Battleship

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AudioPlayer Class Reference

`#include <AudioPlayer.h>`

Public Types

- enum Clip {
  ButtonClick , SelectShip , PlaceShip , Cannon ,
  Hit , Miss , GameOver , PopUp }

Static Public Member Functions

- static void play (const Clip &clip)
- static void play (const std::string &file)

4.1.1 Member Enumeration Documentation

4.1.1.1 Clip

`enum AudioPlayer::Clip`

Collection of general sound effect clips

Enumerator

ButtonClick	
SelectShip	
PlaceShip	
Cannon	
Hit	
Miss	
GameOver	
PopUp	

4.1.2 Member Function Documentation

4.1.2.1 play() [1/2]

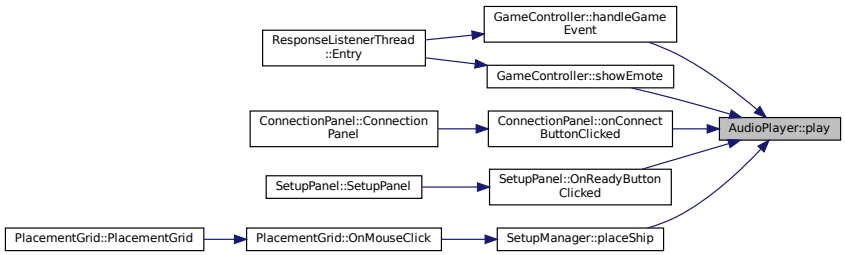
```
void AudioPlayer::play (
            const Clip & clip )  [static]
```

Play the selected audio clip

Parameters

clip	

Here is the caller graph for this function:



4.1.2.2 play() [2/2]

```
void AudioPlayer::play (
            const std::string & file )  [static]
```

Play a general audio file in .wav format. Use this to play the sound of emotes: AudioPlayer::play(EmoteHandler↩ ::getSound(emote));

Parameters

file	full path to the .wav-file (Has to be specific WAV format! Default output of Audacity works)"

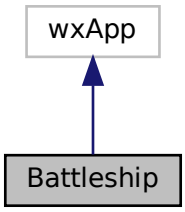
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/AudioPlayer.h
- /home/nico/Desktop/battleship/src/client/AudioPlayer.cpp

4.2 Battleship Class Reference

`#include <Battleship.h>`

Inheritance diagram for Battleship:



Public Member Functions

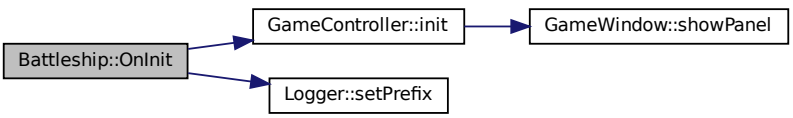
- auto OnInit () -> bool override

4.2.1 Member Function Documentation

4.2.1.1 OnInit()

`auto Battleship::OnInit ( ) -> bool [override]`

Here is the call graph for this function:



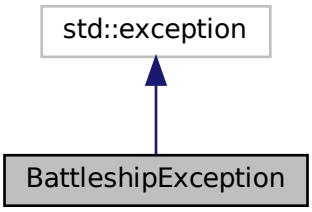
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/Battleship.h
- /home/nico/Desktop/battleship/src/client/Battleship.cpp

4.3 BattleshipException Class Reference

`#include <BattleshipException.h>`

Inheritance diagram for BattleshipException:



Public Member Functions

- BattleshipException (const std::string &message)
- const char * what () const noexcept override

4.3.1 Constructor & Destructor Documentation

4.3.1.1 BattleshipException()

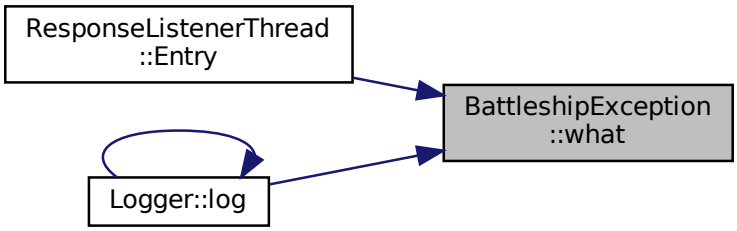
```
BattleshipException::BattleshipException (
            const std::string & message )  [inline], [explicit]
```

4.3.2 Member Function Documentation

4.3.2.1 what()

```
const char* BattleshipException::what ( ) const  [inline], [override], [noexcept]
```

Here is the caller graph for this function:



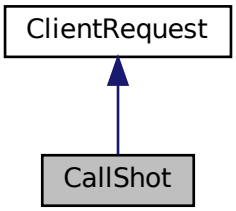
The documentation for this class was generated from the following file:

- /home/nico/Desktop/battleship/src/common/exceptions/BattleshipException.h

4.4 CallShot Class Reference

```
#include <CallShot.h>
```

Inheritance diagram for CallShot:



Public Member Functions

- CallShot (uuid playerId, Coordinate position)
- auto getPosition () const -> Coordinate

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 CallShot()

```
CallShot::CallShot (
            uuid playerId,
            Coordinate position )
```

4.4.2 Member Function Documentation

4.4.2.1 getPosition()

```
auto CallShot::getPosition ( ) const -> Coordinate
```

Here is the caller graph for this function:

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

- /home/nico/Desktop/battleship/src/common/network/requests/CallShot.h
- /home/nico/Desktop/battleship/src/common/network/requests/CallShot.cpp

4.5 ClientNetworkManager Class Reference

```
#include <ClientNetworkManager.h>
```

Static Public Member Functions

- static void init (const std::string &host, const uint16_t port)
- static void sendRequest (const ClientRequest &request)
- static std::unique_ptr< ServerResponse > parseResponse (const std::string &message)

4.5.1 Member Function Documentation

4.5.1.1 init()

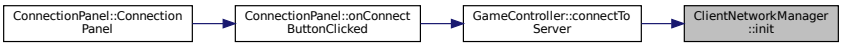
```
static void ClientNetworkManager::init (
            const std::string & host,
            const uint16_t port )  [static]
```

establishes a connection to the server specified by host address and port number

Parameters

host	server ip
port	server port

Here is the caller graph for this function:



4.5.1.2 parseResponse()

```
std::unique_ptr< ServerResponse > ClientNetworkManager::parseResponse (
            const std::string & message )  [static]
```

deserializes a server response

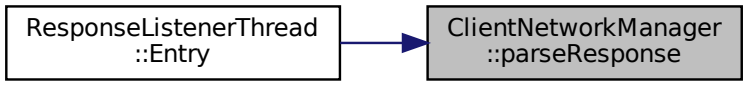
Parameters

message	server response in json format

Returns

response object

Here is the caller graph for this function:



4.5.1.3 sendRequest()

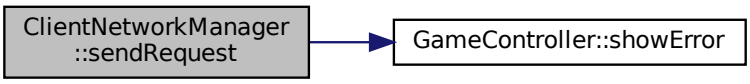
```
void ClientNetworkManager::sendRequest (
            const ClientRequest & request )  [static]
```

serializes and sends a client request to the server

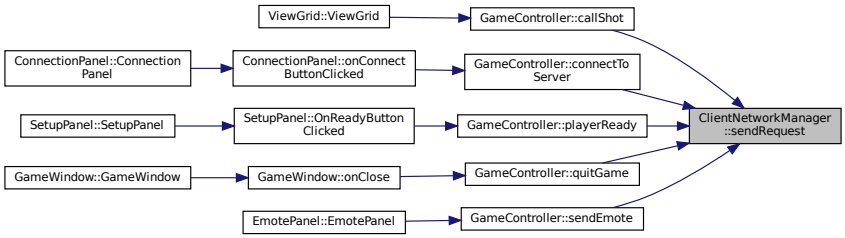
Parameters

request	message to the server

Here is the call graph for this function:



Here is the caller graph for this function:



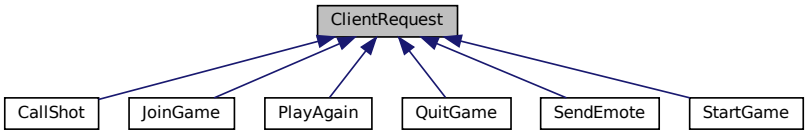
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/ClientNetworkManager.h
- /home/nico/Desktop/battleship/src/client/ClientNetworkManager.cpp

