扫雷

1.0.3

制作者 Doxygen 1.9.6

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:

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:

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:

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:

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

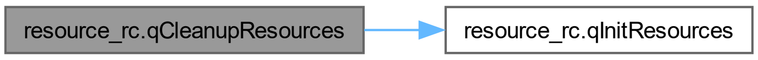
- b qt_resource_data
- b qt_resource_name
- b qt_resource_struct_v1
- b qt_resource_struct_v2
- list qt_version = [int(v) for v in QtCore.qVersion().split('.')]
- int rcc_version = 1
- b qt_resource_struct = qt_resource_struct_v1
- else :

6.1.1 函数说明

6.1.1.1 qCleanupResources()

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

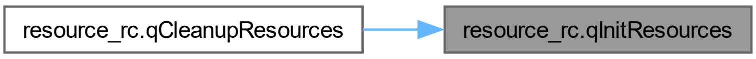
函数调用图:



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:

```
=  b"\
\x00\x03\
\x00\x00\x70\x37\
\x00\x69\
\x00\x6d\x00\x67\
\x00\x08\
\x0c\xa6\xc3\x95\
\x00\x52\
\x00\x65\x00\x73\x00\x6f\x00\x75\x00\x72\x00\x63\x00\x65\
\x00\x0d\
\x0a\xb7\x65\x47\
\x00\x73\
\x00\x61\x00\x66\x00\x65\x00\x43\x00\x68\x00\x75\x00\x6e\x00\x6b\x00\x2e\x00\x70\x00\x6e\x00\x67\
\x00\x0f\
\x0e\x3d\xd6\xa7\
\x00\x75\
      \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\
\x00\x10\
\x02\x89\x10\x27\
\x00\x75\
      \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
\x00\x10\
\x04\x9b\x1b\xa7\
\x00\x65\
      \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
"
```

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:
```
=  b"\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x01\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x02\
\x00\x00\x00\x0c\x00\x02\x00\x00\x00\x01\x00\x00\x00\x03\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x04\x00\x00\x00\x04\
\x00\x00\x00\x66\x00\x00\x00\x00\x00\x01\x00\x00\x1b\xb3\
\x00\x00\x00\x8c\x00\x00\x00\x00\x00\x01\x00\x00\x4a\xa6\
\x00\x00\x00\x22\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\
\x00\x00\x00\x42\x00\x00\x00\x00\x00\x01\x00\x00\x09\xf1\
"
```

6.1.2.6 qt_resource_struct_v2

b resource_rc.qt_resource_struct_v2

初始值:
```
=  b"\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x01\
\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x01\x00\x00\x00\x02\
\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x0c\x00\x02\x00\x00\x00\x01\x00\x00\x00\x03\
\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x00\x00\x02\x00\x00\x00\x04\x00\x00\x00\x04\
\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x66\x00\x00\x00\x00\x00\x01\x00\x00\x1b\xb3\
\x00\x00\x01\x87\x4a\x54\xe6\xaf\
\x00\x00\x00\x8c\x00\x00\x00\x00\x00\x01\x00\x00\x4a\xa6\
\x00\x00\x01\x87\x7e\x8a\xcd\x1e\
\x00\x00\x00\x22\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\
\x00\x00\x01\x87\x7e\x89\x10\x87\
\x00\x00\x00\x42\x00\x00\x00\x00\x00\x01\x00\x00\x09\xf1\
\x00\x00\x01\x87\x7e\x8a\x24\x97\
"
```

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

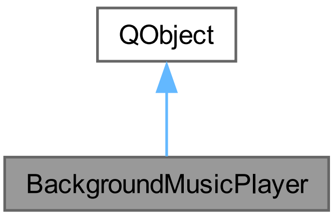
类说明

7.1 BackgroundMusicPlayer类 参考

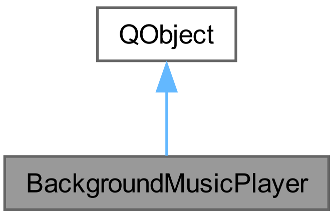
The BackgroundMusicPlayer class 这个类设置了退出自动删除，不必也不应该使用智能指针 该类是多线程类,将其moveToThread后通过信号使用它

```
#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:

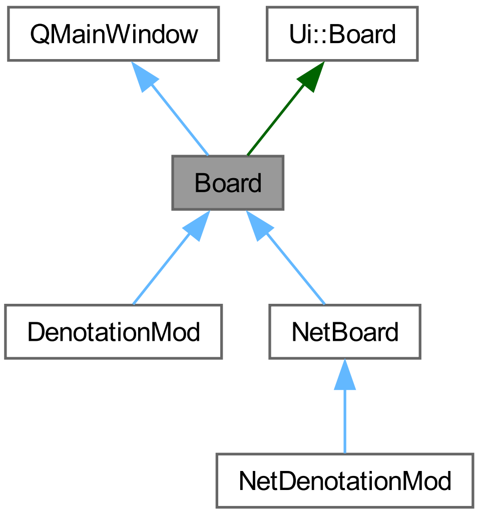
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.cpp

7.2 Board类 参考

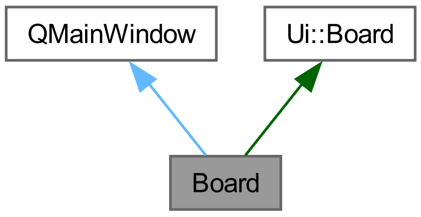
The Board class 经典模式的游戏类

`#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

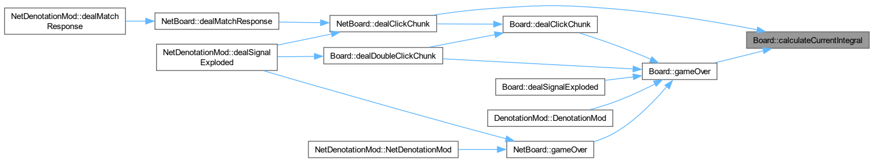
7.2.3.1 calculateCurrentIntegral()

```
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:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
NetDenotationMod::NetDenotationMod
Board::dealClickChunk
Board::dealSignalExploded
DenotationMod::DenotationMod
NetBoard::gameOver
Board::gameOver
Board::calculateCurrentIntegral

7.2.3.2 dealClickChunk

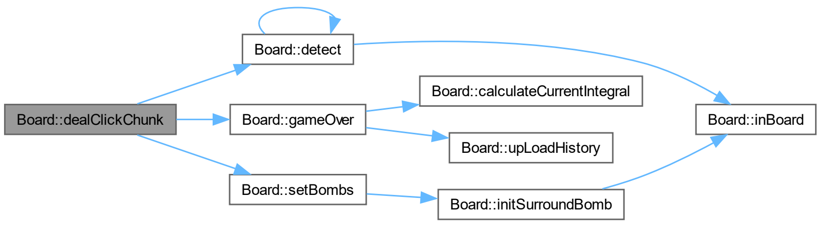
```
void Board::dealClickChunk (
            Chunk::RowCol rc )  [protected], [virtual], [slot]
```

Board::dealClickChunk 单击事件槽函数

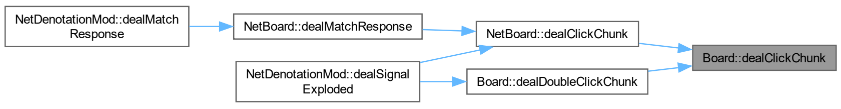
参数

rc	

函数调用图:



这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk

7.2.3.3 dealDoubleClickChunk

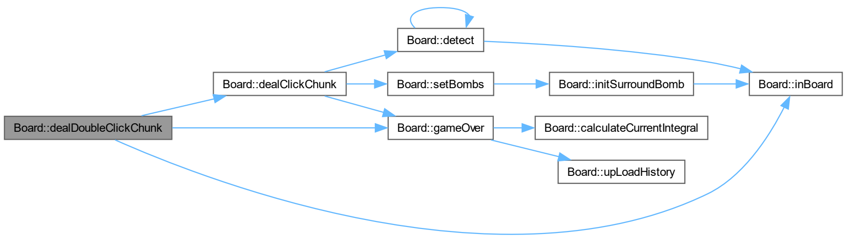
```
void Board::dealDoubleClickChunk (
             Chunk::RowCol rc,
             bool system = false )  [protected], [virtual], [slot]
```

Board::dealDoubleClickChunk 双击事件槽函数

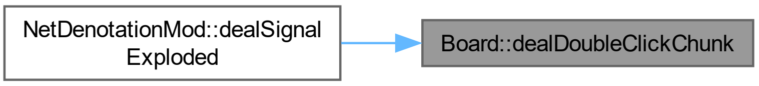
参数

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:

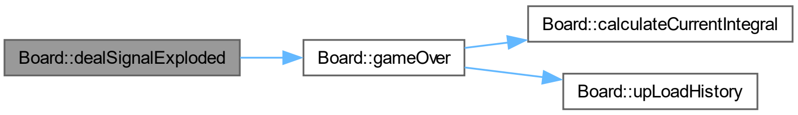

NetDenotationMod::dealSignal Exploded
Board::dealDoubleClickChunk

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()

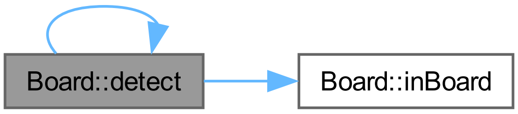
```
void Board::detect (
            Chunk::RowCol rc )  [protected], [virtual]
```

Board::detect 递归探索周围的雷区

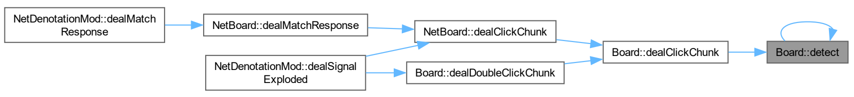
参数

rc	

函数调用图:



这是这个函数的调用关系图:


NetDenotationMod::dealMatchResponse
NetBoard::dealMatchResponse
NetDenotationMod::dealSignalExploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk
Board::detect

7.2.3.6 gameOver()

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

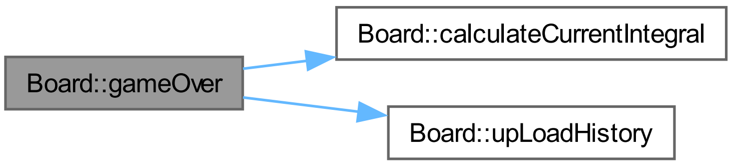
Board::gameOver 游戏结束槽函数

参数

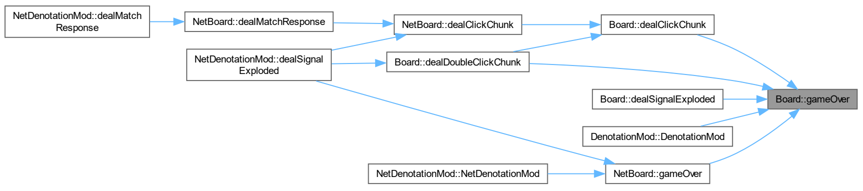
loseOrWin	输赢?

被 NetBoard 重载.

