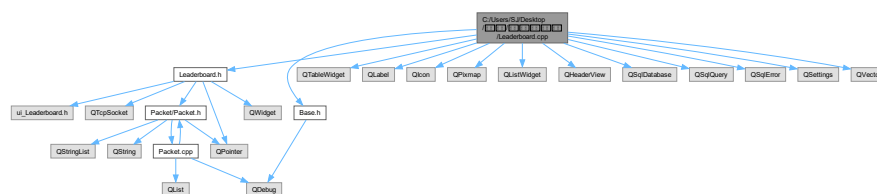
8.18 C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp 文件参考

```

#include "Leaderboard.h"
#include "Base.h"
#include <QTableWidget>
#include <QLabel>
#include <QIcon>
#include <QPixmap>
#include <QListWidget>
#include <QHeaderView>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QSettings>
#include <QVector>

```

Leaderboard.cpp 的引用(Include)关系图:



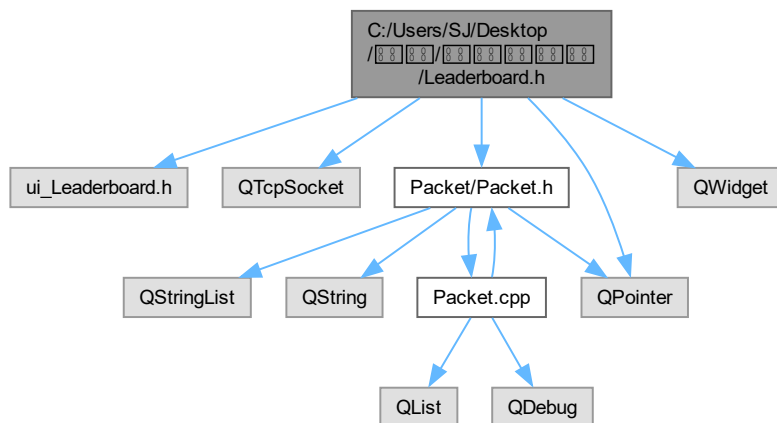
8.19 C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h 文件参考

```

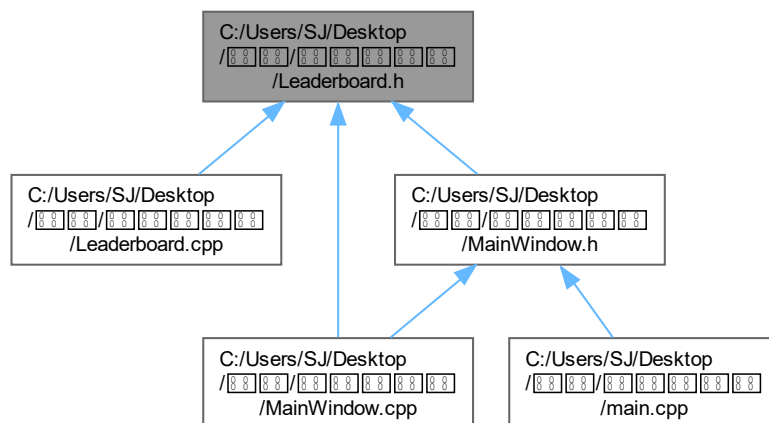
#include "ui_Leaderboard.h"
#include <QTcpSocket>
#include "Packet/Packet.h"
#include <QWidget>
#include <QPointer>

```


Leaderboard.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `Leaderboard`

The `Leaderboard` class 显示历史战绩

8.20 Leaderboard.h

[浏览该文件的文档.](#)

```
00001 #pragma once
00002
```


8.21.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

函数调用图:



8.21.1.2 myMessageOutput()

```
void myMessageOutput (
    QtMsgType type,
    const QMessageLogContext & context,
    const QString & msg )
```

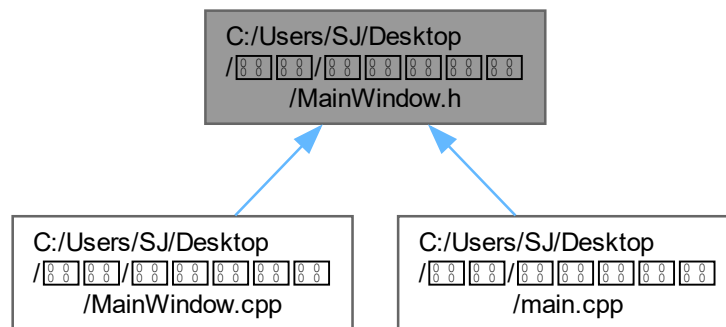
这是这个函数的调用关系图:



8.22 C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp 文件参考

```
#include "MainWindow.h"
#include <QFileInfo>
#include <QMessageBox>
#include <QUrl>
#include "Base.h"
#include "SettingWindow.h"
#include "DenotationMod.h"
#include <QFormLayout>
#include <QDialogButtonBox>
#include <QLineEdit>
#include <QPushButton>
```


此图展示该文件直接或间接的被哪些文件引用了:



类

- class `MainWindow`

The `MainWindow` class 主窗口对象

8.24 MainWindow.h

浏览该文件的文档.

```

00001 #pragma once
00002
00003 #include "ui_MainWindow.h"
00004 #include <QMainWindow>
00005 #include "Board.h"
00006 #include <QSettings>
00007 #include <QTcpSocket>
00008 #include "Packet/Packet.h"
00009 #include "Leaderboard.h"
00010 #include <QByteArray>
00011 #include <QSqlDatabase>
00016 class MainWindow : public QMainWindow, private Ui::MainWindow
00017 {
00018     Q_OBJECT
00019
00020 public:
00021     MainWindow(QWidget *parent = nullptr);
00022     ~MainWindow();
00023 private:
00024     void init();
00025     qint32 getSettingsIntValue(QString sectionName, QString ValueName);
00026     virtual bool tryLogin(QString email, QString password);
00027     virtual void showSignUpAndLogin();
00028     virtual void showCaptcha();
00029     virtual void downloadRemoteHistoryFile();
00030 private slots:
00031     void on_pushButtonBeginGame_clicked();
00032     void on_pushButtonSetting_clicked();
00033     void on_pushButtonDenotation_clicked();
00034     void on_pushButtonLeaderboard_clicked();
00035 private slots:
00036     virtual void dealConnected();
00037     virtual void dealDisconnected();
00038     virtual void dealRecv();
00039     virtual void dealGameOver();
00040 private:
00041     // const QString IP = "101.42.8.164"; //服务器ip
00042     const QString IP = "127.0.0.1"; //连接本地测试
00043     Board* board;
  
```

```

00044     QPointer<Leaderboard> leaderBoard;
00045     QTcpSocket* socket;
00046     QString difficulty;
00047     bool isLogin = false;
00048     QSqlDatabase database;
00049 private:
00050     friend class Packet<MainWindow>;
00051     Packet<MainWindow> packet;
00052     virtual void dealLoginResponse(QStringList list);
00053     virtual void dealSignUpResponse(QStringList list);
00054     virtual void dealCaptchaResponse(QStringList list);
00055     virtual void dealSignalUploadHistory(QString gameMod, QString rowNum, QString colNum, QString
bombNum, QString integral);
00056     virtual void dealTansferHistoryFileHead(QStringList list);
00057     virtual void dealTansferringHistoryFile(QStringList list);
00058     virtual void dealTansferHistoryFileEnd(QStringList list);
00059 signals:
00060     void signalMainSocketNewRecvMessage(QByteArray mes);
00061 };

```

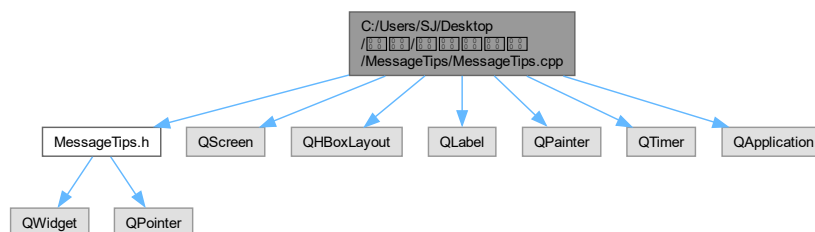
8.25 C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp 文件参考

```

#include "MessageTips.h"
#include <QScreen>
#include <QHBoxLayout>
#include <QLabel>
#include <QPainter>
#include <QTimer>
#include <QApplication>

```

MessageTips.cpp 的引用(Include)关系图:



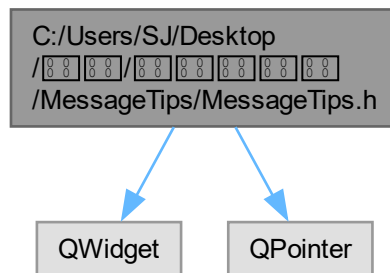
8.26 C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h 文件参考

```

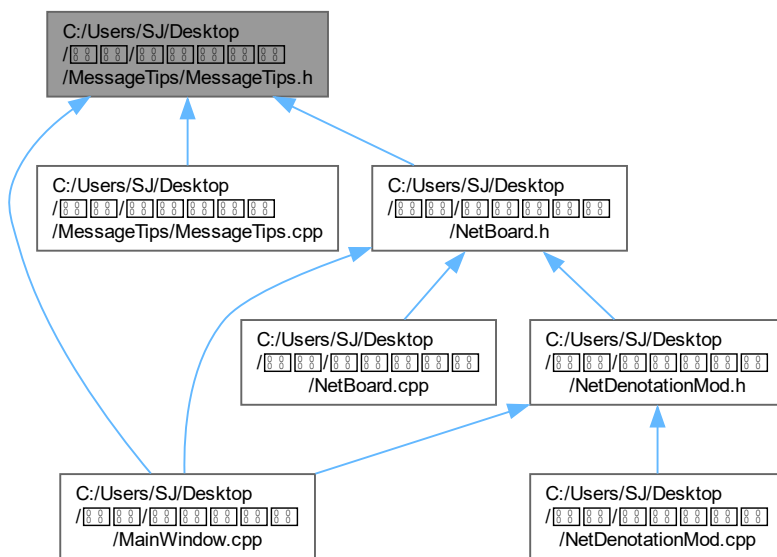
#include <QWidget>
#include <QPointer>

```

MessageTips.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `MessageTips`

The `MessageTips` class 实现自动消失的消息框,由于时间原因,此代码借鉴于csdn 必须使用指针类型或者指定父对象 @my doing 添加了对qt6的兼容 添加关闭时自动删除, 释放资源,避免内存泄露