4.6 ClientRequest Class Reference

`#include <ClientRequest.h>`

Inheritance diagram for ClientRequest:



Public Member Functions

- auto getRequestType () const -> RequestType
- auto getPlayerId () const -> uuid
- virtual ~ClientRequest ()=default

Protected Member Functions

- ClientRequest (uuid playerId, RequestType requestType)

4.6.1 Constructor & Destructor Documentation

4.6.1.1 ~ClientRequest()

```
virtual ClientRequest::~ClientRequest ( )  [virtual], [default]
```

4.6.1.2 ClientRequest()

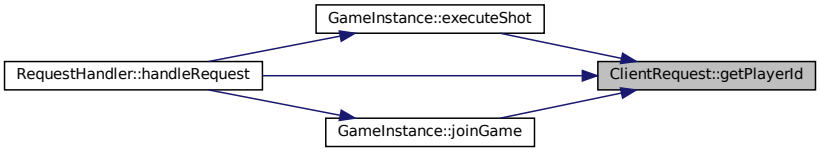
```
ClientRequest::ClientRequest (
            uuid playerId,
            RequestType requestType )  [protected]
```

4.6.2 Member Function Documentation

4.6.2.1 getPlayerId()

```
auto ClientRequest::getPlayerId ( ) const -> uuid
```

Here is the caller graph for this function:



4.6.2.2 getRequestType()

```
auto ClientRequest::getRequestType ( ) const -> RequestType
```

Here is the caller graph for this function:

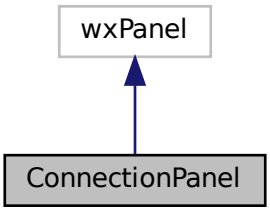
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/network/requests/ClientRequest.h
- /home/nico/Desktop/battleship/src/common/network/requests/ClientRequest.cpp

4.7 ConnectionPanel Class Reference

```
#include <ConnectionPanel.h>
```

Inheritance diagram for ConnectionPanel:



Public Member Functions

- ConnectionPanel (wxWindow *parent)

  Constructor for ConnectionPanel.
- void onConnectButtonClicked (wxCommandEvent &event)

  Button event handler. Will trigger GameController::connectToServer() to establish a connection to the server.
- wxString getServerAddress ()

  Getter for the server address.
- wxString getServerPort ()

  Getter for the server port.
- wxString getUserName ()

  Getter for the username.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 ConnectionPanel()

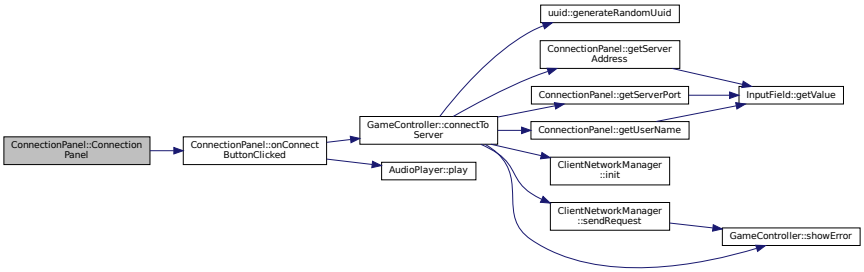
```
ConnectionPanel::ConnectionPanel (
            wxWindow * parent )
```

Constructor for ConnectionPanel.

Parameters

parent	

Here is the call graph for this function:



4.7.2 Member Function Documentation

4.7.2.1 getServerAddress()

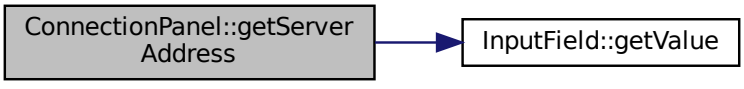
```
wxString ConnectionPanel::getServerAddress ( )
```

Getter for the server address.

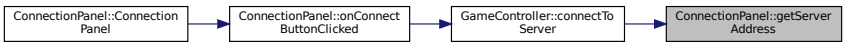
Returns

wxString

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.2.2 getServerPort()

```
wxString ConnectionPanel::getServerPort ( )
```

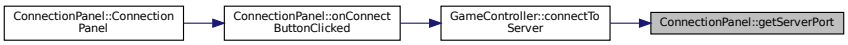
Getter for the server port.

Returns

wxString

Here is the call graph for this function:

Here is the caller graph for this function:



4.7.2.3 getUserName()

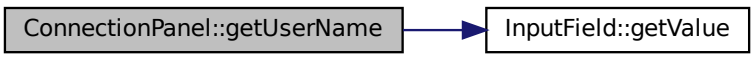
```
wxString ConnectionPanel::getUserName ( )
```

Getter for the username.

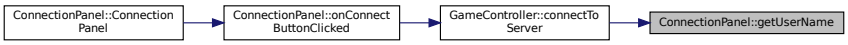
Returns

wxString

Here is the call graph for this function:



Here is the caller graph for this function:

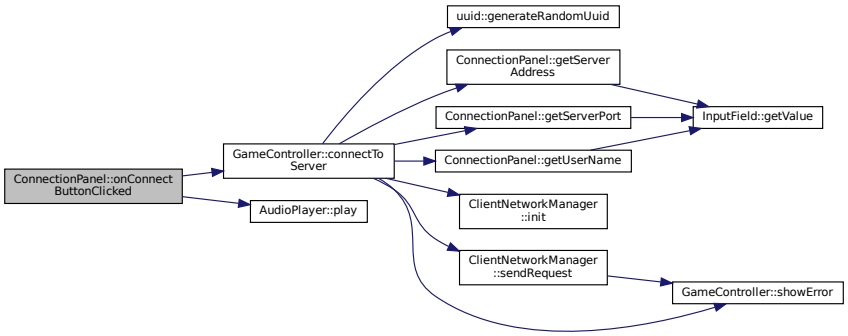


4.7.2.4 onConnectButtonClicked()

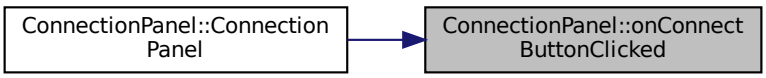
```
void ConnectionPanel::onConnectButtonClicked (
            wxCommandEvent & event )
```

Button event handler. Will trigger GameController::connectToServer() to establish a connection to the server.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /home/nico/Desktop/battleship/src/client/panels/ConnectionPanel.h
- /home/nico/Desktop/battleship/src/client/panels/ConnectionPanel.cpp

4.8 Coordinate Struct Reference

`#include <Coordinate.h>`

Public Member Functions

- auto operator<=> (const Coordinate &) const =default

Public Attributes

- int x = 0
- int y = 0

4.8.1 Member Function Documentation

4.8.1.1 operator<=>()

```
auto Coordinate::operator<=> (
            const Coordinate &  ) const  [default]
```

4.8.2 Member Data Documentation

4.8.2.1 x

```
int Coordinate::x = 0
```

4.8.2.2 y

```
int Coordinate::y = 0
```

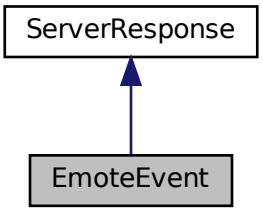
The documentation for this struct was generated from the following file:

- /home/nico/Desktop/battleship/src/common/game_state/Coordinate.h

4.9 EmoteEvent Class Reference

```
#include <EmoteEvent.h>
```

Inheritance diagram for EmoteEvent:



Public Member Functions

- EmoteEvent (EmoteType emote, uuid playerId)

Public Attributes

- const EmoteType emote
- const uuid playerId

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 EmoteEvent()

```
EmoteEvent::EmoteEvent (
            EmoteType emote,
            uuid playerId )
```

4.9.2 Member Data Documentation

4.9.2.1 emote

```
const EmoteType EmoteEvent::emote
```

4.9.2.2 playerId

```
const uuid EmoteEvent::playerId
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/EmoteEvent.h
- /home/nico/Desktop/battleship/src/common/network/responses/EmoteEvent.cpp

4.10 EmoteHandler Class Reference