函数调用图:



这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
Board::dealClickChunk
NetDenotationMod::dealSignal Exploded
Board::dealDoubleClickChunk
Board::dealSignalExploded
Board::gameOver
DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

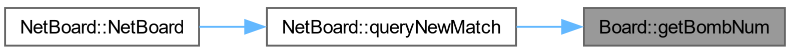
7.2.3.7 getBombNum()

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

Board::getBombNum

返回

这是这个函数的调用关系图:



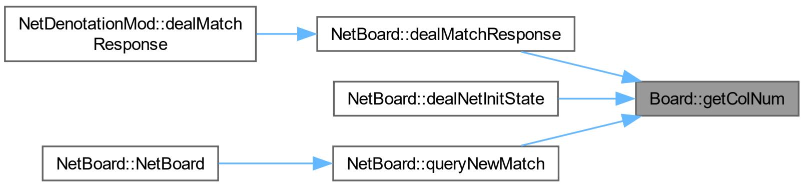
7.2.3.8 getColNum()

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

Board::getColNum

返回

这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealNetInitState
Board::getColNum
NetBoard::NetBoard
NetBoard::queryNewMatch

7.2.3.9 getGameTimer()

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

Board::getGameTimer 获得游戏计时器对象

返回

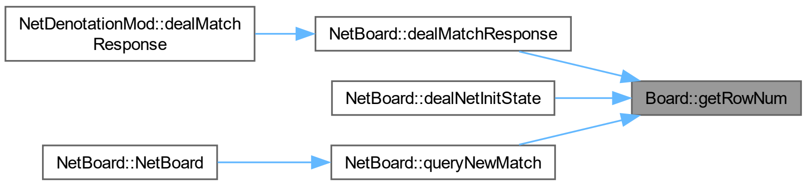
7.2.3.10 getRowNum()

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

Board::getRowNum

返回

这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealNetInitState
Board::getRowNum
NetBoard::NetBoard
NetBoard::queryNewMatch

7.2.3.11 inBoard() [1/2]

```
bool Board::inBoard (
            Chunk::RowCol rc )  [inline], [protected]
```

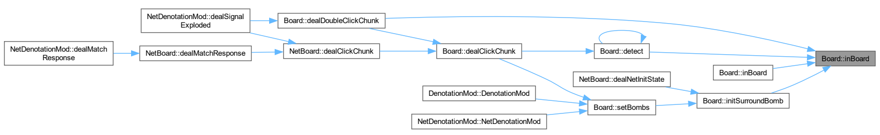
Board::inBoard 判断是否在棋盘内部

参数

rc	

返回

这是这个函数的调用关系图:


NetDenotationMod::dealSignal Exploded
Board::dealDoubleClickChunk
NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
Board::dealClickChunk
Board::detect
Board::inBoard
Board::inBoard
NetBoard::dealNetInitState
DenotationMod::DenotationMod
Board::initSurroundBomb
Board::setBombs
NetDenotationMod::NetDenotationMod

7.2.3.12 inBoard() [2/2]

```
bool Board::inBoard (
            qint32 row,
            qint32 col )  [inline], [protected]
```

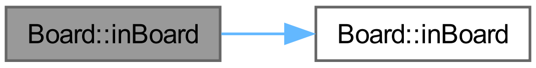
Board::inBoard 重载

参数

row	
col	

返回

函数调用图:

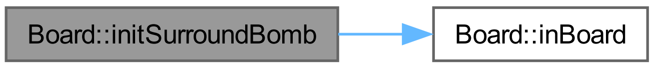


7.2.3.13 initSurroundBomb()

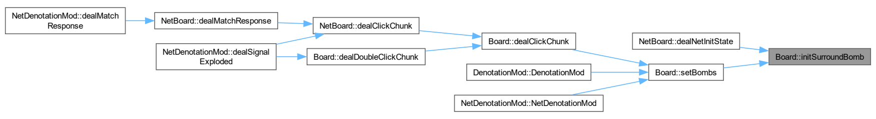
```
void Board::initSurroundBomb ( )  [protected]
```

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:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk
DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
NetBoard::dealNetInitState
Board::setBombs
Board::initSurroundBomb

7.2.3.14 setBombs()

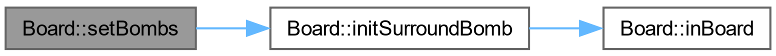
```
void Board::setBombs (
            Chunk::RowCol firstClickedRC )  [protected]
```

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

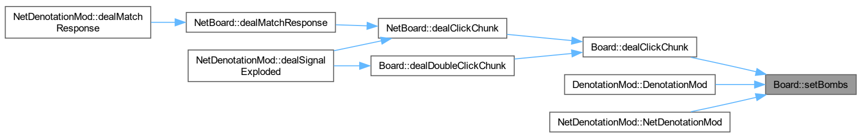
参数

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:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk
DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
Board::setBombs

7.2.3.15 setIsFirstClick()

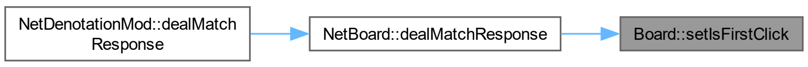
```
void Board::setIsFirstClick (
             bool newIsFirstClick )
```

Board::setIsFirstClick

参数

newIsFirstClick	是否是第一次点击

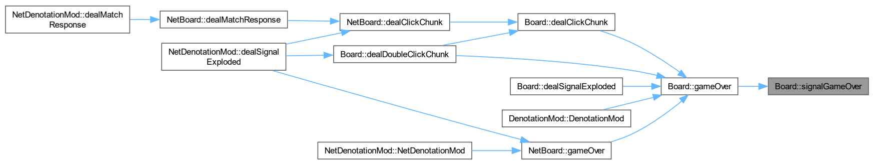
这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
Board::setIsFirstClick

7.2.3.16 signalGameOver

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

这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk
Board::dealSignalExploded
DenotationMod::DenotationMod
Board::gameOver
Board::signalGameOver
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

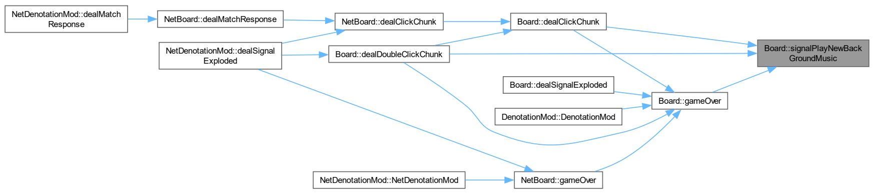
7.2.3.17 signalMove

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

7.2.3.18 signalPlayNewBackGroundMusic

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

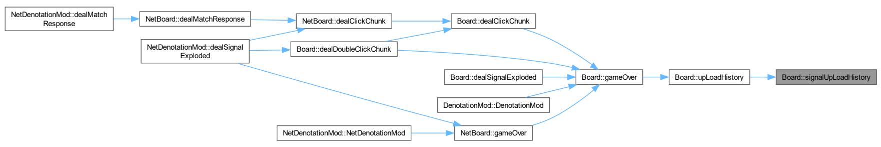
这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetBoard::dealClickChunk
Board::dealDoubleClickChunk
Board::dealClickChunk
Board::signalPlayNewBack GroundMusic
Board::dealSignalExploded
DenotationMod::DenotationMod
Board::gameOver
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

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:

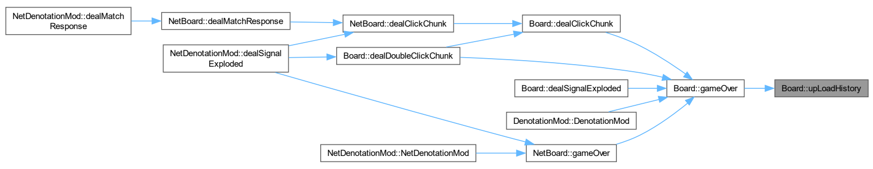

NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
Board::dealClickChunk
NetDenotationMod::dealSignal Exploded
Board::dealDoubleClickChunk
Board::dealSignalExploded
Board::gameOver
Board::upLoadHistory
Board::signalUpLoadHistory
DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

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:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
Board::dealClickChunk
NetDenotationMod::dealSignal Exploded
Board::dealDoubleClickChunk
Board::dealSignalExploded
Board::gameOver
Board::upLoadHistory
DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

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:

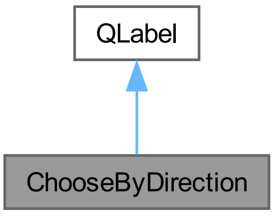
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Board.cpp

7.3 ChooseByDirection类 参考

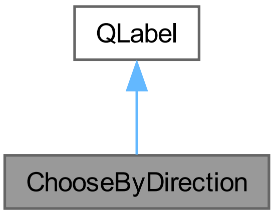
The ChooseByDirection class 自定义的根据方向键切换模式的控件,支持鼠标点击，会自动获取焦点 T 为可选选项的数据类型

```
#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 addItems()

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

ChooseByDirection::addItems 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:

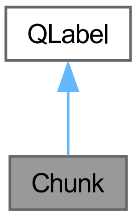
- C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.cpp

7.4 Chunk类 参考

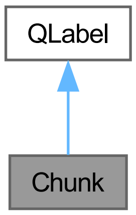
The Chunk class 单个块对象

```
#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 ( )
```

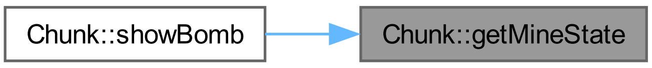
7.4.4.3 getMineState()

```
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:



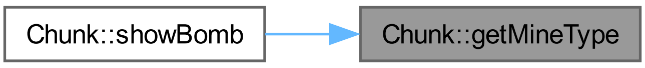
7.4.4.4 getMineType()

```
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

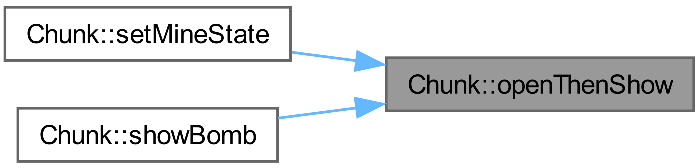
返回

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()

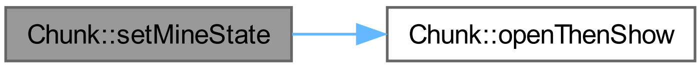
```
void Chunk::setMineState (
            MineState newMineState )  [virtual]
```

Chunk::setMineState

参数

newMineState	

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 (
             qint32 row,
             qint32 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

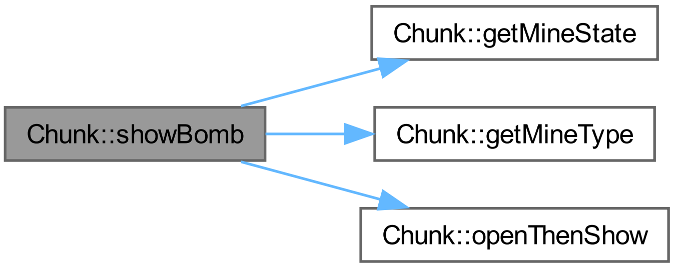
newSurroundBomb	

7.4.4.14 showBomb()

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

Chunk::showBomb 翻开所有雷

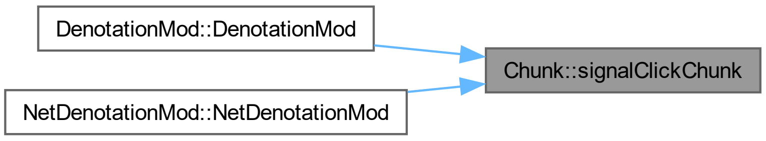
函数调用图:



7.4.4.15 signalClickChunk

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

这是这个函数的调用关系图:


DenotationMod::DenotationMod
NetDenotationMod::NetDenotationMod
Chunk::signalClickChunk

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 (
             qint32 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:

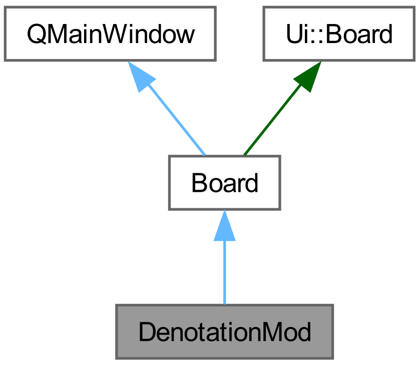
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.cpp

7.5 DenotationMod类 参考

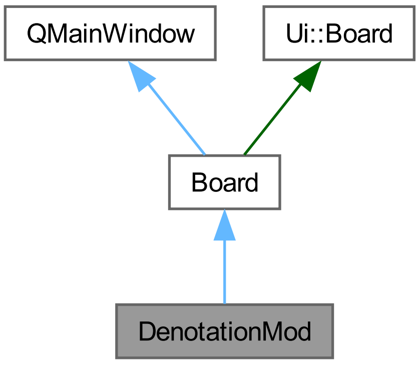
The DenotationMod class 爆炸模式，触雷不会死，但点击次数有限

```
#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()

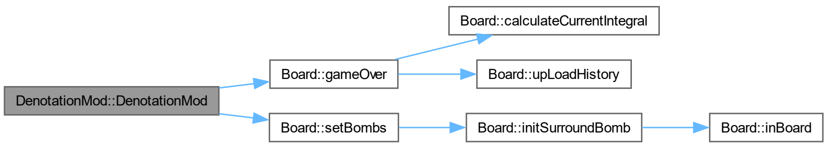
```
DenotationMod::DenotationMod (
            qint32 rowNum,
            qint32 colNum,
            qint32 bombNum,
            QWidget * parent = nullptr,
            QString GameMod = "Denotation" )
```

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:

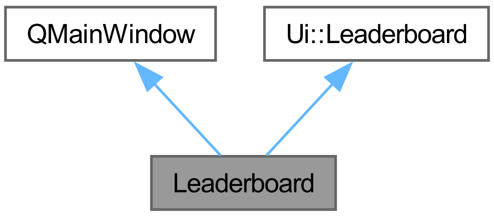
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.cpp

7.6 Leaderboard类 参考

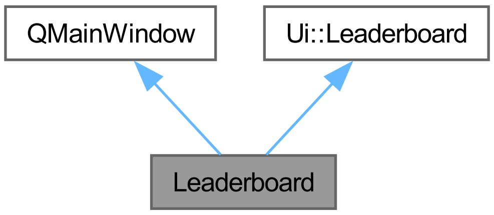
The Leaderboard class 显示历史战绩

```
#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

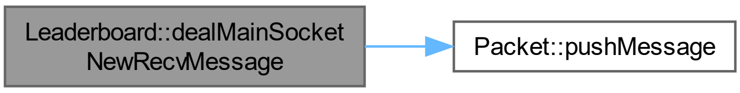
```
void Leaderboard::dealMainSocketNewRecvMessage (
            QByteArray mes )  [virtual], [slot]
```

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:

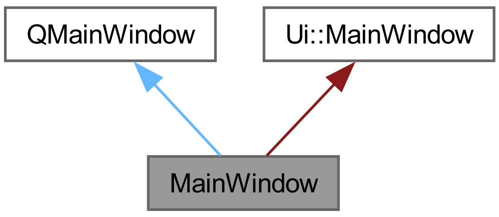
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.cpp

7.7 MainWindow类 参考

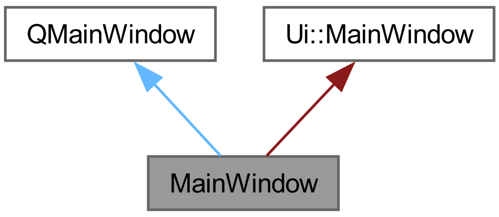
The MainWindow class 主窗口对象

`#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()

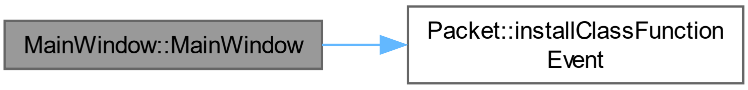
```
MainWindow::MainWindow (
            QWidget ∗ parent = nullptr )
```

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:

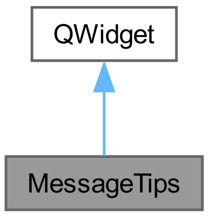
- C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp

7.8 MessageTips类 参考

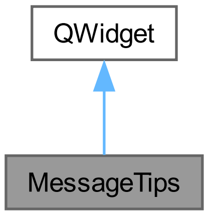
The MessageTips class 实现自动消失的消息框,由于时间原因,此代码借鉴于csdn 必须使用指针类型或者指定父对象 @my doing 添加了对qt6的兼容 添加关闭时自动删除，释放资源,避免内存泄露

```
#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()

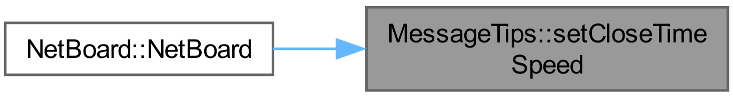
```
void MessageTips::setCloseTimeSpeed (
            int closeTime = 100,
            double closeSpeed = 0.1 )
```

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()

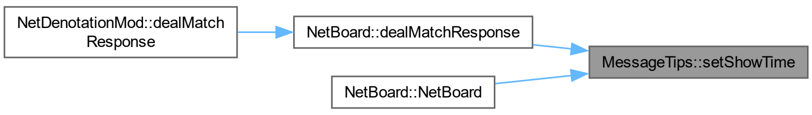
```
void MessageTips::setShowTime (
             int value )
```

MessageTips::setShowTime

参数

value	

这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::NetBoard
MessageTips::setShowTime

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:

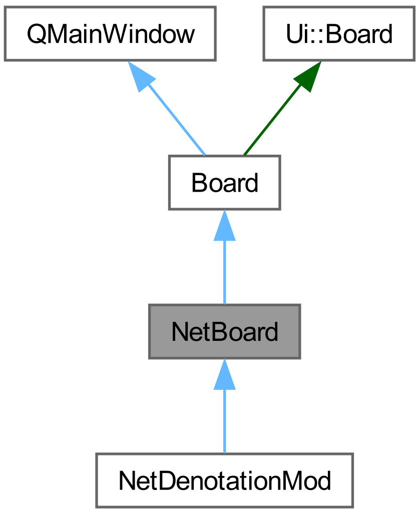
- C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.cpp

7.9 NetBoard类 参考

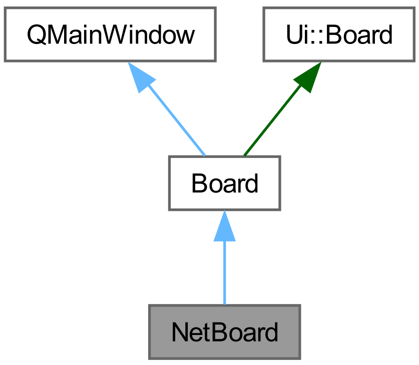
The NetBoard class 经典模式的网络对战

```
#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()

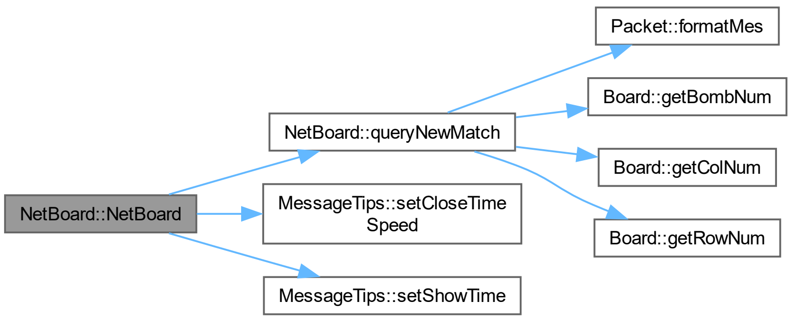
```
NetBoard::NetBoard (
            QTcpSocket * socket,
            qint32 rowNum,
            qint32 colNum,
            qint32 bombNum,
            QWidget * parent = nullptr,
            QString GameMod = "NetBoard" )
```

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<<new↩ Mes; 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

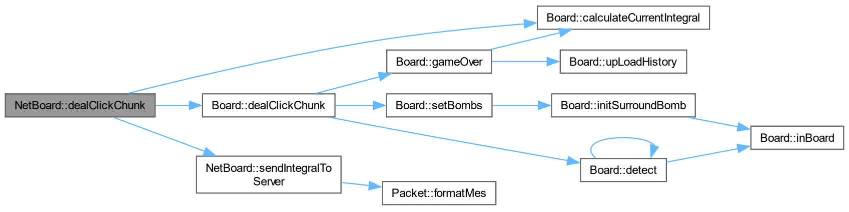
参数

rc	

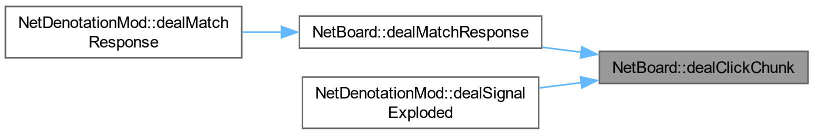
注解

: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:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
NetDenotationMod::dealSignal Exploded

7.9.3.3 dealMainSocketNewRecvMessage

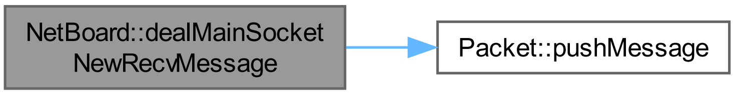
```
void NetBoard::dealMainSocketNewRecvMessage (
            QByteArray mes )  [virtual], [slot]
```

NetBoard::dealMainSocketNewRecvMessage 接受mainWindow转发的消息

参数

mes	

函数调用图:



7.9.3.4 dealMatchResponse

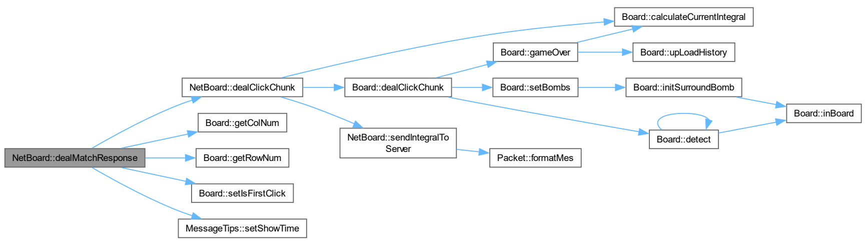
```
void NetBoard::dealMatchResponse (
            QStringList list )  [protected], [virtual], [slot]
```

NetBoard::dealMatchResponse 处理匹配结果

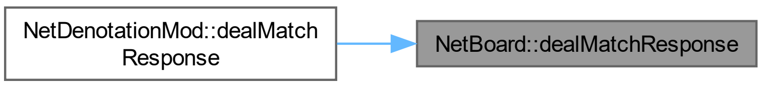
参数

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

函数调用图:



这是这个函数的调用关系图:


NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse

7.9.3.5 dealNetInitState

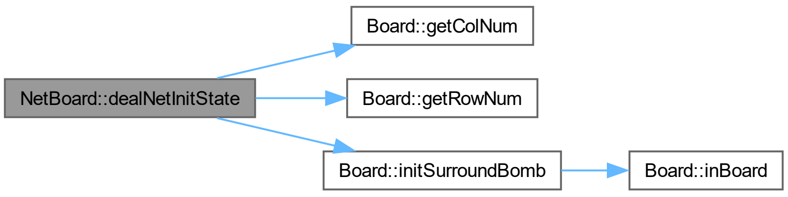
```
void NetBoard::dealNetInitState (
            QStringList list )  [virtual], [slot]
```

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]
```

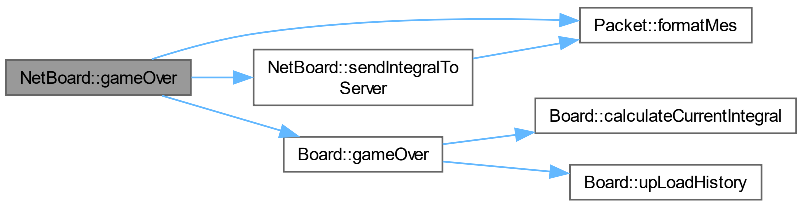
NetBoard::gameOver 重载游戏结束事件,

参数

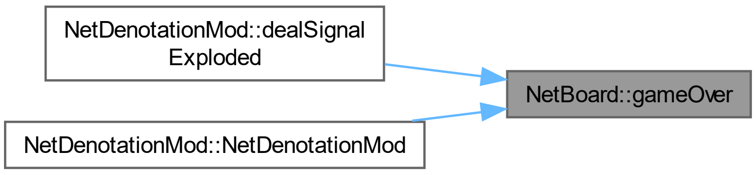
loseOrWin	

重载 Board .