8.27 MessageTips.h

[浏览该文件的文档.](#)

```

00001 #ifndef MESSAGETIPS_H
00002 #define MESSAGETIPS_H
00003
00004 #include <QWidget>
00005 #include<QPointer>
00006 // #pragma execution_character_set(push) // push the previous character
00007 # pragma execution_character_set("utf-8")
00008
00009 class QHBoxLayout;
00010 class QLabel;
00022 class MessageTips : public QWidget
00023 {
00024     Q_OBJECT
00025 public:
00026     explicit MessageTips(QString showStr="none", QWidget *parent = nullptr);
00027     ~MessageTips();
00028     double getOpacityValue() const;
00029     void setOpacityValue(double value);
00030
00031     qint32 getTextSize() const;
00032     void setTextSize(int value);
00033
00034     QColor getTextColor() const;
00035     void setTextColor(const QColor &value);
00036
00037     QColor getBackgroundColor() const;
00038     void setBackgroundColor(const QColor &value);
00039
00040     QColor getFrameColor() const;
00041     void setFrameColor(const QColor &value);
00042
00043     qint32 getFrameSize() const;
00044     void setFrameSize(int value);
00045
00046     qint32 getShowTime() const;
00047     void setShowTime(int msec);
00048
00049     void setCloseTimeSpeed(int closeTime = 100, double closeSpeed = 0.1);
00050 public:
00051     // virtual void show() override;
00052 protected:
00053     void paintEvent(QPaintEvent *event) override;
00054 private:
00055     void InitLayout(); // 初始化窗体的布局和部件
00056     QPointer<HBoxLayout> hBoxLayout; // 布局显示控件布局
00057     QPointer<QLabel> mText; // 用于显示文字的控件
00058     QString showStr; // 显示的字符串
00059
00060     double opacityValue; // 窗体初始化透明度
00061     QFont* font;
00062     qint32 textSize; // 显示字体大小
00063     QColor textColor; // 字体颜色
00064     QColor backgroundColor; // 窗体的背景色
00065     QColor frameColor; // 边框颜色
00066     qint32 frameSize; // 边框粗细大小
00067
00068     qint32 showTime; // 显示时间
00069     qint32 closeTime; // 关闭需要时间
00070     double closeSpeed; // 窗体消失的平滑度, 大小0~1
00071
00072 signals:
00073 };
00074
00075 // #pragma execution_character_set(pop) // pop the previous character set
00076 #endif // MESSAGETIPS_H

```

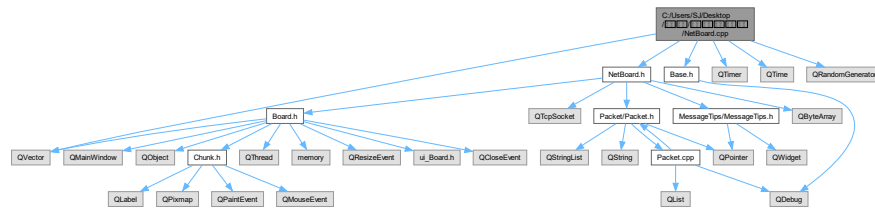
8.28 C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp 文件参考

```

#include "NetBoard.h"
#include "Base.h"
#include <QTimer>
#include <QTime>
#include <QRandomGenerator>
#include <QVector>

```

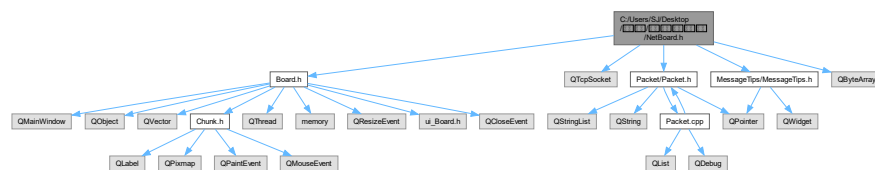

NetBoard.cpp 的引用(Include)关系图:



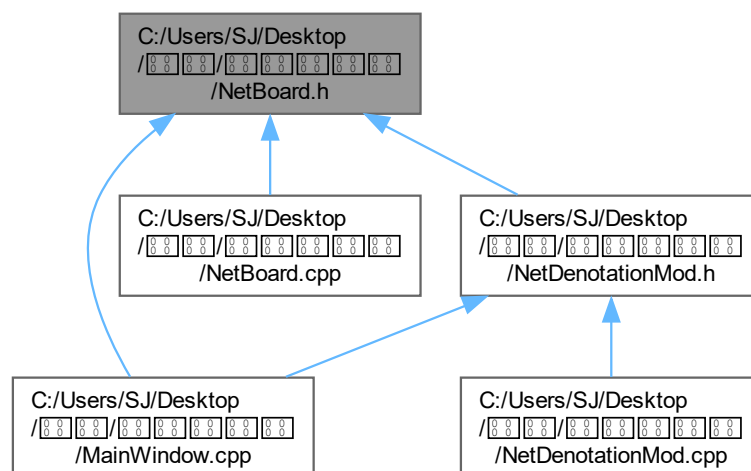
8.29 C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h 文件参考

```
#include "Board.h"
#include <QTcpSocket>
#include "Packet/Packet.h"
#include "MessageTips/MessageTips.h"
#include <QByteArray>
```

NetBoard.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class [NetBoard](#)