```
#include <EmoteHandler.h>
```

Static Public Member Functions

- static std::string getSound (EmoteType emote)
- static std::string getImage (EmoteType emote)
- static std::string getImageLarge (EmoteType emote)

4.10.1 Detailed Description

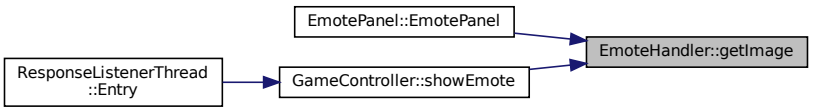
Class to provide images and audio files for all available emotes

4.10.2 Member Function Documentation

4.10.2.1 getImage()

```
std::string EmoteHandler::getImage (
            EmoteType emote )  [static]
```

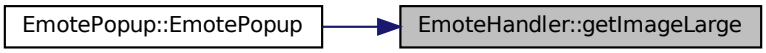
Here is the caller graph for this function:



4.10.2.2 getImageLarge()

```
std::string EmoteHandler::getImageLarge (
            EmoteType emote )  [static]
```

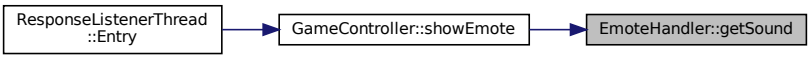
Here is the caller graph for this function:



4.10.2.3 getSound()

```
std::string EmoteHandler::getSound (
            EmoteType emote )  [static]
```

Here is the caller graph for this function:



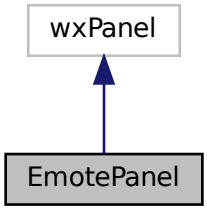
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/EmoteHandler.h
- /home/nico/Desktop/battleship/src/client/EmoteHandler.cpp

4.11 EmotePanel Class Reference

`#include <EmotePanel.h>`

Inheritance diagram for EmotePanel:



Public Member Functions

- EmotePanel (wxWindow *parent, wxPoint pos)

4.11.1 Detailed Description

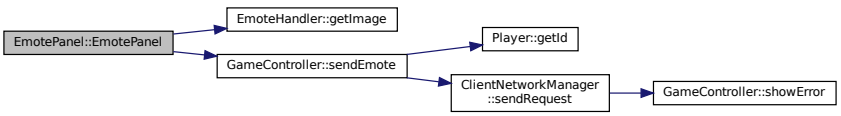
Panel to display the available emotes the user can use to communicate with the opponent

4.11.2 Constructor & Destructor Documentation

4.11.2.1 EmotePanel()

```
EmotePanel::EmotePanel (
            wxWindow * parent,
            wxPoint pos )
```

Here is the call graph for this function:



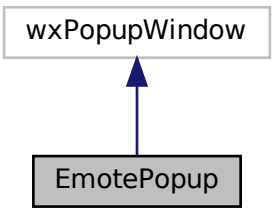
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/EmotePanel.h
- /home/nico/Desktop/battleship/src/client/uiElements/EmotePanel.cpp

4.12 EmotePopup Class Reference

`#include <EmotePopup.h>`

Inheritance diagram for EmotePopup:



Public Member Functions

- EmotePopup (wxWindow *parent, wxPoint pos, EmoteType emote)

4.12.1 Detailed Description

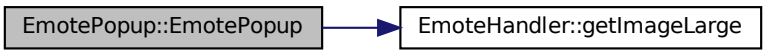
Popup window to display emotes on screen. They disappear when clicking on the image.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 EmotePopup()

```
EmotePopup::EmotePopup (
            wxWindow * parent,
            wxPoint pos,
            EmoteType emote )
```

Here is the call graph for this function:



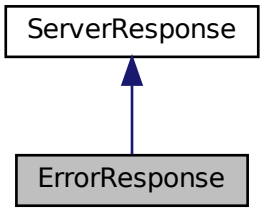
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/EmotePopup.h
- /home/nico/Desktop/battleship/src/client/uiElements/EmotePopup.cpp

4.13 ErrorResponse Class Reference

`#include <ErrorResponse.h>`

Inheritance diagram for ErrorResponse:



Public Member Functions

- ErrorResponse (BattleshipException exception)

Public Attributes

- const BattleshipException exception

Additional Inherited Members

4.13.1 Constructor & Destructor Documentation

4.13.1.1 ErrorResponse()

```
ErrorResponse::ErrorResponse (
            BattleshipException exception )  [explicit]
```

4.13.2 Member Data Documentation

4.13.2.1 exception

```
const BattleshipException ErrorResponse::exception
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/ErrorResponse.h
- /home/nico/Desktop/battleship/src/common/network/responses/ErrorResponse.cpp

4.14 GameController Class Reference

```
#include <GameController.h>
```

Static Public Member Functions

- static void init (GameWindow *gameWindow)

  Initializes all panels, displays the connection panel. Is also used to reset everything on GameOver or QuitGameEvent.
- static void connectToServer ()

  Connects to server. Is called when connect button on ConnectionPanel is clicked.
- static void enterSetupPhase ()

  enter setup phase. Is called when server responds with JoinGameSuccess response. Will show setup panel.
- static void startGame (const StartGameSuccess &response)

  Function that is called when server responds with StartGameSuccess. Will show main game panel.
- static void handleGameEvent (const GameEvent &event)

  Handles an incoming GameEvent. GameEvents are sent by the server when one of the players has placed a shot. The function will update the game state and display the new game state in the main game panel.
- static void callShot (Coordinate position)

  Sends a shot request to the server.
- static void sendEmote (EmoteType emote)

  Sends an emote to the server.
- static void showEmote (const EmoteEvent &emoteEvent)

  Displays an emote to the screen and plays the corresponding sound.
- static void showError (const std::string &title, const std::string &message, bool popup)

  Prints an error message to the console. If popup is true, it will also display a popup with the error message.
- static void gameOver (uuid winnerId)

  Displays dialog box when game is finished and resets the game so both players are back at the connection panel.
- static void handleQuitGameEvent (uuid quitterId)

  Shows popup saying that the other player left, closing the popup brings you back to the connection panel.
- static SetupPanel * getSetupPanel ()
- static wxEvtHandler * getMainThreadEventHandler ()
- static void playerReady ()

  function that is called when ready button in SetupPanel is clicked. Will send request to server to start game and creates GameState used on client side.
- static void quitGame ()

  Sends a QuitGame request to the server, is called upon closing the main window.

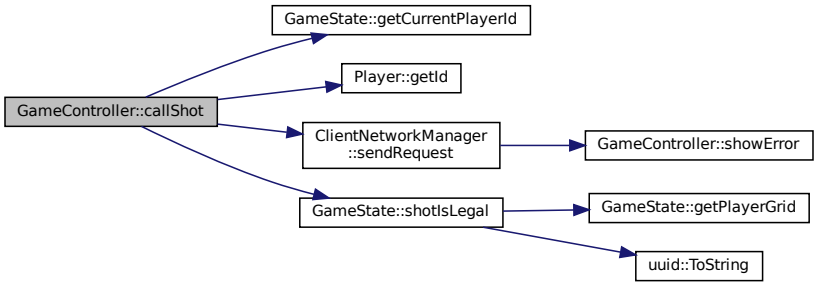
4.14.1 Member Function Documentation

4.14.1.1 callShot()

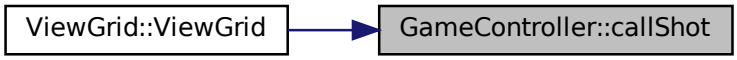
```
void GameController::callShot (
            Coordinate position )  [static]
```

Sends a shot request to the server.

Here is the call graph for this function:



Here is the caller graph for this function:

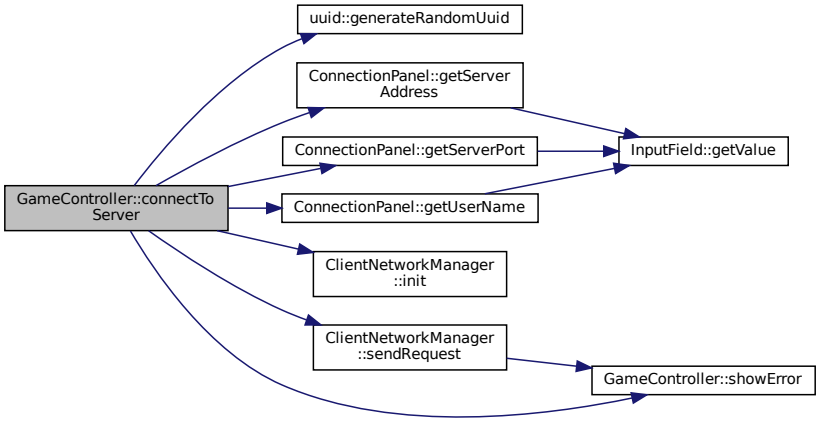


4.14.1.2 connectToServer()

```
void GameController::connectToServer ( )  [static]
```

Connects to server. Is called when connect button on ConnectionPanel is clicked.