函数调用图:



这是这个函数的调用关系图:

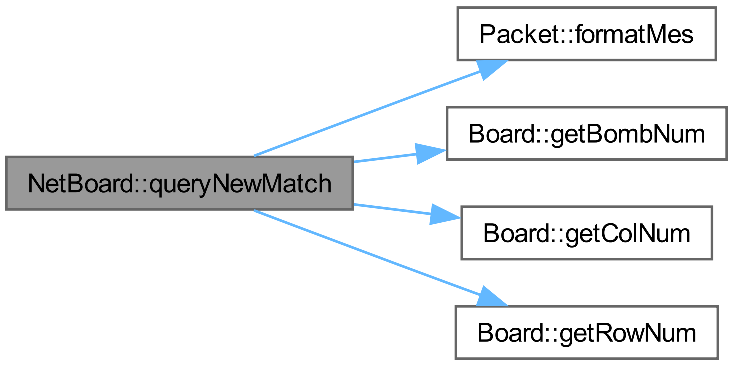

NetDenotationMod::dealSignal Exploded
NetDenotationMod::NetDenotationMod
NetBoard::gameOver

7.9.3.8 queryNewMatch()

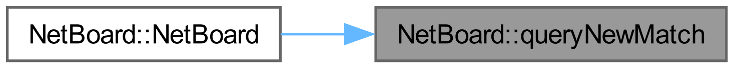
```
void NetBoard::queryNewMatch ( )  [protected], [virtual]
```

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()

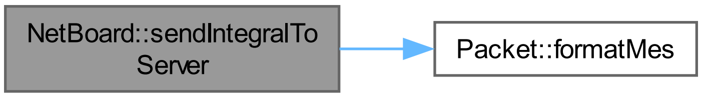
```
void NetBoard::sendIntegralToServer (
             qint32 integral )  [protected], [virtual]
```

NetBoard::sendIntegralToServer 发送自己的积分到服务器

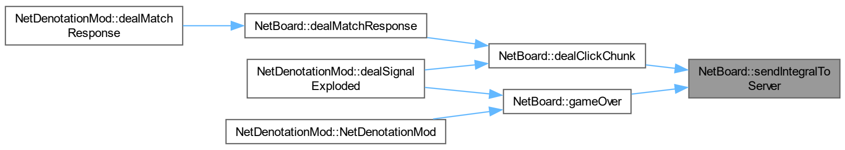
参数

integral	

函数调用图:



这是这个函数的调用关系图:



NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetDenotationMod::dealSignal Exploded
NetDenotationMod::NetDenotationMod
NetBoard::dealClickChunk
NetBoard::gameOver
NetBoard::sendIntegralTo Server

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:

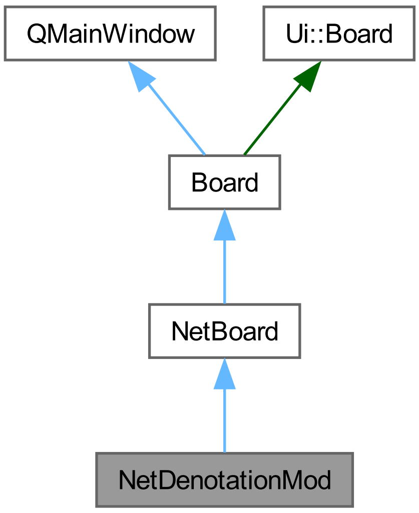
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.cpp

7.10 NetDenotationMod类 参考

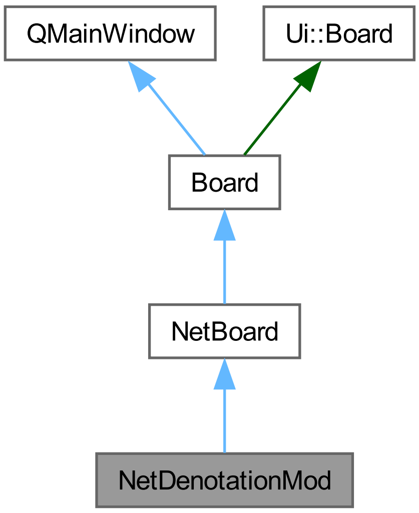
The NetDenotationMod class 爆炸模式网络对战

```
#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()

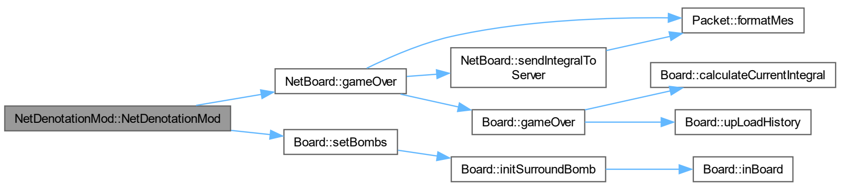
```
NetDenotationMod::NetDenotationMod (
            QTcpSocket * socket,
            qint32 rowNum,
            qint32 colNum,
            qint32 bombNum,
            QWidget * parent = nullptr,
            QString GameMod = "NetDenotation" )
```

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

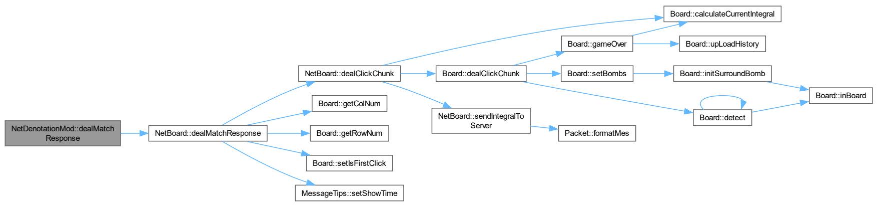
```
void NetDenotationMod::dealMatchResponse (
             QStringList list )  [override], [protected], [virtual], [slot]
```

NetDenotationMod::dealMatchResponse

参数

list	

函数调用图:

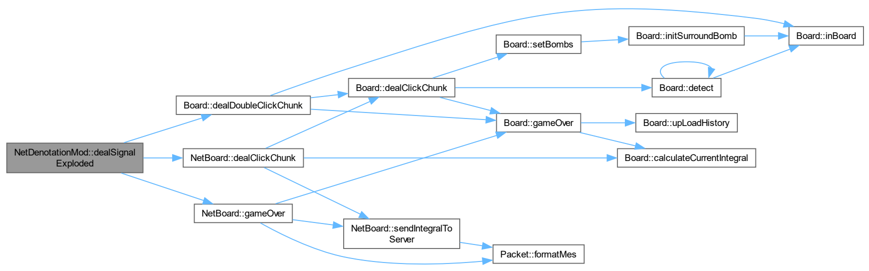


7.10.3.2 dealSignalExploded

```
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(-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,install↩ClassFunctionEvent 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,QString↩List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::*call↩Back)(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,install↩ClassFunctionEvent 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,QString↩List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

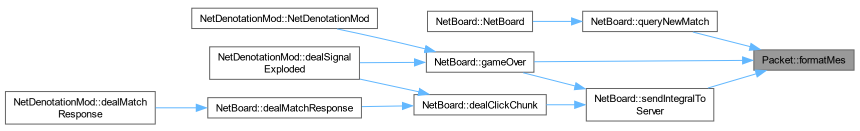
7.11.3.1 formatMes() [1/2]

```
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]
```

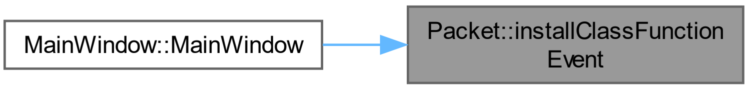
这是这个函数的调用关系图:


NetDenotationMod::NetDenotationMod
NetBoard::NetBoard
NetBoard::queryNewMatch
NetDenotationMod::dealSignal Exploded
NetBoard::gameOver
Packet::formatMes
NetDenotationMod::dealMatch Response
NetBoard::dealMatchResponse
NetBoard::dealClickChunk
NetBoard::sendIntegralTo Server

7.11.3.3 installClassFunctionEvent()

```
template<typename T >
void Packet< T >::installClassFunctionEvent (
             QString funcName,
             qint32 parameterNum,
             void(T::*)(QStringList) callBack )  [virtual]
```

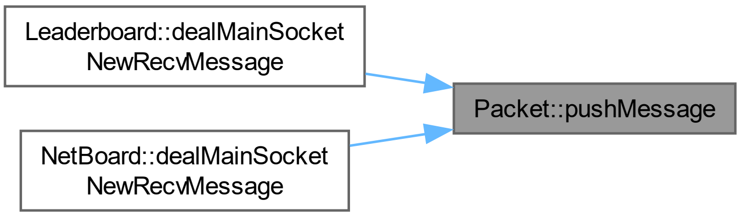
这是这个函数的调用关系图:



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:


Leaderboard::dealMainSocket NewRecvMessage
NetBoard::dealMainSocket NewRecvMessage
Packet::pushMessage

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:

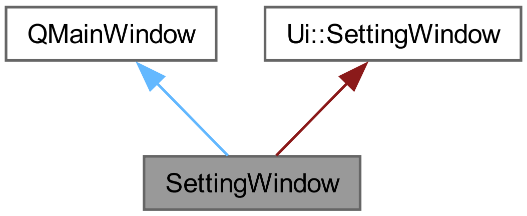
- C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h

7.13 SettingWindow类 参考

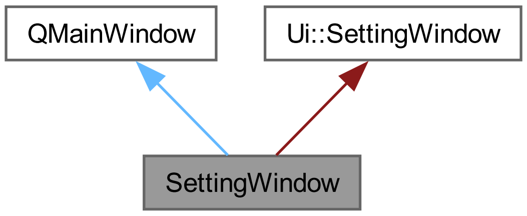
The SettingWindow class 设置界面

```
#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:

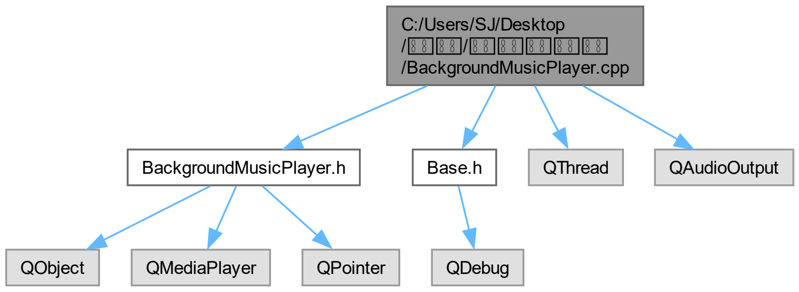
- C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h
- C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.cpp

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:

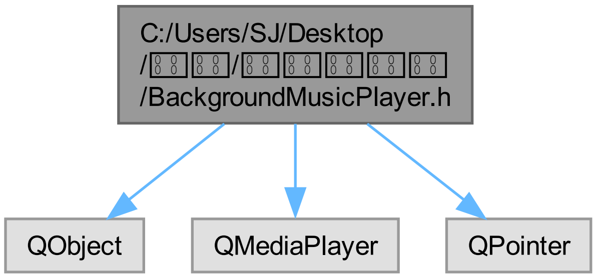


8.2 C:/Users/SJ/Desktop/扫雷/客户端源码/BackgroundMusicPlayer.h 文件参考

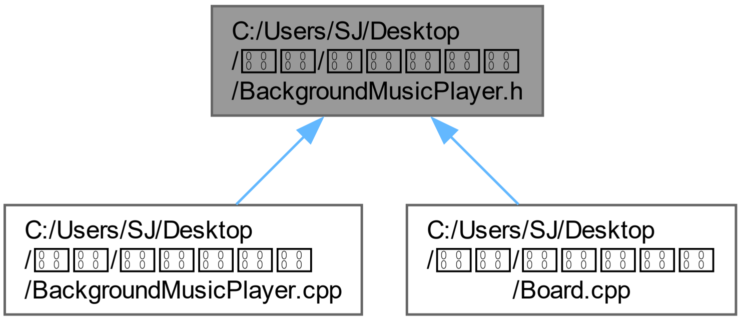
```
#include <QObject>
#include <QMediaPlayer>
```

```
#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

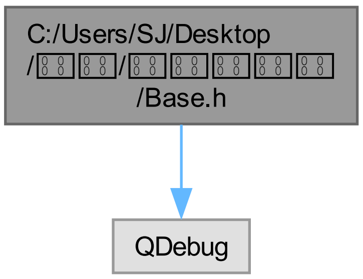
浏览该文件的文档.

```
#pragma once

#include <QObject>
#include<QMediaPlayer>
#include<QPointer>
class BackgroundMusicPlayer : public QObject
{
    Q_OBJECT
public:
    explicit BackgroundMusicPlayer(QObject *parent = nullptr);

private:
    QPointer<QMediaPlayer> player;
    QPointer<QAudioOutput>audioOutput;
signals:
public slots:
    virtual void playNewBackgroundMusc(QString path);
};

```

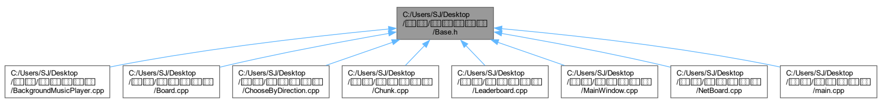
8.4 C:/Users/SJ/Desktop/扫雷/客户端源码/Base.h 文件参考

`#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:


C:/Users/SJ/Desktop/□□/□□□□□□/Base.h
C:/Users/SJ/Desktop/□□/□□□□□□/BackgroundMusicPlayer.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/Board.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/ChooseByDirection.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/Chunk.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/Leaderboard.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/MainWindow.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/NetBoard.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/main.cpp

宏定义

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

  摘要 调试宏
- #define dendl 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 dendl

```
#define dendl 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

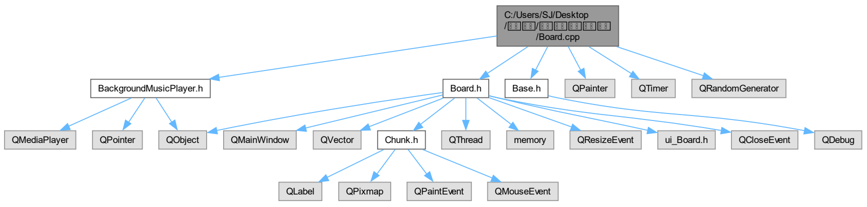
浏览该文件的文档.