The [NetBoard](#) class 经典模式的网络对战

8.30 NetBoard.h

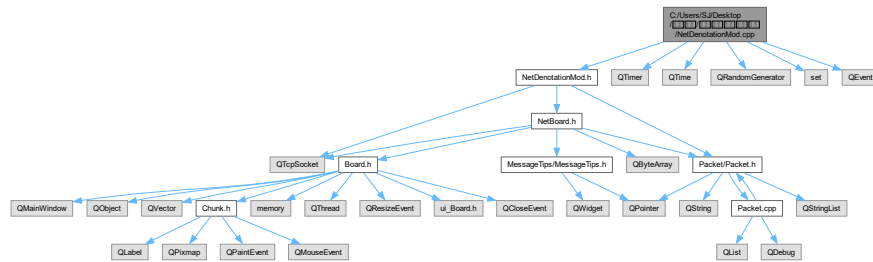
[浏览该文件的文档.](#)

```
00001 #pragma once
00002
00003 #include "Board.h"
00004 #include <QTcpSocket>
00005 #include "Packet/Package.h"
00006 #include "MessageTips/MessageTips.h"
00007 #include <QByteArray>
00012 class NetBoard : public Board
00013 {
00014     Q_OBJECT
00015 public:
00016     NetBoard(QTcpSocket* socket, qint32 rowNum, qint32 colNum, qint32 bombNum, QWidget *parent =
        nullptr, QString GameMod = "NetBoard");
00017 private:
00018     void initPacket();
00019
00020     virtual void initNetBoardThenSendBySocket();
00021     QString generateGameStateStr(qint32 selfIntegral, qint32 antiIntegral, QString antiEmail);
00022 protected:
00023     virtual void gameOver(QString loseOrWin) override; // <
00024     virtual void sendIntegralToServer(qint32 integral);
00025     virtual void queryNewMatch();
00026 private:
00027     friend class Packet<NetBoard>;
00028     QTcpSocket* socket;
00029     Packet<NetBoard> packet;
00030     QString antiPlayerEmail;
00031     qint32 antiCurrentIntegral=0;
00032 protected slots:
00033     virtual void dealMatchResponse(QStringList list);
00034     virtual void dealClickChunk(Chunk::RowCol rc) override; // <
00035 public slots:
00036     virtual void dealMainSocketNewRecvMessage(QByteArray mes);
00037     virtual void dealNetInitState(QStringList list);
00038     virtual void dealUpdateAntiIntegral(QStringList list);
00039     virtual void dealAntiGameOver(QStringList list);
00040 private:
00041     MessageTips* matchingMessageTips;
00042 };
00043
```

8.31 C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.cpp 文件参考

```
#include "NetDenotationMod.h"
#include <QTimer>
#include <QTime>
#include <QRandomGenerator>
#include <set>
#include <QEvent>
```

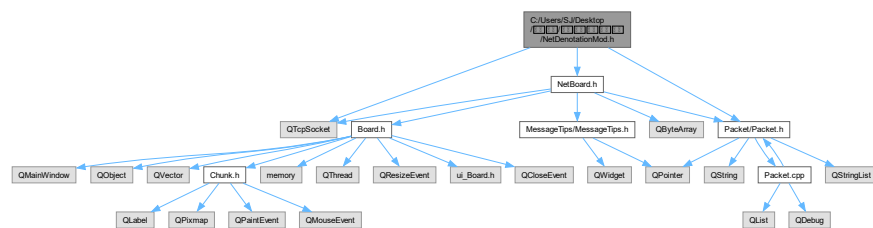
NetDenotationMod.cpp 的引用(Include)关系图:



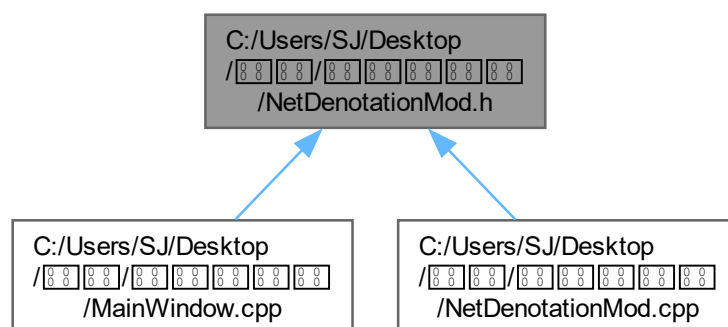
8.32 C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h 文件参考

```
#include "NetBoard.h"
#include <QTcpSocket>
#include "Packet/Packet.h"
```

NetDenotationMod.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class [NetDenotationMod](#)

The [NetDenotationMod](#) class 爆炸模式网络对战

8.33 NetDenotationMod.h

[浏览该文件的文档.](#)

```

00001 #pragma once
00002 #include "NetBoard.h"
00003 #include<QTcpSocket>
00004 #include"Packet/Packet.h"
00009 class NetDenotationMod : public NetBoard
00010 {
00011     Q_OBJECT
00012 public:
00013     NetDenotationMod(QTcpSocket* socket,qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent =
        nullptr,QString GameMod = "NetDenotation");
00014 protected:
00015 private:
00016     void randomOpenNotBombChunk(qint32 num);
00017 protected slots:
00018     virtual void dealSignalExploded()override;
00019     virtual void dealMatchResponse(QStringList list)override;
00020 private:
00021     virtual bool eventFilter(QObject *watched,QEvent *e)override;
00022 //     virtual void detect(Chunk::RowCol rc)override;
00023 private:
00024     qint32 moveNum;
00025     qint32 remainBombNum;
00026     QTcpSocket* socket;
00027     friend class Packet<NetDenotationMod>;
00028     Packet<NetDenotationMod> packet;
00029 };
00030