Here is the call graph for this function:



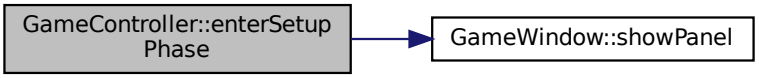
Here is the caller graph for this function:

4.14.1.3 enterSetupPhase()

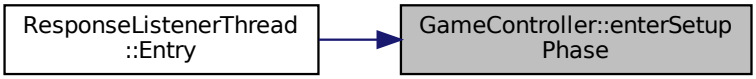
```
void GameController::enterSetupPhase ( )  [static]
```

enter setup phase. Is called when server responds with JoinGameSuccess response. Will show setup panel.

Here is the call graph for this function:



Here is the caller graph for this function:

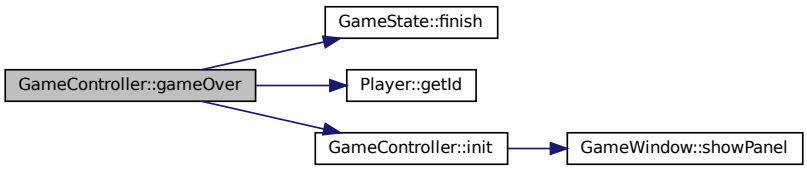


4.14.1.4 gameOver()

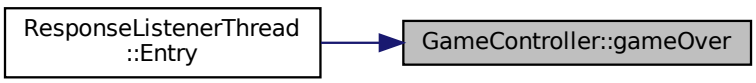
```
void GameController::gameOver (
            uuid winnerId )  [static]
```

Displays dialog box when game is finished and resets the game so both players are back at the connection panel.

Here is the call graph for this function:



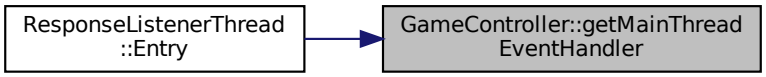
Here is the caller graph for this function:



4.14.1.5 getMainThreadEventHandler()

```
wxEvtHandler * GameController::getMainThreadEventHandler ( )  [static]
```

Here is the caller graph for this function:



4.14.1.6 getSetupPanel()

```
static SetupPanel* GameController::getSetupPanel ( )  [inline], [static]
```

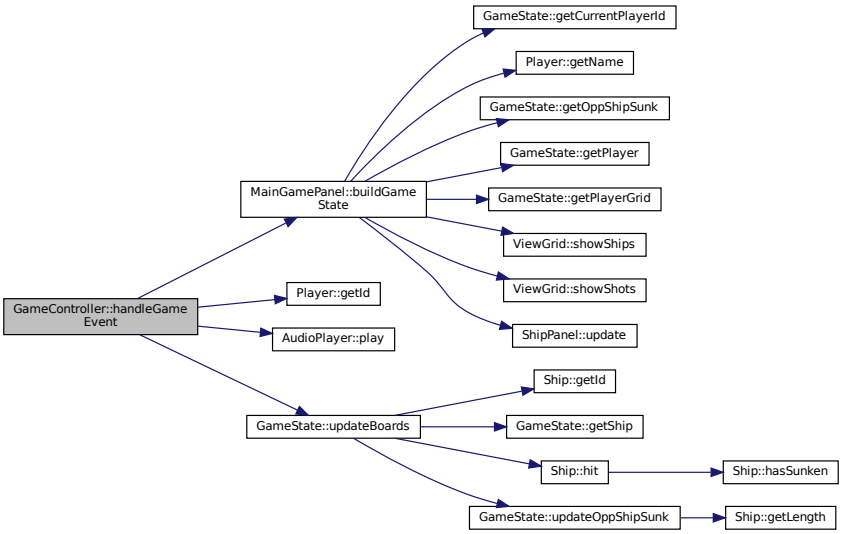
Here is the caller graph for this function:

4.14.1.7 handleGameEvent()

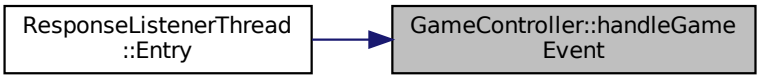
```
void GameController::handleGameEvent (
            const GameEvent & event )  [static]
```

Handles an incoming GameEvent. GameEvents are sent by the server when one of the players has placed a shot. The function will update the game state and display the new game state in the main game panel.

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.1.8 handleQuitGameEvent()

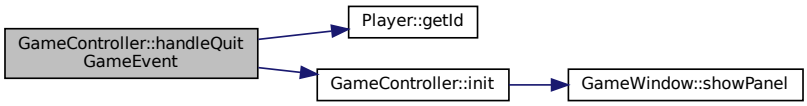
```
void GameController::handleQuitGameEvent (
            uuid quitterId )  [static]
```

Shows popup saying that the other player left, closing the popup brings you back to the connection panel.

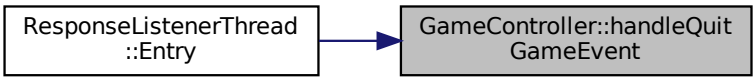
Parameters

leaver↩ Id	id of the player who quit

Here is the call graph for this function:



Here is the caller graph for this function:

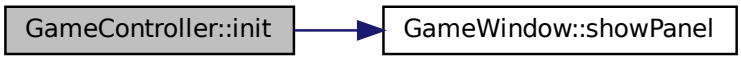


4.14.1.9 init()

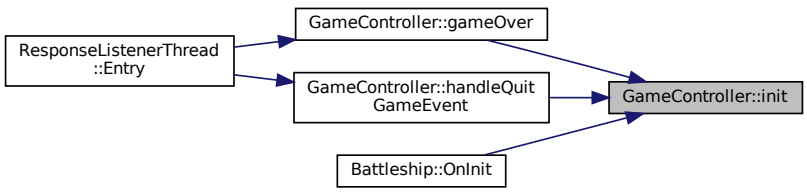
```
void GameController::init (
            GameWindow * gameWindow )  [static]
```

Initializes all panels, displays the connection panel. Is also used to reset everything on GameOver or QuitGameEvent.

Here is the call graph for this function:



Here is the caller graph for this function:

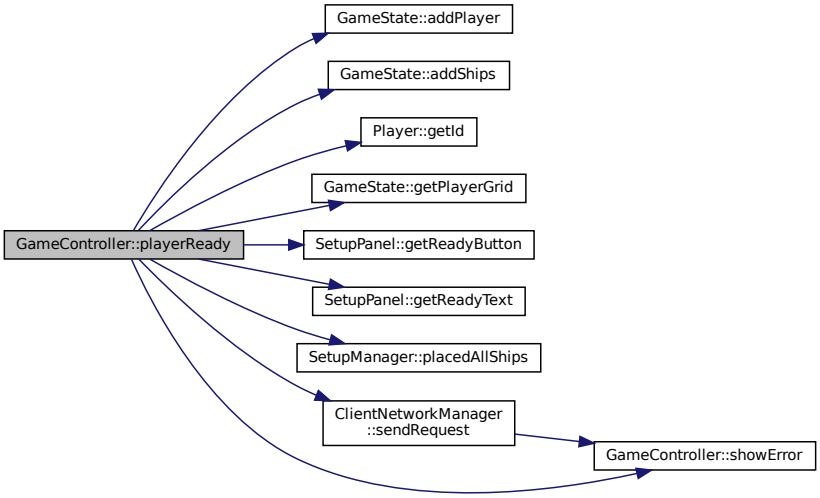


4.14.1.10 playerReady()

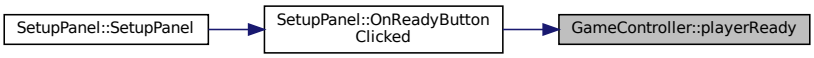
```
void GameController::playerReady ( )  [static]
```

function that is called when ready button in SetupPanel is clicked. Will send request to server to start game and creates GameState used on client side.