```
#pragma once
#include<QDebug>
#define dout qDebug()<<"["<<__LINE__<<","<<__FUNCTION__<<","<<__FILE__<<"]"//< debug out (line,function name,file)
#define dendl Qt::endl//< debug endl
template<typename T>
inline bool inRange(T l,T r,T p)
{
    return l <= p && p <= r;
}
template<typename T>
inline bool inRect(T topx,T topy,T width,T height,T px,T py)
{
    return inRange<T>(topx,width+topx,px) && inRange<T>(topy,height+topy,py);
}
```

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

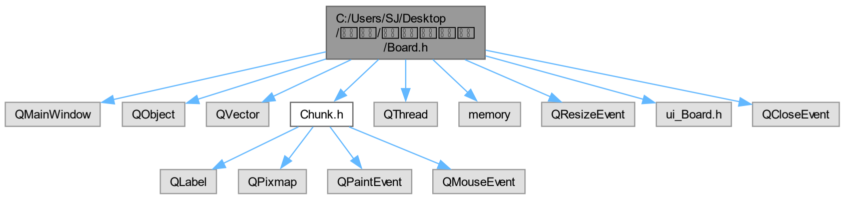
```
#include "Board.h"
#include "Base.h"
#include <QPainter>
#include <QTimer>
#include <QRandomGenerator>
```

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

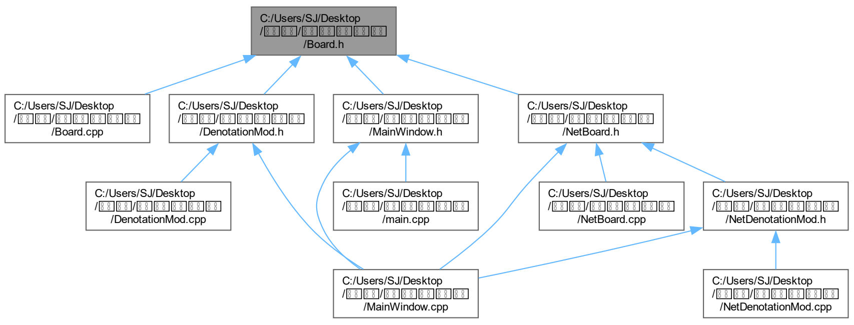


8.7 C:/Users/SJ/Desktop/扫雷/客户端源码/Board.h 文件参考

```
#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

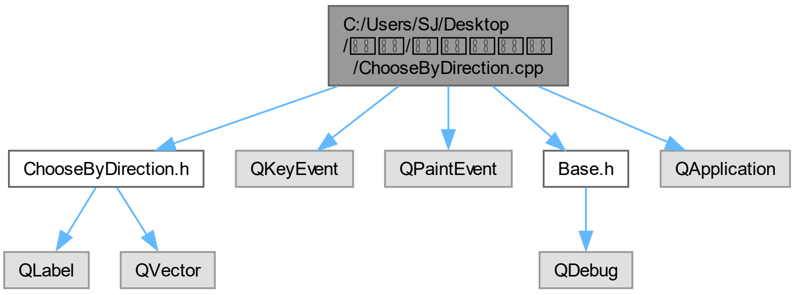
浏览该文件的文档.

```
#pragma once

#include <QMainWindow>
#include <QObject>
#include <QVector>
#include"Chunk.h"
#include<QThread>
#include<memory>
#include<QResizeEvent>
#include"ui_Board.h"
#include<QCloseEvent>
class DenotationMod;
class Board : public QMainWindow,protected Ui::Board
{
    Q_OBJECT
    friend class DenotationMod;
public:
    Board(qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent = nullptr,QString GameMod = "Classic");
    ~Board();
    qint32 getRowNum() const;

    qint32 getColNum() const;

    qint32 getBombNum() const;

    QPointer<QTimer> getGameTimer() const;

    void setIsFirstClick(bool newIsFirstClick);

protected:
    const QString GameMod;
    QVector<QVector<QPointer<Chunk>>>chunks;//<
    const qint32 rowNum;//<
    const qint32 colNum;//<
    const qint32 bombNum;//<
    qint32 flagBombNum=0;
    qint32 minedNum;
    const static qint32 SurroundDirectionNum = 8;//<
    constexpr static qint32 SurroundDirection[SurroundDirectionNum][2] = {
        {1,0},{0,1},{-1,0},{0,-1},
        {1,1},{1,-1},{-1,1},{-1,-1}};//<
    QPointer<QTimer> gameTimer;
    bool isFirstClick;//<
    qint32 selfCurrentIntegral=0;
private:

    std::unique_ptr<QPainter>painter();
//    QPixmap* pix;//<
    std::unique_ptr<QPixmap>pix;
    QPointer<QThread> backgroundPlayerThread;

private:
    void init();
protected:
    void setBombs(Chunk::RowCol firstClickedRC);
    virtual void detect(Chunk::RowCol rc);
    void initSurroundBomb();
    inline bool inBoard(Chunk::RowCol rc);//<
    inline bool inBoard(qint32 row,qint32 col);//<
    virtual void gameOver(QString loseOrWin);//<
    virtual void upLoadHistory();
    virtual qint32 calculateCurrentIntegral();
private slots:
    virtual void paintEvent(QPaintEvent* e)override;
    virtual void resizeEvent(QResizeEvent* e)override;
    virtual void closeEvent(QCloseEvent* e)override;
protected slots:
    //自定义事件槽
    virtual void dealSignalExploded();//<
    virtual void dealClickChunk(Chunk::RowCol rc);//<
    virtual void dealDoubleClickChunk(Chunk::RowCol rc,bool system=false);//<
signals:
    void signalGameOver();
    void signalPlayNewBackGroundMusic(QString path);
    void signalMove();
    void signalUpLoadHistory(QString gameMod,QString rowNum,QString colNum,QString bombNum,QString
integral);
};

```

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>
```

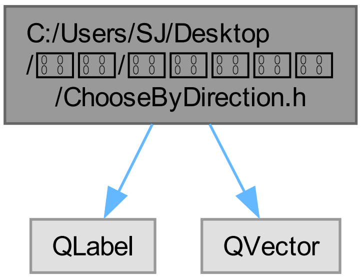
ChooseByDirection.cpp 的引用(Include)关系图:



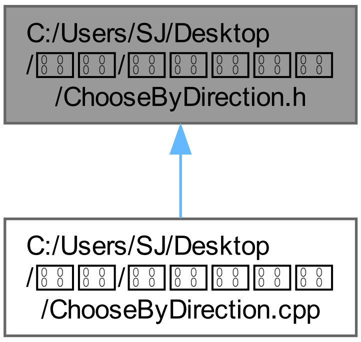
8.10 C:/Users/SJ/Desktop/扫雷/客户端源码/ChooseByDirection.h 文件参考

```
#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.

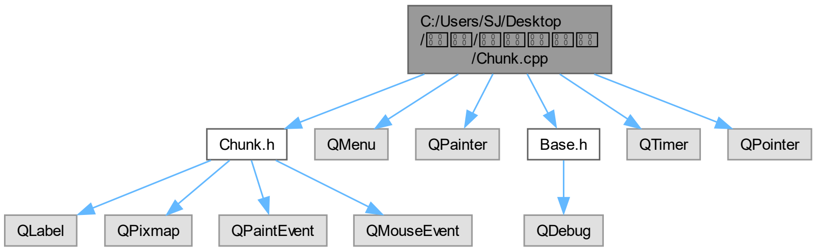
```
#pragma once

#include <QLabel>
#include<QVector>
class ChooseByDirection : public QLabel
{
    Q_OBJECT
public:
    ChooseByDirection(QWidget* parent);
public:
    virtual void setItems(const QVector<QString> &newItems);
    virtual void addItems(const QString& newItem);
    virtual const QString getCurrentItems()const;
private:
    QVector<QString> items;//
    qint32 indexForItems=0;
private slots:
    virtual void keyPressEvent(QKeyEvent* e)override;
    virtual void paintEvent(QPaintEvent*e)override;
    virtual void focusInEvent(QFocusEvent *e)override;
    virtual void focusOutEvent(QFocusEvent *e)override;
    virtual bool eventFilter(QObject *watched,QEvent *e)override;
    virtual void mousePressEvent(QMouseEvent* e)override;
    virtual void enterEvent(QEnterEvent *e)override;
//    virtual void leaveEvent(QEvent *e)override;//do nothing now
};

```

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>
```

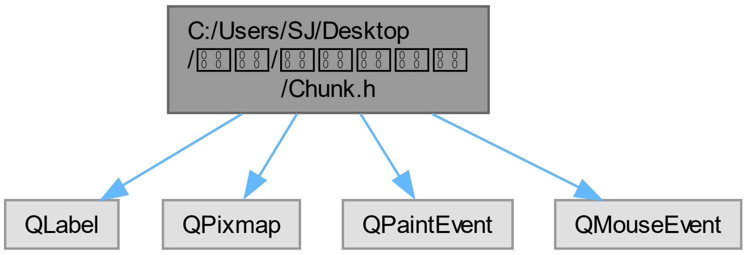
Chunk.cpp 的引用(Include)关系图:



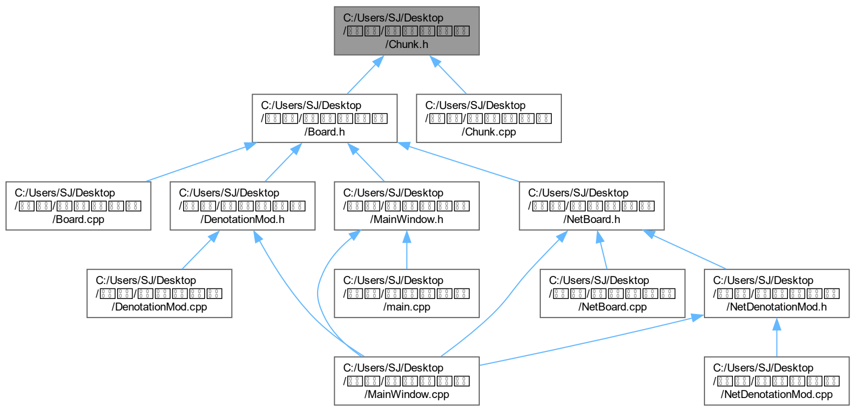
8.13 C:/Users/SJ/Desktop/扫雷/客户端源码/Chunk.h 文件参考

```
#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:


C:/Users/SJ/Desktop /Chunk.h
C:/Users/SJ/Desktop /Board.h
C:/Users/SJ/Desktop /Chunk.cpp
C:/Users/SJ/Desktop /Board.cpp
C:/Users/SJ/Desktop /DenotationMod.h
C:/Users/SJ/Desktop /MainWindow.h
C:/Users/SJ/Desktop /NetBoard.h
C:/Users/SJ/Desktop /DenotationMod.cpp
C:/Users/SJ/Desktop /main.cpp
C:/Users/SJ/Desktop /NetBoard.cpp
C:/Users/SJ/Desktop /NetDenotationMod.h
C:/Users/SJ/Desktop /MainWindow.cpp
C:/Users/SJ/Desktop /NetDenotationMod.cpp