```

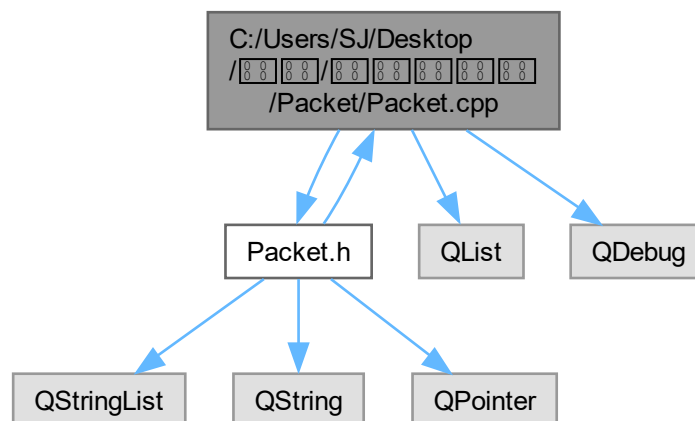
8.34 C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.cpp 文件参考

```

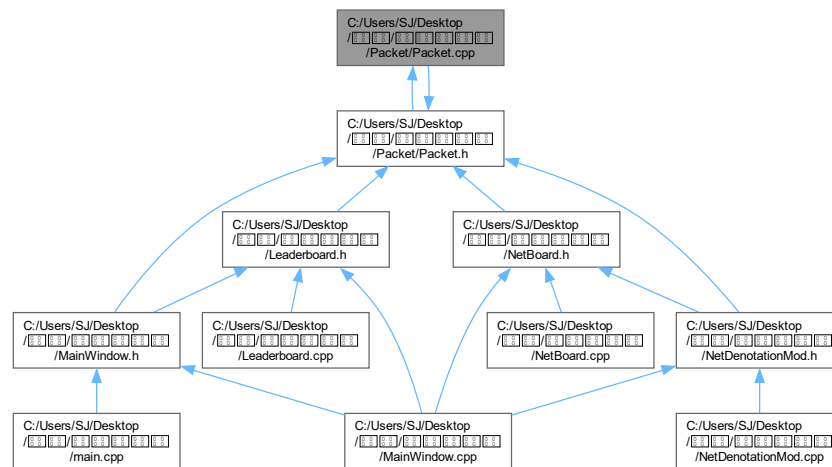
#include "Packet.h"
#include <QList>
#include <QDebug>

```

Packet.cpp 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



宏定义

- `#define dout qDebug()<<"["<<__LINE__<<","<<__FUNCTION__<<","<<__FILE__<<"]"`
- `#define endl Qt::endl`

8.34.1 宏定义说明

8.34.1.1 endl

```
#define endl Qt::endl
```

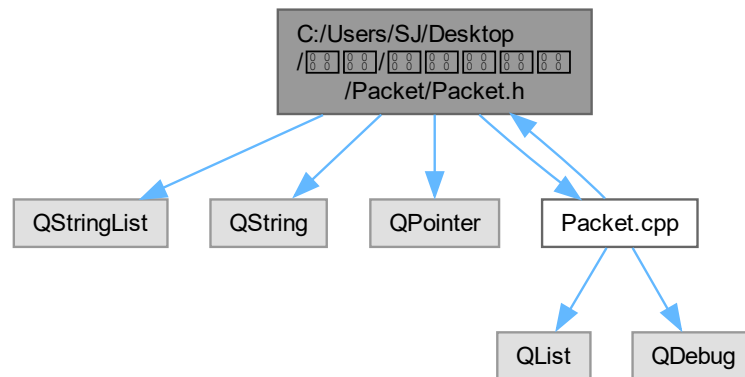
8.34.1.2 dout

```
#define dout qDebug()<<"["<<__LINE__<<","<<__FUNCTION__<<","<<__FILE__<<"]"
```

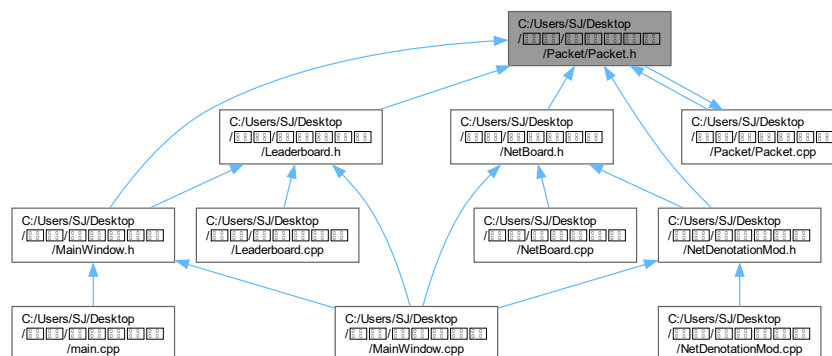
8.35 C:/Users/SJ/Desktop/扫雷/客户端源码/Package/Package.h 文件参考

```
#include <QStringList>
#include <QString>
#include <QPointer>
#include "Packet.cpp"
```

Packet.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class `Packet< T >`

用于socket协议的信息封装和解包, 可以绑定信息-回调函数, Packet.cpp和Packet.h都得放在头文件中(-I Packet.cpp Packet.h) 如果要绑定私有行为, 应该将Packet<T>声明为友元 T为parent对应的类名, installClassFunctionEvent 会在触发时调用parent的成员函数 所有要绑定的函数都应该以void为返回值, QStringList为参数