Here is the call graph for this function:



Here is the caller graph for this function:

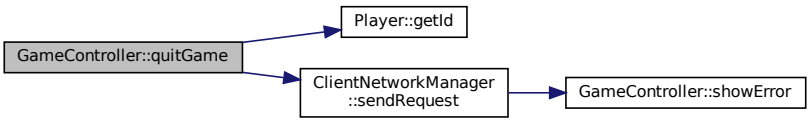


4.14.1.11 quitGame()

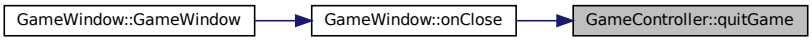
```
void GameController::quitGame ( )  [static]
```

Sends a QuitGame request to the server, is called upon closing the main window.

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.1.12 sendEmote()

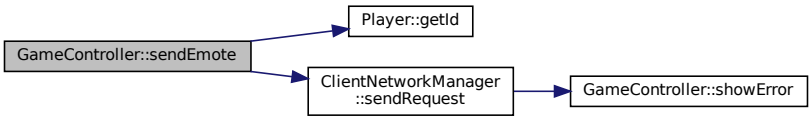
```
void GameController::sendEmote (
            EmoteType emote )  [static]
```

Sends an emote to the server.

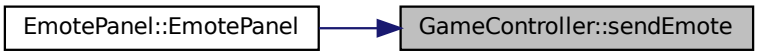
Parameters

emote	The emote to be sent. EmoteType is an enum with 6 possible emotes.

Here is the call graph for this function:



Here is the caller graph for this function:

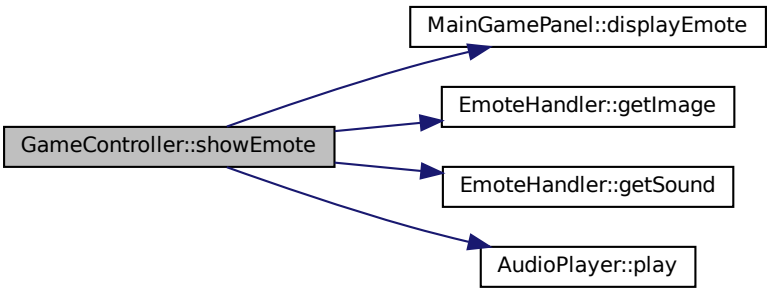


4.14.1.13 showEmote()

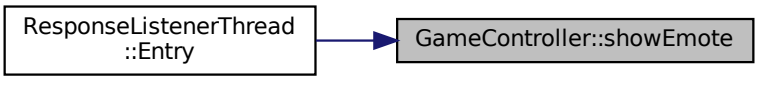
```
void GameController::showEmote (
            const EmoteEvent & emoteEvent )  [static]
```

Displays an emote to the screen and plays the corresponding sound.

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.1.14 showError()

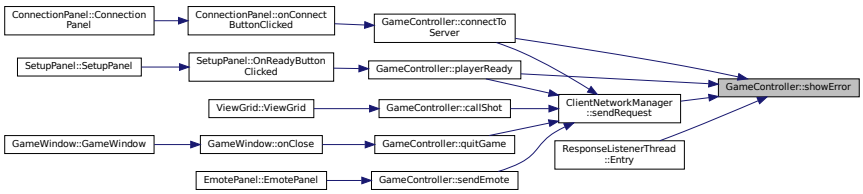
```
void GameController::showError (
            const std::string & title,
            const std::string & message,
            bool popup )  [static]
```

Prints an error message to the console. If popup is true, it will also display a popup with the error message.

Parameters

title	The title of the popup.
message	The message to be displayed.
popup	Whether or not to display a popup window.

Here is the caller graph for this function:

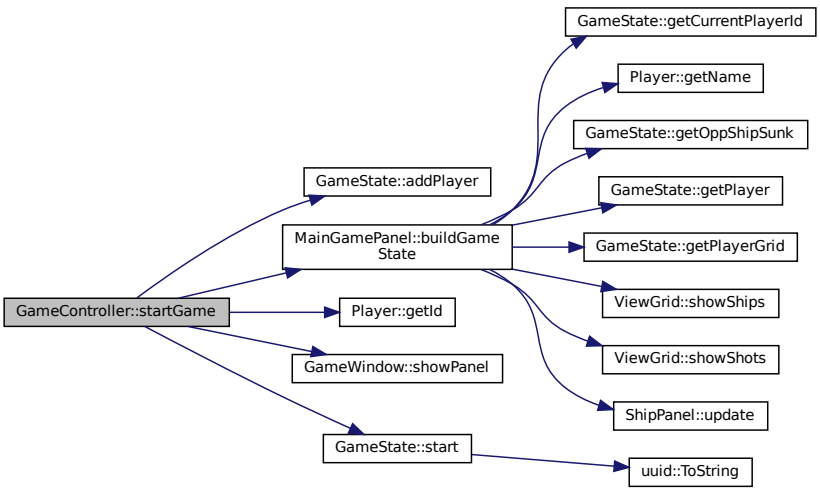


4.14.1.15 startGame()

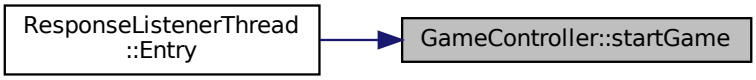
```
void GameController::startGame (
            const StartGameSuccess & response )  [static]
```

Function that is called when server responds with StartGameSuccess. Will show main game panel.

Here is the call graph for this function:



Here is the caller graph for this function:



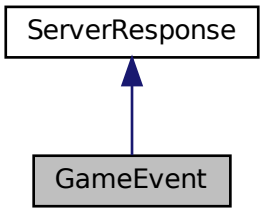
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/GameController.h
- /home/nico/Desktop/battleship/src/client/GameController.cpp

4.15 GameEvent Class Reference

`#include <GameEvent.h>`

Inheritance diagram for GameEvent:



Public Member Functions

- GameEvent (uuid playerId, Coordinate position, bool hit, bool sunk, Ship hitShip, uuid nextPlayerId)

Public Attributes

- const uuid playerId
- const Coordinate position
- const bool hit = false
- const bool sunk = false
- const Ship hitShip
- const uuid nextPlayerId

Additional Inherited Members

4.15.1 Detailed Description

A Game Event is emitted by the server if and only if a new shot was registered

4.15.2 Constructor & Destructor Documentation

4.15.2.1 GameEvent()

```
GameEvent::GameEvent (
            uuid playerId,
            Coordinate position,
            bool hit,
            bool sunk,
            Ship hitShip,
            uuid nextPlayerId )
```

4.15.3 Member Data Documentation

4.15.3.1 hit

```
const bool GameEvent::hit = false
```

4.15.3.2 hitShip

```
const Ship GameEvent::hitShip
```

4.15.3.3 nextPlayerId

```
const uuid GameEvent::nextPlayerId
```

4.15.3.4 playerId

```
const uuid GameEvent::playerId
```

4.15.3.5 position

```
const Coordinate GameEvent::position
```

4.15.3.6 sunk

```
const bool GameEvent::sunk = false
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/GameEvent.h
- /home/nico/Desktop/battleship/src/common/network/responses/GameEvent.cpp

4.16 GameInstance Class Reference

```
#include <GameInstance.h>
```

Public Member Functions

- bool joinGame (const JoinGame &joinGameRequest)
- bool startGame (const Player &player, std::string &err)
- bool executeShot (const CallShot &shotRequest)
- bool reset ()
- GameState & getGameState ()
- bool isReady (const Player &player)

4.16.1 Member Function Documentation

4.16.1.1 executeShot()