类

- class Chunk

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

8.14 Chunk.h

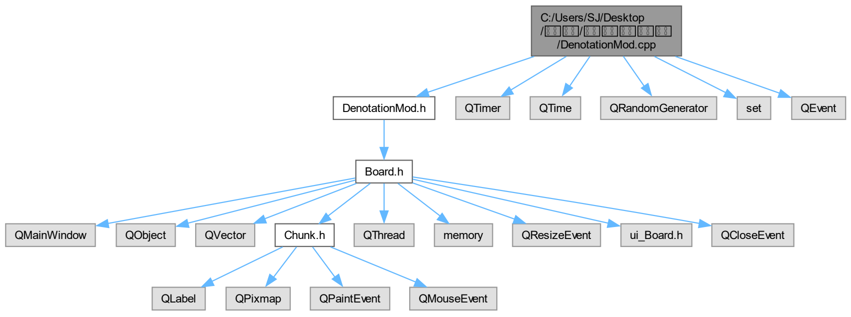
浏览该文件的文档.

```
#pragma once
#include <QLabel>
#include<QPixmap>
#include<QPaintEvent>
#include<QMouseEvent>
class Chunk : public QLabel
{
    Q_OBJECT
public:
    struct RowCol
    {
        qint32 row;
        qint32 col;
        RowCol(qint32 row = 0,qint32 col = 0):row(row),col(col) {};
        friend bool operator<(const RowCol& l,const RowCol& r)
        {
            if(l.row == r.row)
            {
                return l.col < r.col;
            }
            return l.row < r.row;
        }
//         friend bool operator==(const RowCol& l,const RowCol& r)
//         {
//             return (r.row == l.row) && (l.col == r.col);
//         }
    };
    enum class MineType{Bomb,NotBomb};//<
    enum class MineState{UnMined, Mined, FlagBomb, FlagQuestion};//<
    enum class RIGHT_KEY_MENU{FlagBomb,FlagQuestion};
public:
    explicit Chunk(QWidget *parent = nullptr);
//     Chunk& operator=(const Chunk& h);
//     Chunk(const Chunk& h);
public:
    virtual void setRowCol(qint32 row,qint32 col);
    const qint32 getChunkSize();
    virtual void setMineType(const MineType mt);
    virtual MineType getMineType()const;
    virtual qint32 getSurroundBomb() const;
    virtual void setSurroundBomb(qint32 newSurroundBomb);
    virtual const RowCol getRowCol() const;
    virtual MineState getMineState() const;
    virtual void setMineState(MineState newMineState);
    virtual void floatByDoubleClick();

//    void setClickable(bool newClickable);
    virtual void showBomb();
    virtual void openThenShow();
    void setPix(const QPixmap &newPix);

    const QPixmap &getPix() const;

private:
    virtual void drawSurroundBombNum(qint32 num);
    [[deprecated]] virtual void initRightKeyMenu();
    virtual void onTaskBoxContextMenuEvent();
private:
    virtual void paintEvent(QPaintEvent* e)override;
    virtual void mousePressEvent(QMouseEvent* e)override;
    virtual void mouseDoubleClickEvent(QMouseEvent* e)override;
private:
    const static qint32 DirectorNum = 4;//<
private:
    QPixmap pix;//<全局画布,
    QSize pixSize{500,500};
    RowCol rowCol;//<
    MineType mineType = MineType::NotBomb;//<
    MineState mineState = MineState::UnMined;//<
    qint32 surroundBomb = 0;//<
    const qint32 director[DirectorNum][2] = {{1,0},{0,1},{-1,0},{0,-1}};//<
    bool clickable = true;
signals:
    void signalExploded();//<
    void signalClickChunk(Chunk::RowCol rc);//<
    void signalDoubleClickChunk(Chunk::RowCol rc,bool system=false);//<
    void signalFlagBombChanged(qint32 changedNum);
};
```

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>
```

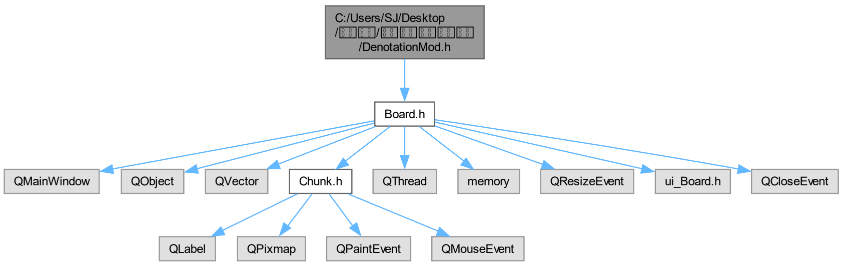
DenotationMod.cpp 的引用(Include)关系图:



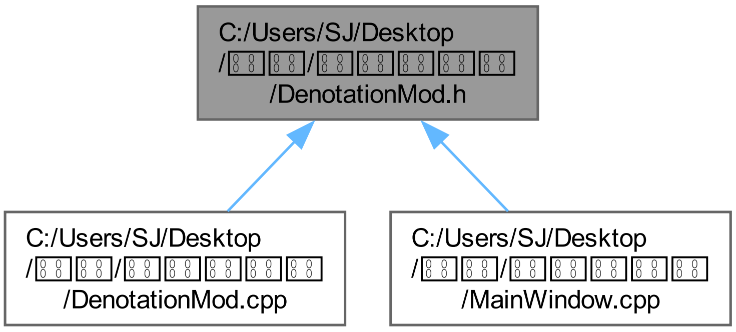
8.16 C:/Users/SJ/Desktop/扫雷/客户端源码/DenotationMod.h 文件参考

`#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

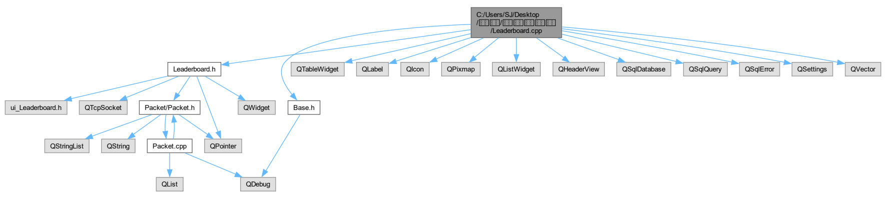
浏览该文件的文档.

```
#pragma once

#include "Board.h"
//#include"Chunk.h"
class DenotationMod : public Board
{
    Q_OBJECT
public:
    DenotationMod(qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent = nullptr,QString GameMod
= "Denotation");
private:
    virtual void randomOpenNotBombChunk(qint32 num);
private slots:
    virtual void dealSignalExploded()override;
//    virtual void upLoadHistory()override;
private:
    virtual bool eventFilter(QObject *watched,QEvent *e)override;
private:
//    const QString GameMod = "Denotation";
    qint32 moveNum;
    qint32 remainBombNum;
};

```

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>
```

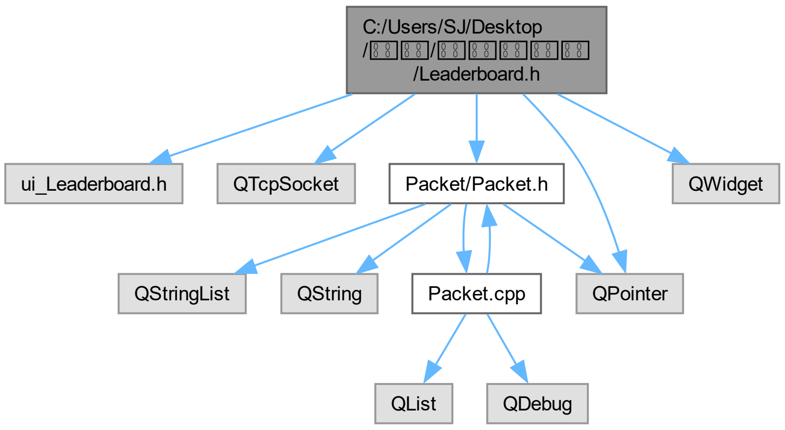
Leaderboard.cpp 的引用(Include)关系图:



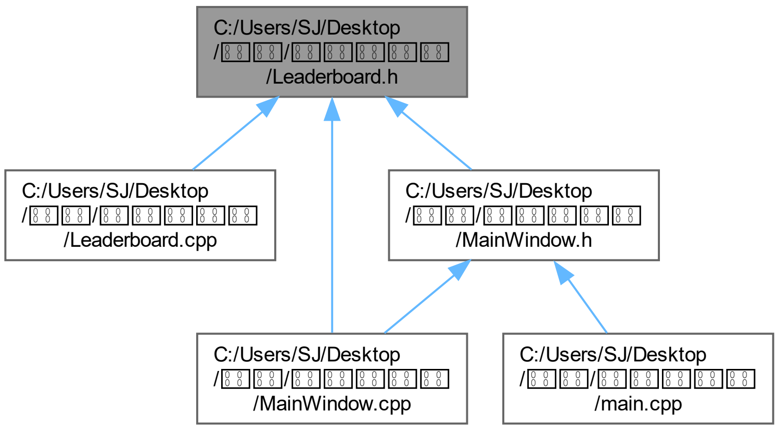
8.19 C:/Users/SJ/Desktop/扫雷/客户端源码/Leaderboard.h 文件参考

```
#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:


C:/Users/SJ/Desktop /Leaderboard.h
C:/Users/SJ/Desktop /Leaderboard.cpp
C:/Users/SJ/Desktop /MainWindow.h
C:/Users/SJ/Desktop /MainWindow.cpp
C:/Users/SJ/Desktop /main.cpp

类

- class Leaderboard

  The Leaderboard class 显示历史战绩

8.20 Leaderboard.h

浏览该文件的文档.

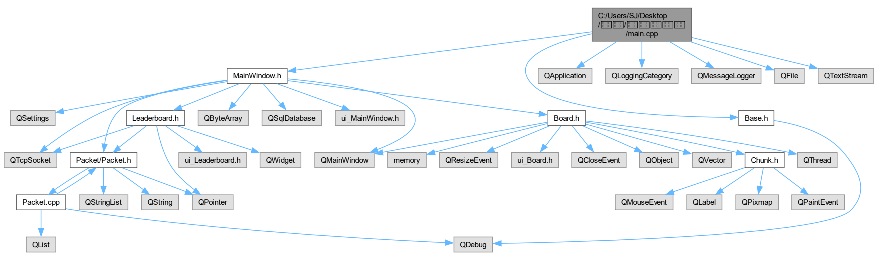
```
00001 #pragma once
00002
```

```
#include "ui_Leaderboard.h"
#include<QTcpSocket>
#include"Packet/Packet.h"
#include<QWidget>
#include<QPointer>
class Leaderboard : public QMainWindow, public Ui::Leaderboard
{
    Q_OBJECT

public:
    explicit Leaderboard(QTcpSocket* socket,QWidget *parent = nullptr);
public slots:
    virtual void dealMainSocketNewRecvMessage(QByteArray mes);
private:
    QTcpSocket* socket;
    friend class Packet<Leaderboard>;
    Packet<Leaderboard> packet;
private:
    QPointer<QWidget>allHistoryWidget=nullptr;
    QPointer<QWidget>classicOrderWidget=nullptr;
    QPointer<QWidget>denotationOrderWidget=nullptr;
    QPointer<QWidget>netBoardOrderWidget=nullptr;
    QPointer<QWidget>netDenotationOrderWidget=nullptr;
    void initAllHistoryWidget();
    void initSelfOrderWidget(QWidget* widget,QString gameMod);
    QString formatEachGameItem(QString matchID,QString Date,
        QString gameMod,QString rowNum,QString colNum,QString bombNum,
        QString selfEmail,QString antiEmail,QString selfIntegral,QString antiIntegral);
};

```

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

```
#include "MainWindow.h"
#include <QApplication>
#include <QLoggingCategory>
#include <QMessageLogger>
#include <QFile>
#include <QTextStream>
#include "Base.h"
```

main.cpp 的引用(Include)关系图:



函数

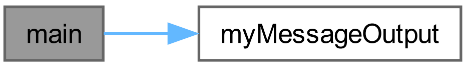
- void myMessageOutput (QtMsgType type, const QMessageLogContext &context, const QString &msg)
- int main (int argc, char ∗argv[ ])

8.21.1 函数说明

8.21.1.1 main()

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

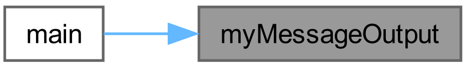
函数调用图:



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:

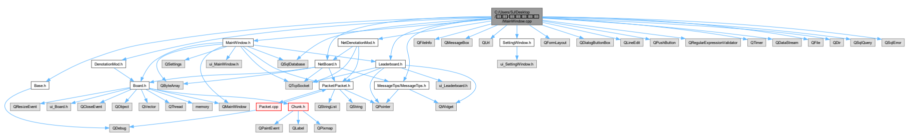


8.22 C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.cpp 文件参考