8.36 Packet.h

[浏览该文件的文档。](#)

```
00001 // Packet.h -- 模板类的声明
00002 #pragma once
00003 #include <QStringList>
00004 #include <QString>
00005 #include <QPointer>
00012 template<typename T>
00013 class Packet
00014 {
00015     struct FunctionEvent
00016     {
00017     public:
00018         QString funcName;
00019         qint32 parameterNum;
00020         void (T::*callBack) (QStringList);
00021         FunctionEvent (QString funcName, qint32 parameterNum, void (T::*callBack) (QStringList))
00022             :funcName (funcName), parameterNum (parameterNum), callBack (callBack) {};
00023     };
00024 public:
00025     Packet (T* parent);
00026     virtual void pushMessage (QString newMes); // 压入信息, 可能会触发callBack
00027     virtual QString formatMes (QStringList newMesList); // 将消息封装
00028     virtual QString formatMes (QString newMes); // 重载
00029     virtual void installClassFunctionEvent (QString funcName, qint32 parameterNum, void
(T::*callBack) (QStringList));
00030 private:
00031     virtual void distributerEvent ();
00032     virtual inline QStringList splitMes (QString mes);
00033 private:
00034     static const QString separator;
00035     QStringList recvList;
00036     QString recvBuff;
00037     T* parent;
00038     // warning: 这里使用普通指针我也不知道会出什么问题不
00039     QList<FunctionEvent*> funcEvents;
00040 };
00041
00042 #include "Packet.cpp" // 包含模板类的实现
```

8.37 C:/Users/SJ/Desktop/扫雷/客户端源码/resource_rc.py 文件参考

命名空间

- namespace [resource_rc](#)

函数

- def [resource_rc.qInitResources \(\)](#)
- def [resource_rc.qCleanupResources \(\)](#)

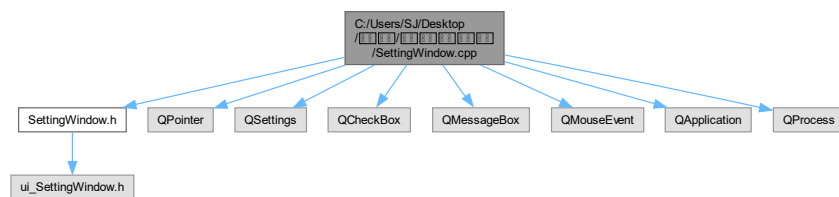
变量

- b [resource_rc.qt_resource_data](#)
- b [resource_rc.qt_resource_name](#)
- b [resource_rc.qt_resource_struct_v1](#)
- b [resource_rc.qt_resource_struct_v2](#)
- list [resource_rc.qt_version](#) = [int(v) for v in QtCore.qVersion().split('.')]
- int [resource_rc.rcc_version](#) = 1
- b [resource_rc.qt_resource_struct](#) = qt_resource_struct_v1
- [resource_rc.else](#) :

8.38 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.cpp 文件参考

```
#include "SettingWindow.h"  
#include <QPointer>  
#include <QSettings>  
#include <QCheckBox>  
#include <QMessageBox>  
#include <QMouseEvent>  
#include <QApplication>  
#include <QProcess>
```

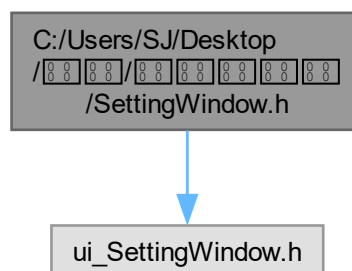
SettingWindow.cpp 的引用(Include)关系图:



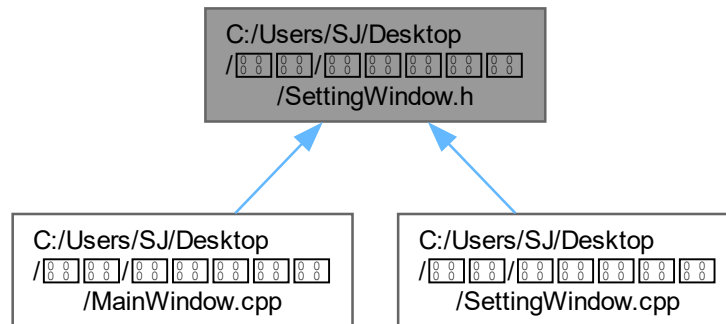
8.39 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h 文件参考

```
#include "ui_SettingWindow.h"
```

SettingWindow.h 的引用(Include)关系图:



此图展示该文件直接或间接的被哪些文件引用了:



类

- class [SettingWindow](#)
The [SettingWindow](#) class 设置界面

8.40 SettingWindow.h

浏览该文件的文档.

```
00001 #pragma once
00002
00003 #include "ui_SettingWindow.h"
00008 class SettingWindow : public QMainWindow, private Ui::SettingWindow
00009 {
00010     Q_OBJECT
00011
00012 public:
00013     explicit SettingWindow(QWidget *parent = nullptr);
00014     virtual void reloadApp();
00015 private slots:
00016     void on_pushButtonCancel_clicked();
00017     void on_pushButtonOk_clicked();
00018     virtual void mousePressEvent(QMouseEvent* e) override;
00019     void on_pushButtonExitLogin_clicked();
00020 };
00021
```