```
bool GameInstance::executeShot (
            const CallShot & shotRequest )
```

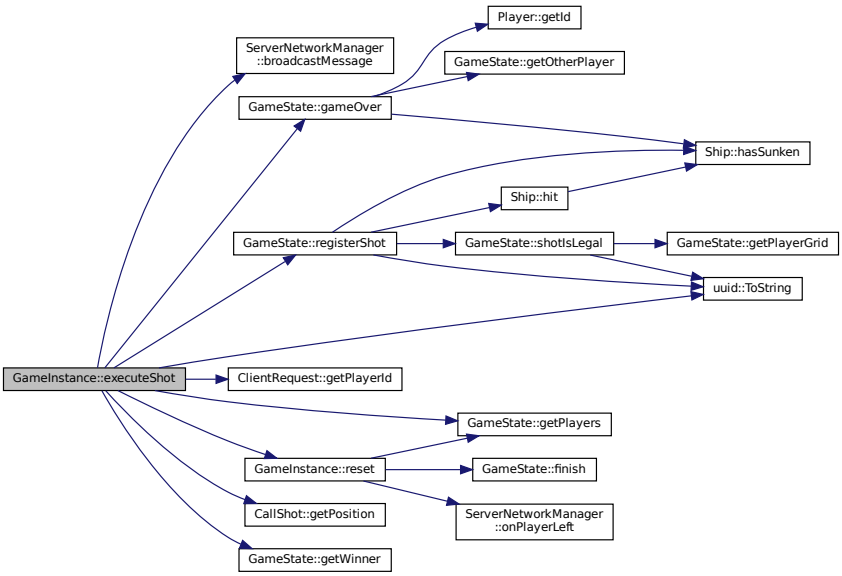
handles a call shot request. registers the shot in the gamestate and sends out a corresponding gameEvent

Parameters

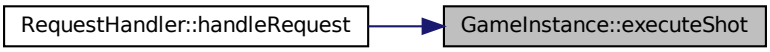
shotRequest	

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.1.2 getGameState()

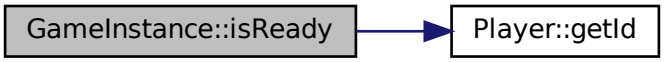
```
GameState & GameInstance::getGameState ( )
```

Here is the caller graph for this function:

4.16.1.3 isReady()

```
bool GameInstance::isReady (
            const Player & player )
```

Here is the call graph for this function:



Here is the caller graph for this function:

4.16.1.4 joinGame()

```
bool GameInstance::joinGame (
            const JoinGame & joinGameRequest )
```

handles a client request to join the game. adds player to the GameState

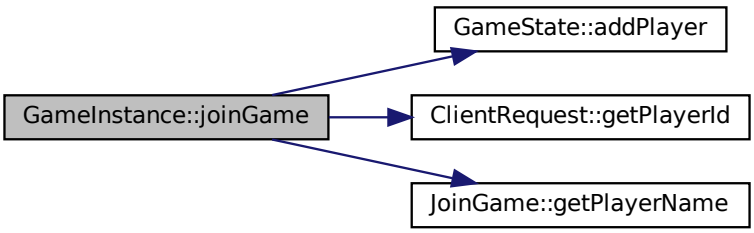
Parameters

joinGameRequest	contains player name and id

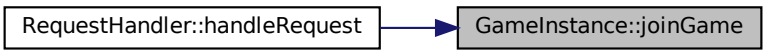
Returns

true if player was added successfully, false otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.1.5 reset()

```
bool GameInstance::reset ( )
```

Function to handle reset after QuitGame or GameOver. Recreates a new GameState to be ready for next Game. Also removes current players network information from the ServerNetworkManager <>

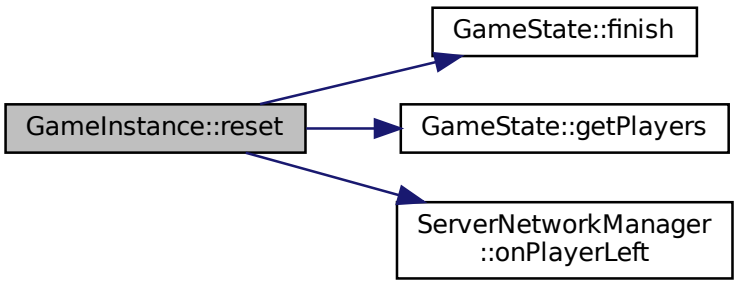
Parameters

QuitGame	

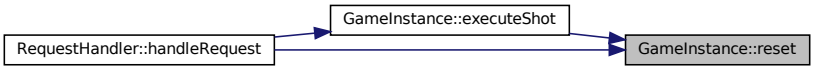
Returns

True if all _gameState attributes are reset. Else, false.

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.1.6 startGame()

```
bool GameInstance::startGame (
            const Player & player,
            std::string & err )
```

starts the game as soon as both players are ready

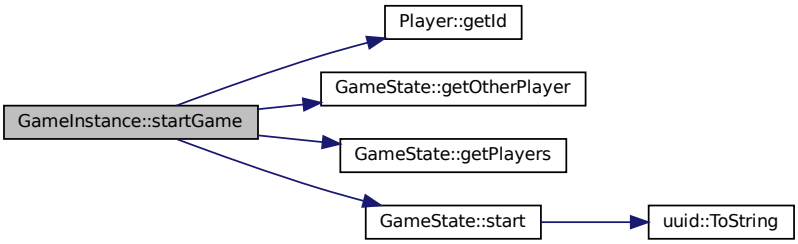
Parameters

player	the player who pressed ready
err	

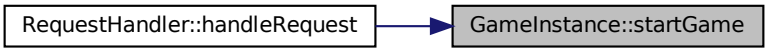
Returns

true if game was started. false if not started

Here is the call graph for this function:



Here is the caller graph for this function:



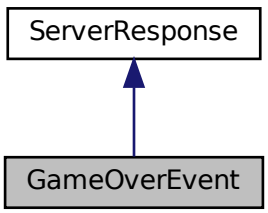
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/server/GameInstance.h
- /home/nico/Desktop/battleship/src/server/GameInstance.cpp

4.17 GameOverEvent Class Reference

`#include <GameOverEvent.h>`

Inheritance diagram for GameOverEvent:



Public Member Functions

- GameOverEvent (uuid winnerPlayerId)

Public Attributes

- const uuid winnerPlayerId

Additional Inherited Members

4.17.1 Constructor & Destructor Documentation

4.17.1.1 GameOverEvent()

```
GameOverEvent::GameOverEvent (
            uuid winnerPlayerId )  [explicit]
```

4.17.2 Member Data Documentation

4.17.2.1 winnerPlayerId

```
const uuid GameOverEvent::winnerPlayerId
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/GameOverEvent.h
- /home/nico/Desktop/battleship/src/common/network/responses/GameOverEvent.cpp

4.18 GameState Class Reference

```
#include <GameState.h>
```

Public Types

- enum class State { Starting , Playing , Finished }
- enum class Type { ServerState , ClientState }

Public Member Functions

- GameState (GameState::Type type)
- bool addPlayer (Player player)
- bool removePlayer (Player player)
- bool addShips (uuid playerId, std::vector< Ship > shipPlacement)
- bool start (uuid currentPlayerId)
- uuid getCurrentPlayerId ()
- const PlayerGrid & getPlayerGrid (uuid playerId) const
- const std::vector< Player > & getPlayers () const
- Ship * getShip (std::vector< Ship > &ships, uuid shipId)
- const Player * getPlayer (uuid playerId) const
- const Player * getOtherPlayer (uuid playerId)
- const State & getState () const
- const std::array< bool, 5 > & getOppShipSunk ()
- bool shotIsLegal (uuid playerId, Coordinate position)
- bool registerShot (uuid playerId, Coordinate position, bool *hit, Ship **hitShipPtr, bool *sunk, uuid *next↩ PlayerId)
- bool updateBoards (const GameEvent &event)

  updates the CLIENT SIDE boards after a game event happened.
- bool updateOppShipSunk (const Ship &hitShip)
- bool gameOver ()
- uuid getWinner ()
- void finish ()

4.18.1 Member Enumeration Documentation

4.18.1.1 State

`enum GameState::State [strong]`

Enum indicating the phase of the game state. Must be switched to Playing before first move is made

Enumerator

Starting	
Playing	
Finished	

4.18.1.2 Type

`enum GameState::Type [strong]`

Enum indicating whether a server side game state is being stored or a client side game state

Enumerator

ServerState	
ClientState	

4.18.2 Constructor & Destructor Documentation

4.18.2.1 GameState()

```
GameState::GameState (
            GameState::Type type )
```

Constructor setting initial values

Parameters

type	defines if this is a server side or client side state

4.18.3 Member Function Documentation

4.18.3.1 addPlayer()

```
auto GameState::addPlayer (
            Player player )
```

Adds a player to the game state

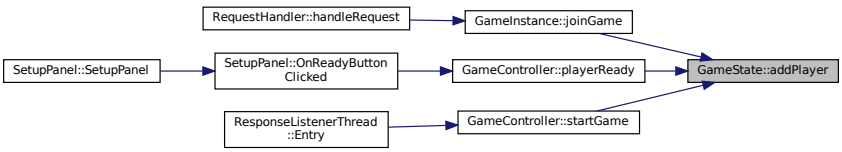
Parameters

player	the player object

Returns

true if player was added successfully, false if a problem occurred.