```
#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>
```

```
#include <QRegularExpressionValidator>
#include "Leaderboard.h"
#include "NetBoard.h"
#include "MessageTips/MessageTips.h"
#include "NetDenotationMod.h"
#include <QTimer>
#include <QDataStream>
#include <QFile>
#include <QDir>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
```

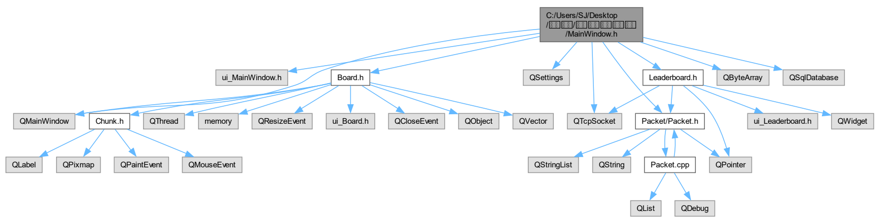
MainWindow.cpp 的引用(Include)关系图:



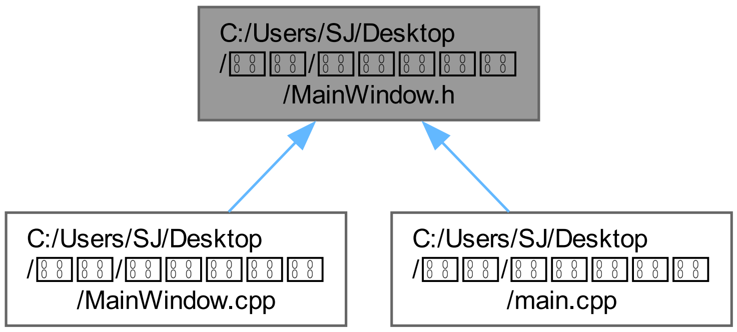
8.23 C:/Users/SJ/Desktop/扫雷/客户端源码/MainWindow.h 文件参考

```
#include "ui_MainWindow.h"
#include <QMainWindow>
#include "Board.h"
#include <QSettings>
#include <QTcpSocket>
#include "Packet/Packet.h"
#include "Leaderboard.h"
#include <QByteArray>
#include <QSqlDatabase>
```

MainWindow.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:


C:/Users/SJ/Desktop/□□/□□□□□□/MainWindow.h
C:/Users/SJ/Desktop/□□/□□□□□□/MainWindow.cpp
C:/Users/SJ/Desktop/□□/□□□□□□/main.cpp

类

- class MainWindow

  The MainWindow class 主窗口对象

8.24 MainWindow.h

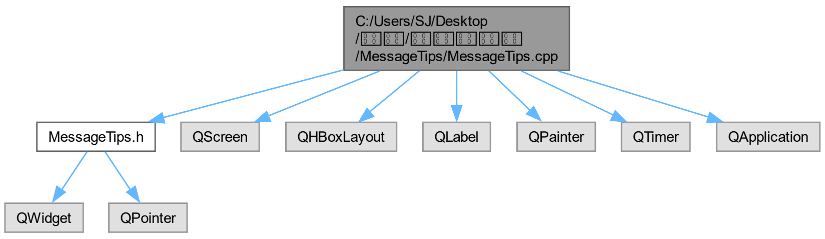
浏览该文件的文档.

```
#pragma once

#include "ui_MainWindow.h"
#include <QMainWindow>
#include"Board.h"
#include<QSettings>
#include<QTcpSocket>
#include"Packet/Packet.h"
#include"Leaderboard.h"
#include<QByteArray>
#include<QSqlDatabase>
class MainWindow : public QMainWindow, private Ui::MainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
private:
    void init();
    qint32 getSettingsIntValue(QString sectinName,QString ValueName);
    virtual bool tryLogin(QString email,QString password);
    virtual void showSignUpAndLogin();
    virtual void showCaptcha();
    virtual void downloadRemoteHistoryFile();
private slots:
    void on_pushButtonBeginGame_clicked();
    void on_pushButtonSetting_clicked();
    void on_pushButtonDenotation_clicked();
    void on_pushButtonLeaderboard_clicked();
private slots:
    virtual void dealConnected();
    virtual void dealDisconnected();
    virtual void dealRecv();
    virtual void dealGameOver();
private:
//    const QString IP = "101.42.8.164";//服务器ip
    const QString IP = "127.0.0.1";//连接本地测试
    Board* board;
    QPointer<Leaderboard> leaderBoard;
    QTcpSocket* socket;
    QString difficulty;
    bool isLogin = false;
    QSqlDatabase database;
private:
    friend class Packet<MainWindow>;
    Packet<MainWindow> packet;
    virtual void dealLoginResponse(QStringList list);
    virtual void dealSignUpResponse(QStringList list);
    virtual void dealCaptchaResponse(QStringList list);
    virtual void dealSignalUpLoadHistory(QString gameMod,QString rowNum,QString colNum,QString
bombNum,QString integral);
    virtual void dealTansferHistoryFileHead(QStringList list);
    virtual void dealTansferingHistoryFile(QStringList list);
    virtual void dealTansferHistoryFileEnd(QStringList list);
signals:
    void signalMainSocketNewRecvMessage(QByteArray mes);
};
```

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>
```

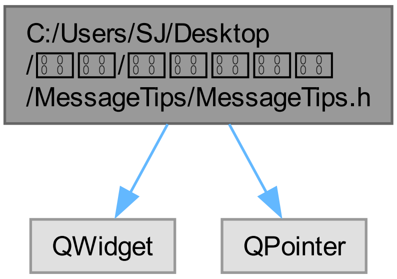
MessageTips.cpp 的引用(Include)关系图:



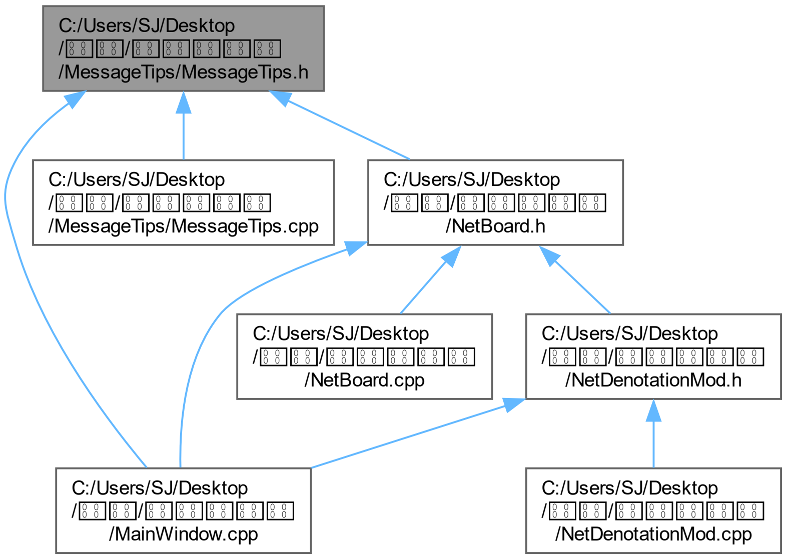
8.26 C:/Users/SJ/Desktop/扫雷/客户端源码/MessageTips/MessageTips.h 文件参考

```
#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

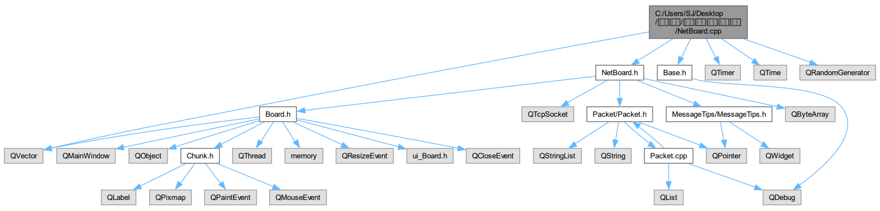
浏览该文件的文档.

```
#ifndef MESSAGETIPS_H
#define MESSAGETIPS_H

#include <QWidget>
#include<QPointer>
//#pragma execution_character_set(push) // push the previous character
# pragma execution_character_set("utf-8")

class QHBoxLayout;
class QLabel;
class MessageTips : public QWidget
{
    Q_OBJECT
public:
    explicit MessageTips(QString showStr="none", QWidget *parent = nullptr);
    ~MessageTips();
    double getOpacityValue() const;
    void setOpacityValue(double value);

    qint32 getTextSize() const;
    void setTextSize(int value);

    QColor getTextColor() const;
    void setTextColor(const QColor &value);

    QColor getBackgroundColor() const;
    void setBackgroundColor(const QColor &value);

    QColor getFrameColor() const;
    void setFrameColor(const QColor &value);

    qint32 getFrameSize() const;
    void setFrameSize(int value);

    qint32 getShowTime() const;
    void setShowTime(int msec);

    void setCloseTimeSpeed(int closeTime = 100,double closeSpeed = 0.1);
public:
//    virtual void show()override;
protected:
    void paintEvent(QPaintEvent *event) override;

private:
    void InitLayout();//初始化窗体的布局和部件
    QPointer<QHBoxLayout> hBoxlayout;//布局显示控件布局
    QPointer<QLabel> mText;//用于显示文字的控件
    QString showStr;//显示的字符串

    double opacityValue;//窗体初始化透明度
    QFont* font;
    qint32     textSize;//显示字体大小
    QColor  textColor;//字体颜色
    QColor  backgroundColor;//窗体的背景色
    QColor  frameColor;//边框颜色
    qint32     frameSize;//边框粗细大小

    qint32     showTime;//显示时间
    qint32     closeTime;//关闭需要时间
    double  closeSpeed;//窗体消失的平滑度，大小0~1

signals:

};
//#pragma execution_character_set(pop) // pop the previous character set
#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>
```

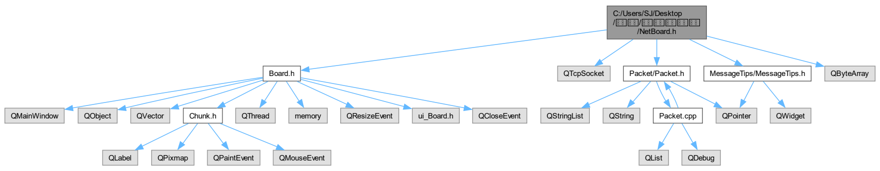
NetBoard.cpp 的引用(Include)关系图:



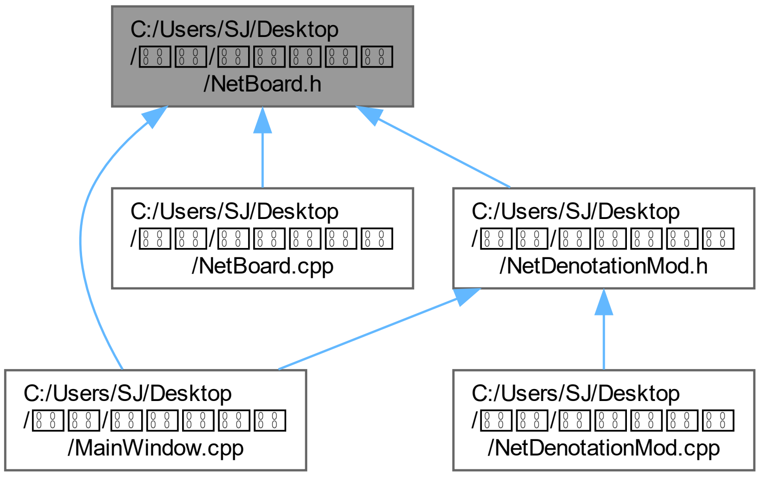
8.29 C:/Users/SJ/Desktop/扫雷/客户端源码/NetBoard.h 文件参考

```
#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

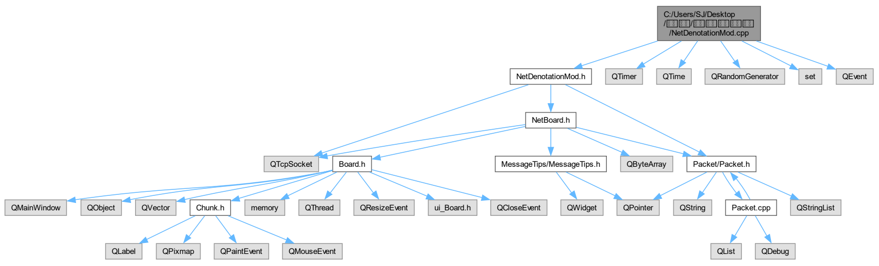
浏览该文件的文档.

```
#pragma once

#include "Board.h"
#include<QTcpSocket>
#include"Packet/Packet.h"
#include"MessageTips/MessageTips.h"
#include<QByteArray>
class NetBoard : public Board
{
    Q_OBJECT
public:
    NetBoard(QTcpSocket* socket,qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent =
nullptr,QString GameMod = "NetBoard");
private:
    void initPacket();

    virtual void initNetBoardThenSendBySocket();
    QString generateGameStateStr(qint32 selfIntegral,qint32 antiIntegral,QString antiEmail);
protected:
    virtual void gameOver(QString loseOrWin)override;//<
    virtual void sendIntegralToServer(qint32 integral);
    virtual void queryNewMatch();
private:
    friend class Packet<NetBoard>;
    QTcpSocket* socket;
    Packet<NetBoard> packet;
    QString antiPlayerEmail;
    qint32 antiCurrentIntegral=0;
protected slots:
    virtual void dealMatchResponse(QStringList list);
    virtual void dealClickChunk(Chunk::RowCol rc)override;//<
public slots:
    virtual void dealMainSocketNewRecvMessage(QByteArray mes);
    virtual void dealNetInitState(QStringList list);
    virtual void dealUpdateAntiIntegral(QStringList list);
    virtual void dealAntiGameOver(QStringList list);
private:
    MessageTips* matchingMessageTips;
};