Index

- ~Board
 - Board, [20](#)
- ~MainWindow
 - MainWindow, [51](#)
- ~MessageTips
 - MessageTips, [54](#)
- addItems
 - ChooseByDirection, [34](#)
- BackgroundMusicPlayer, [15](#)
 - BackgroundMusicPlayer, [16](#)
 - playNewBackgroundMusc, [16](#)
- Base.h
 - dendl, [84](#)
 - dout, [84](#)
 - inRange, [84](#)
 - inRect, [84](#)
- Board, [17](#)
 - ~Board, [20](#)
 - Board, [19](#)
 - bombNum, [30](#)
 - calculateCurrentIntegral, [20](#)
 - chunks, [31](#)
 - colNum, [31](#)
 - dealClickChunk, [20](#)
 - dealDoubleClickChunk, [21](#)
 - dealSignalExploded, [22](#)
 - DenotationMod, [30](#)
 - detect, [22](#)
 - flagBombNum, [31](#)
 - GameMod, [31](#)
 - gameOver, [23](#)
 - gameTimer, [31](#)
 - getBombNum, [24](#)
 - getColNum, [24](#)
 - getGameTimer, [25](#)
 - getRowNum, [25](#)
 - inBoard, [25](#), [26](#)
 - initSurroundBomb, [27](#)
 - isFirstClick, [31](#)
 - minedNum, [31](#)
 - rowNum, [31](#)
 - selfCurrentIntegral, [32](#)
 - setBombs, [27](#)
 - setIsFirstClick, [28](#)
 - signalGameOver, [28](#)
 - signalMove, [29](#)
 - signalPlayNewBackGroundMusic, [29](#)
 - signalUpLoadHistory, [29](#)
 - SurroundDirection, [32](#)
 - SurroundDirectionNum, [32](#)
 - upLoadHistory, [30](#)
- Bomb
 - Chunk, [37](#)
- bombNum
 - Board, [30](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp, [81](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h, [81](#), [82](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Base.h, [83](#), [85](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp, [85](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h, [86](#), [87](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.cpp, [88](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h, [88](#), [89](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp, [89](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h, [90](#), [91](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp, [92](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h, [93](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp, [94](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h, [94](#), [95](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/main.cpp, [96](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp, [97](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.h, [98](#), [99](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp, [100](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h, [100](#), [101](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp, [102](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h, [103](#), [104](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.cpp, [104](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h, [105](#), [106](#)
- C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.cpp, [106](#)

C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.h,
 108, 109
 C:/Users/SJ/Desktop/扫雷/客户端源码/resource_rc.py,
 109
 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.cpp
 110
 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h,
 110, 111
 calculateCurrentIntegral
 Board, 20
 ChooseByDirection, 32
 addItem, 34
 ChooseByDirection, 33
 getCurrentItems, 34
 setItems, 34
 Chunk, 35
 Bomb, 37
 Chunk, 37
 FlagBomb, 37
 FlagQuestion, 37
 floatByDoubleClick, 38
 getChunkSize, 38
 getMineState, 38
 getMineType, 38
 getPix, 39
 getRowCol, 39
 getSurroundBomb, 39
 Mined, 37
 MineState, 36
 MineType, 37
 NotBomb, 37
 openThenShow, 40
 RIGHT_KEY_MENU, 37
 setMineState, 40
 setMineType, 41
 setPix, 41
 setRowCol, 41
 setSurroundBomb, 42
 showBomb, 42
 signalClickChunk, 42
 signalDoubleClickChunk, 43
 signalExploded, 43
 signalFlagBombChanged, 43
 UnMined, 37
 Chunk::RowCol, 77
 col, 78
 operator<, 78
 row, 78
 RowCol, 78
 chunks
 Board, 31
 col
 Chunk::RowCol, 78
 colNum
 Board, 31
 dealAntiGameOver
 NetBoard, 63
 dealClickChunk
 Board, 20
 NetBoard, 63
 dealDoubleClickChunk
 Board, 21
 dealMainSocketNewRecvMessage
 Leaderboard, 48
 NetBoard, 64
 dealMatchResponse
 NetBoard, 65
 NetDenotationMod, 74
 dealNetInitState
 NetBoard, 65
 dealSignalExploded
 Board, 22
 NetDenotationMod, 74
 dealUpdateAntiIntegral
 NetBoard, 66
 endl
 Base.h, 84
 Packet.cpp, 107
 DenotationMod, 44
 Board, 30
 DenotationMod, 46
 detect
 Board, 22
 dout
 Base.h, 84
 Packet.cpp, 107
 else
 resource_rc, 12
 FlagBomb
 Chunk, 37
 flagBombNum
 Board, 31
 FlagQuestion
 Chunk, 37
 floatByDoubleClick
 Chunk, 38
 formatMes
 Packet< T >, 76
 GameMod
 Board, 31
 gameOver
 Board, 23
 NetBoard, 66
 gameTimer
 Board, 31
 getBackgroundColor
 MessageTips, 54
 getBombNum
 Board, 24
 getChunkSize
 Chunk, 38
 getColNum
 Board, 24
 getCurrentItems