Here is the caller graph for this function:



4.18.3.2 addShips()

```
auto GameState::addShips (
            uuid playerId,
            std::vector< Ship > shipPlacement )
```

Creates a new playerGrid from a full ship placement

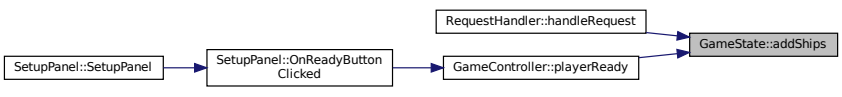
Parameters

playerId	id of the player who placed the ships and owns the board
shipPlacement	a vector containing all ships

Returns

true if grid was created successfully, false if a problem occurred.

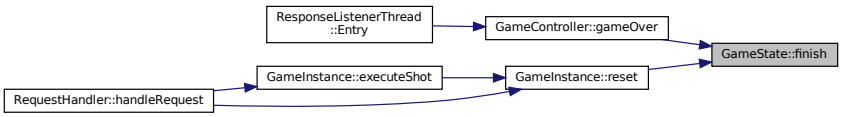
Here is the caller graph for this function:



4.18.3.3 finish()

```
void GameState::finish ( )
```

sets gamestate to Finished. For a next game, a new gamestate should be created. Thus this does not reset everything. Here is the caller graph for this function:



4.18.3.4 gameOver()

```
bool GameState::gameOver ( )
```

Checks if the game is over meaning all ships of one player are sunk. Only to be called by the server.

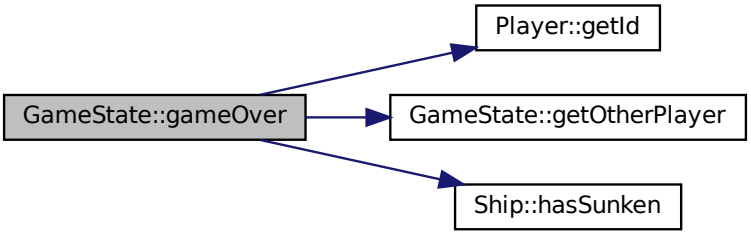
Parameters

winner	id of the winning player. will be "0" if game is not over.

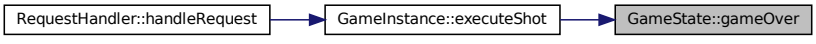
Returns

true if game is over. false if game is not over

Here is the call graph for this function:



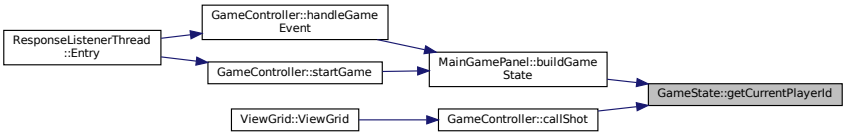
Here is the caller graph for this function:



4.18.3.5 getCurrentPlayerId()

```
uuid GameState::getCurrentPlayerId ( )
```

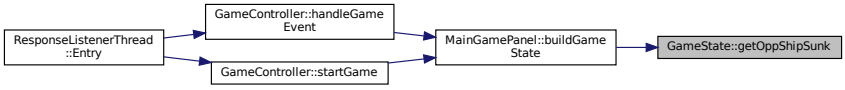
getter for the id of the player whose turn it currently is Here is the caller graph for this function:



4.18.3.6 getOppShipSunk()

```
const std::array< bool, 5 > & GameState::getOppShipSunk ( )
```

returns an array of bool which indicates which opponent ships were already sunk. used for crossing out ships at the bottom of the UI Here is the caller graph for this function:



4.18.3.7 getOtherPlayer()

```
const Player * GameState::getOtherPlayer (
            uuid playerId )
```

returns the id of the player who has NOT the id specified as parameter

Parameters

playerId	

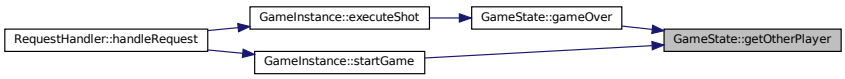
Returns

id of the other player. "Error" if no other player found.

Precondition

there are exactly 2 players added to the gameState

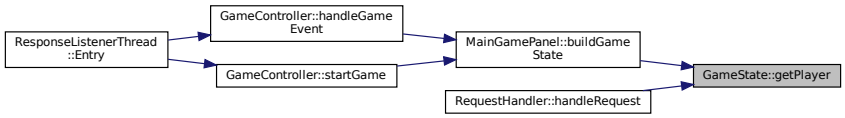
Here is the caller graph for this function:



4.18.3.8 getPlayer()

```
const Player * GameState::getPlayer (
            uuid playerId ) const
```

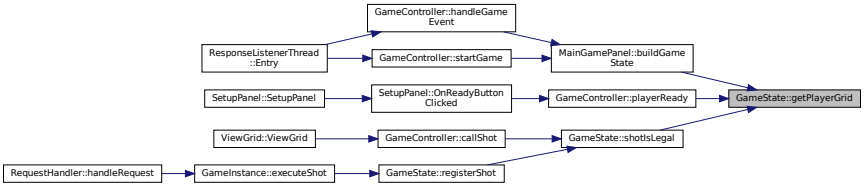
returns the name of the player with the specified id Here is the caller graph for this function:



4.18.3.9 getPlayerGrid()

```
const PlayerGrid & GameState::getPlayerGrid (
            uuid playerId ) const
```

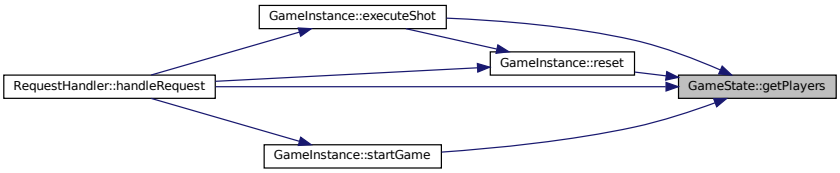
returns a const reference of the grid for e.g. display through UI Here is the caller graph for this function:



4.18.3.10 getPlayers()

```
const std::vector< Player > & GameState::getPlayers ( ) const
```

returns a vector of all the players (2) Here is the caller graph for this function:



4.18.3.11 getShip()

```
Ship * GameState::getShip (
            std::vector< Ship > & ships,
            uuid shipId )
```

returns a reference to the ship with the specified id from a given vector of ships Here is the caller graph for this function:

4.18.3.12 getState()

```
const GameState::State & GameState::getState ( ) const
```

returns the current state of the game (starting, playing, finished) Here is the caller graph for this function:

4.18.3.13 getWinner()

```
uuid GameState::getWinner ( )
```

Returns the winner if the game is over. Should only be called after gameOver() returned true.

Returns

id of winner. null-uuid if game is not over

Here is the caller graph for this function:

4.18.3.14 registerShot()

```
bool GameState::registerShot (
            uuid playerId,
            Coordinate position,
            bool * hit,
            Ship ** hitShipPtr,
            bool * sunk,
            uuid * nextPlayerId )
```

Function to register a shot ON THE SERVER SIDE and process the results

All arguments from ∗hit on are to used to return info back to the caller (in this case to gameInstance) Warning: The double pointer for the ship is intentional. Don't change please!