```

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>
```

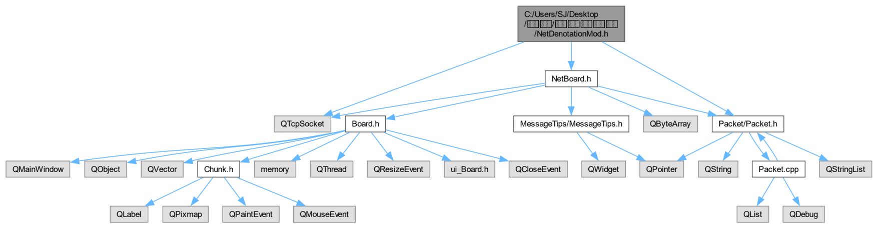
NetDenotationMod.cpp 的引用(Include)关系图:



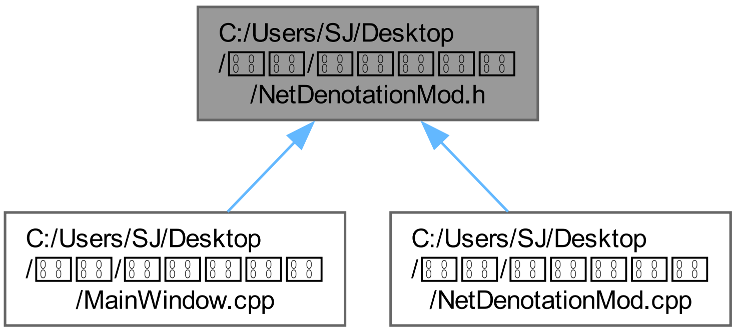
8.32 C:/Users/SJ/Desktop/扫雷/客户端源码/NetDenotationMod.h 文件参考

```
#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

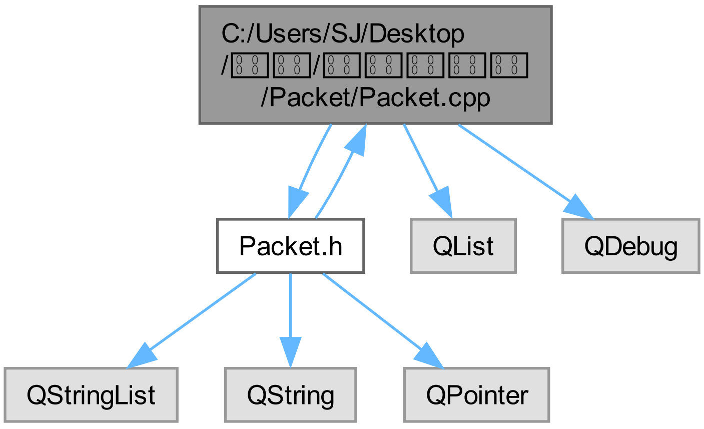
浏览该文件的文档.

```
#pragma once
#include "NetBoard.h"
#include<QTcpSocket>
#include"Packet/Packet.h"
class NetDenotationMod : public NetBoard
{
    Q_OBJECT
public:
    NetDenotationMod(QTcpSocket* socket,qint32 rowNum,qint32 colNum,qint32 bombNum,QWidget *parent =
nullptr,QString GameMod = "NetDenotation");
protected:
private:
    void randomOpenNotBombChunk(qint32 num);
protected slots:
    virtual void dealSignalExploded()override;
    virtual void dealMatchResponse(QStringList list)override;
private:
    virtual bool eventFilter(QObject *watched,QEvent *e)override;
//    virtual void detect(Chunk::RowCol rc)override;
private:
    qint32 moveNum;
    qint32 remainBombNum;
    QTcpSocket* socket;
    friend class Packet<NetDenotationMod>;
    Packet<NetDenotationMod> packet;
};
```

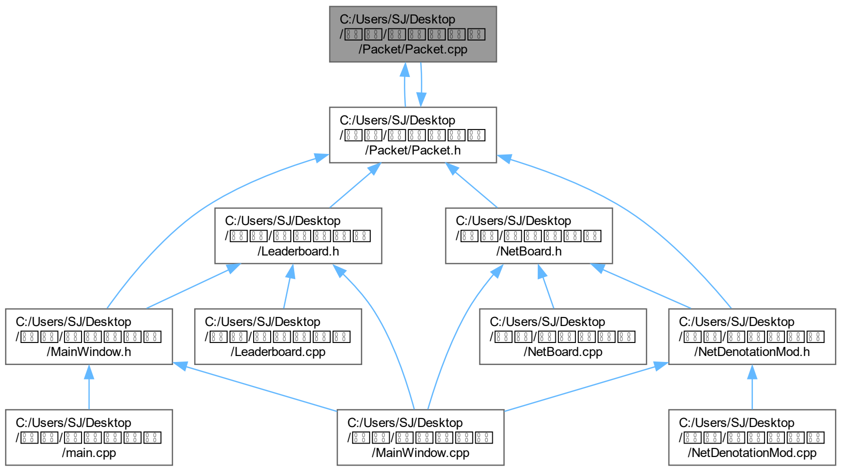
8.34 C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.cpp 文件参考

```
#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:


C:/Users/SJ/Desktop /Packet/Packet.cpp
C:/Users/SJ/Desktop /Packet/Packet.h
C:/Users/SJ/Desktop /Leaderboard.h
C:/Users/SJ/Desktop /NetBoard.h
C:/Users/SJ/Desktop /MainWindow.h
C:/Users/SJ/Desktop /Leaderboard.cpp
C:/Users/SJ/Desktop /NetBoard.cpp
C:/Users/SJ/Desktop /NetDenotationMod.h
C:/Users/SJ/Desktop /main.cpp
C:/Users/SJ/Desktop /MainWindow.cpp
C:/Users/SJ/Desktop /NetDenotationMod.cpp

宏定义

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

8.34.1 宏定义说明

8.34.1.1 dendl

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

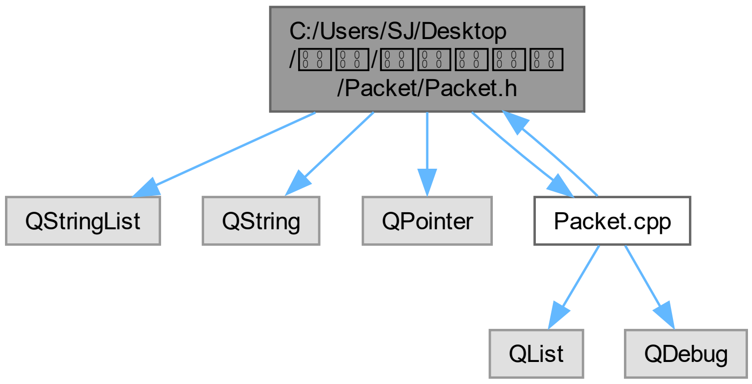
8.34.1.2 dout

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

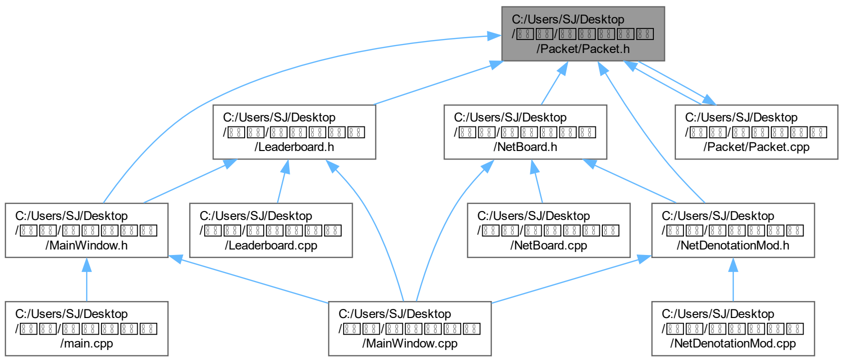
8.35 C:/Users/SJ/Desktop/扫雷/客户端源码/Packet/Packet.h 文件参考

```
#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,install↵ ClassFunctionEvent 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,QString↵ List为参数

8.36 Packet.h

浏览该文件的文档.

```
// Packet.h -- 模板类的声明
#pragma once
#include <QStringList>
#include <QString>
#include<QPointer>
template<typename T>
class Packet
{
    struct FunctionEvent
    {
    public:
        QString funcName;
        qint32 parameterNum;
        void (T::*callBack)(QStringList);
        FunctionEvent(QString funcName, qint32 parameterNum, void (T::*callBack)(QStringList))
            :funcName(funcName),parameterNum(parameterNum),callBack(callBack){};
    };
public:
    Packet(T* parent);
    virtual void pushMessage(QString newMes);//压入信息, 可能会触发callBack
    virtual QString formatMes(QStringList newMesList);//<将消息封装
    virtual QString formatMes(QString newMes);//<重载
    virtual void installClassFunctionEvent(QString funcName, qint32 parameterNum, void
(T::*callBack)(QStringList));
private:
    virtual void distributerEvent();
    virtual inline QStringList splitMes(QString mes);
private:
    static const QString separator;
    QStringList recvList;
    QString recvBuff;
    T* parent;
    //warning:这里使用普通指针我也不知道会出什么问题不
    QList<FunctionEvent*> funcEvents;
};

#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 ()

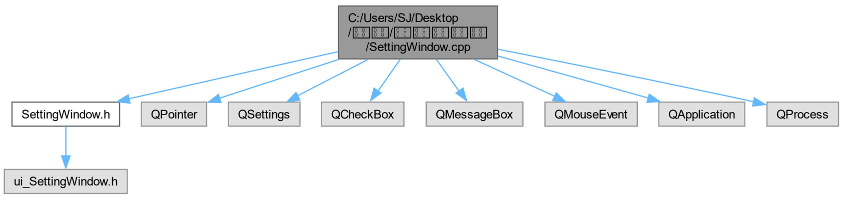
变量

- 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>
```

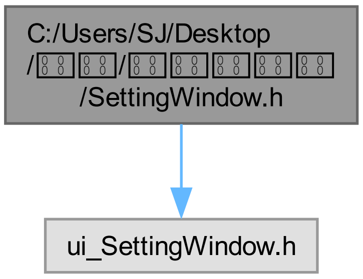
SettingWindow.cpp 的引用(Include)关系图:



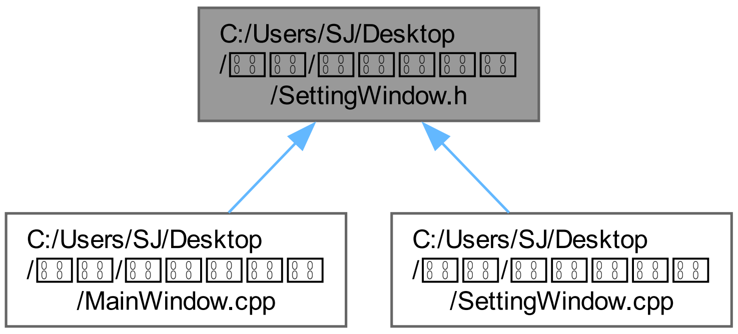
8.39 C:/Users/SJ/Desktop/扫雷/客户端源码/SettingWindow.h 文件参考

```
#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.

```
#pragma once

#include "ui_SettingWindow.h"
class SettingWindow : public QMainWindow, private Ui::SettingWindow
{
    Q_OBJECT

public:
    explicit SettingWindow(QWidget *parent = nullptr);
    virtual void reloadApp();
private slots:
    void on_pushButtonCancel_clicked();
    void on_pushButtonOk_clicked();
    virtual void mousePressEvent(QMouseEvent* e)override;
    void on_pushButtonExitLogin_clicked();
};

```

Index