- ChooseByDirection, 34
- getFrameColor
 - MessageTips, 54
- getFrameSize
 - MessageTips, 54
- getGameTimer
 - Board, 25
- getMineState
 - Chunk, 38
- getMineType
 - Chunk, 38
- getOpacityValue
 - MessageTips, 55
- getPix
 - Chunk, 39
- getRowCol
 - Chunk, 39
- getRowNum
 - Board, 25
- getShowTime
 - MessageTips, 55
- getSurroundBomb
 - Chunk, 39
- getTextColor
 - MessageTips, 55
- getTextSize
 - MessageTips, 55
- inBoard
 - Board, 25, 26
- initSurroundBomb
 - Board, 27
- inRange
 - Base.h, 84
- inRect
 - Base.h, 84
- installClassFunctionEvent
 - Packet< T >, 76
- isFirstClick
 - Board, 31
- Leaderboard, 47
 - dealMainSocketNewRecvMessage, 48
 - Leaderboard, 48
 - Packet< Leaderboard >, 49
- main
 - main.cpp, 96
- main.cpp
 - main, 96
 - myMessageOutput, 97
- MainWindow, 49
 - ~MainWindow, 51
 - MainWindow, 50
 - Packet< MainWindow >, 51
 - signalMainSocketNewRecvMessage, 51
- MessageTips, 52
 - ~MessageTips, 54
 - getBackgroundColor, 54
 - getFrameColor, 54
 - getFrameSize, 54
 - getOpacityValue, 55
 - getShowTime, 55
 - getTextColor, 55
 - getTextSize, 55
 - MessageTips, 53
 - paintEvent, 56
 - setBackgroundColor, 56
 - setCloseTimeSpeed, 56
 - setFrameColor, 57
 - setFrameSize, 57
 - setOpacityValue, 57
 - setShowTime, 58
 - setTextColor, 58
 - setTextSize, 58
- Mined
 - Chunk, 37
- minedNum
 - Board, 31
- MineState
 - Chunk, 36
- MineType
 - Chunk, 37
- myMessageOutput
 - main.cpp, 97
- NetBoard, 59
 - dealAntiGameOver, 63
 - dealClickChunk, 63
 - dealMainSocketNewRecvMessage, 64
 - dealMatchResponse, 65
 - dealNetInitState, 65
 - dealUpdateAntiIntegral, 66
 - gameOver, 66
 - NetBoard, 62
 - Packet< NetBoard >, 69
 - queryNewMatch, 67
 - sendIntegralToServer, 68
- NetDenotationMod, 69
 - dealMatchResponse, 74
 - dealSignalExploded, 74
 - NetDenotationMod, 73
 - Packet< NetDenotationMod >, 75
- NotBomb
 - Chunk, 37
- openThenShow
 - Chunk, 40
- operator<
 - Chunk::RowCol, 78
- Packet
 - Packet< T >, 76
- Packet< Leaderboard >
 - Leaderboard, 49
- Packet< MainWindow >
 - MainWindow, 51
- Packet< NetBoard >

- NetBoard, 69
- Packet< NetDenotationMod >
 - NetDenotationMod, 75
- Packet< T >, 75
 - formatMes, 76
 - installClassFunctionEvent, 76
 - Packet, 76
 - pushMessage, 77
- Packet.cpp
 - dendl, 107
 - dout, 107
- paintEvent
 - MessageTips, 56
- playNewBackgroundMusc
 - BackgroundMusicPlayer, 16
- pushMessage
 - Packet< T >, 77
- qCleanupResources
 - resource_rc, 11
- qInitResources
 - resource_rc, 11
- qt_resource_data
 - resource_rc, 12
- qt_resource_name
 - resource_rc, 12
- qt_resource_struct
 - resource_rc, 12
- qt_resource_struct_v1
 - resource_rc, 13
- qt_resource_struct_v2
 - resource_rc, 13
- qt_version
 - resource_rc, 13
- queryNewMatch
 - NetBoard, 67
- rcc_version
 - resource_rc, 13
- reloadApp
 - SettingWindow, 80
- resource_rc, 11
 - else, 12
 - qCleanupResources, 11
 - qInitResources, 11
 - qt_resource_data, 12
 - qt_resource_name, 12
 - qt_resource_struct, 12
 - qt_resource_struct_v1, 13
 - qt_resource_struct_v2, 13
 - qt_version, 13
 - rcc_version, 13
- RIGHT_KEY_MENU
 - Chunk, 37
- row
 - Chunk::RowCol, 78
- RowCol
 - Chunk::RowCol, 78
- rowNum
 - Board, 31
- selfCurrentIntegral
 - Board, 32
- sendIntegralToServer
 - NetBoard, 68
- setBackgroundColor
 - MessageTips, 56
- setBombs
 - Board, 27
- setCloseTimeSpeed
 - MessageTips, 56
- setFrameColor
 - MessageTips, 57
- setFrameSize
 - MessageTips, 57
- setIsFirstClick
 - Board, 28
- setItems
 - ChooseByDirection, 34
- setMineState
 - Chunk, 40
- setMineType
 - Chunk, 41
- setOpacityValue
 - MessageTips, 57
- setPix
 - Chunk, 41
- setRowCol
 - Chunk, 41
- setShowTime
 - MessageTips, 58
- setSurroundBomb
 - Chunk, 42
- setTextColor
 - MessageTips, 58
- setTextSize
 - MessageTips, 58
- SettingWindow, 79
 - reloadApp, 80
 - SettingWindow, 80
- showBomb
 - Chunk, 42
- signalClickChunk
 - Chunk, 42
- signalDoubleClickChunk
 - Chunk, 43
- signalExploded
 - Chunk, 43
- signalFlagBombChanged
 - Chunk, 43
- signalGameOver
 - Board, 28
- signalMainSocketNewRecvMessage
 - MainWindow, 51
- signalMove
 - Board, 29
- signalPlayNewBackGroundMusic
 - Board, 29

signalUploadHistory
 Board, [29](#)
SurroundDirection
 Board, [32](#)
SurroundDirectionNum
 Board, [32](#)

UnMined
 Chunk, [37](#)
uploadHistory
 Board, [30](#)