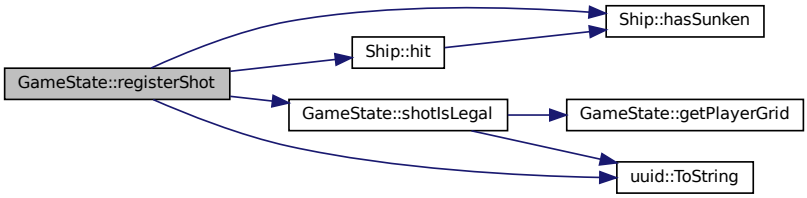
Parameters

playerId	ID of the player who called the shot
position	location of the shot
hit	holds info on whether the shot was a hit
hitShip	holds info on the (potentially) hit ship for the caller to extract (and emit a GameEvent) Turns the pointer to nullptr if no hit is false.
sunk	holds info on whether the hit ship has sunk
next↩ PlayerId	holds id of the player who has to play next

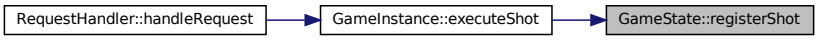
Returns

true if the shot was registered properly, false if a problem occured

Here is the call graph for this function:



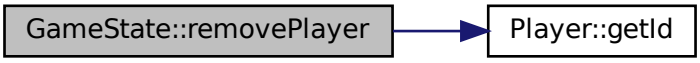
Here is the caller graph for this function:



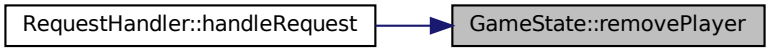
4.18.3.15 removePlayer()

```
bool GameState::removePlayer (
            Player player )
```

Removes a player from the game state if state is still in setup phase. If a player leaves while playing the game ends and a new gameState has to be created. Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.16 shotIsLegal()

```
bool GameState::shotIsLegal (
            uuid playerId,
            Coordinate position )
```

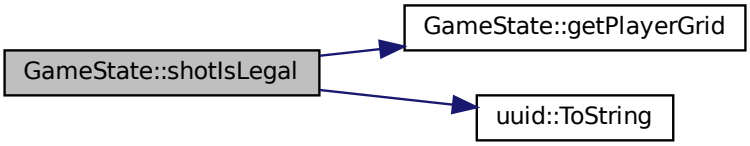
checks if a called shot is a legal move

Parameters

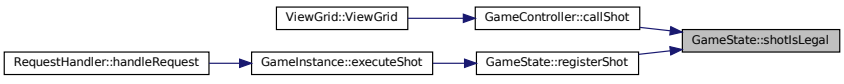
playerId	ID of the player calling the shot
position	location of the shot

Returns

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.17 start()

```
bool GameState::start (
            uuid currentPlayerId )
```

Switches the game state from State::Starting to State::Playing

Precondition

Two players were added to the state. Grids were added (1 for clients, 2 for servers).

Parameters

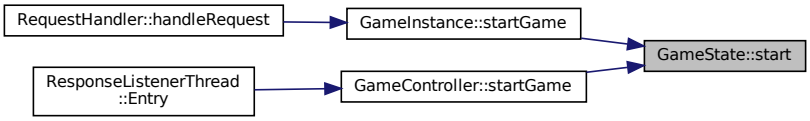
currentPlayerId	id of the player who will go first

Returns

true if the game was started, false if a problem occurred. Check log file for details.

Here is the call graph for this function:

Here is the caller graph for this function:



4.18.3.18 updateBoards()

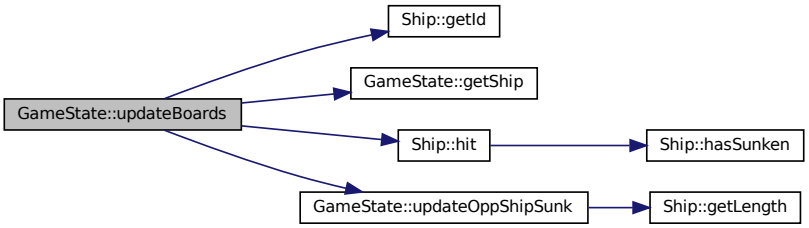
```
bool GameState::updateBoards (
            const GameEvent & event )
```

updates the CLIENT SIDE boards after a game event happened.

Parameters

event	

Here is the call graph for this function:



Here is the caller graph for this function:

4.18.3.19 updateOppShipSunk()

```
bool GameState::updateOppShipSunk (
            const Ship & hitShip )
```

updates the array to cross out ships on the bottom of the screen

Parameters

hitShip	

Returns

Here is the call graph for this function:

Here is the caller graph for this function:

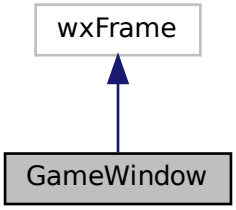
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/game_state/GameState.h
- /home/nico/Desktop/battleship/src/common/game_state/GameState.cpp

4.19 GameWindow Class Reference

`#include <GameWindow.h>`

Inheritance diagram for GameWindow:



Public Member Functions

- GameWindow (const wxString &title, const wxPoint &pos, const wxSize &size)
- void showPanel (wxPanel ∗panel)
- void setStatus (const std::string &message)
- void onClose (wxCloseEvent &event)

4.19.1 Constructor & Destructor Documentation

4.19.1.1 GameWindow()

```
GameWindow::GameWindow (
            const wxString & title,
            const wxPoint & pos,
            const wxSize & size )
```

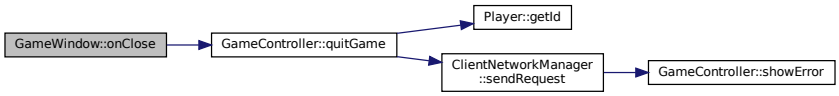
Here is the call graph for this function:

4.19.2 Member Function Documentation

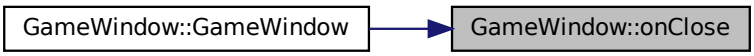
4.19.2.1 onClose()

```
void GameWindow::onClose (
            wxCloseEvent & event )
```

Here is the call graph for this function:



Here is the caller graph for this function:



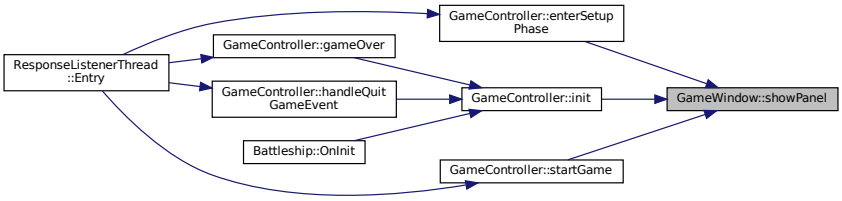
4.19.2.2 setStatus()

```
void GameWindow::setStatus (
            const std::string & message )
```

4.19.2.3 showPanel()

```
void GameWindow::showPanel (
            wxPanel * panel )
```

Here is the caller graph for this function:



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

- /home/nico/Desktop/battleship/src/client/GameWindow.h
- /home/nico/Desktop/battleship/src/client/GameWindow.cpp

4.20 std::hash< uuid > Struct Reference

```
#include <uuid.h>
```

Public Member Functions

- std::size_t operator() (uuid const &id) const noexcept

4.20.1 Member Function Documentation

4.20.1.1 operator()()

```
std::size_t std::hash< uuid >::operator() (
            uuid const & id ) const  [inline], [noexcept]
```

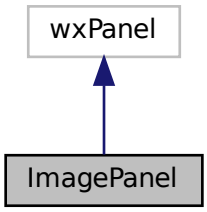
The documentation for this struct was generated from the following file:

- /home/nico/Desktop/battleship/src/common/uuid.h

4.21 ImagePanel Class Reference

`#include <ImagePanel.h>`

Inheritance diagram for ImagePanel:



Public Member Functions

- ImagePanel (wxWindow *parent, wxString file, wxBitmapType format, wxPoint position=wxDefaultPosition, wxSize size=wxDefaultSize, double rotation=0.0)
- void paintEvent (wxPaintEvent &event)
- void onSize (wxSizeEvent &event)

4.21.1 Detailed Description

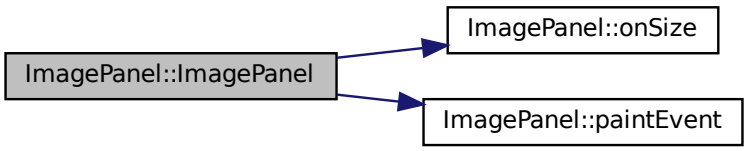
This class can be used to display an image. It can be scaled with parameter <size> and rotated with <rotation> (in radian)

4.21.2 Constructor & Destructor Documentation

4.21.2.1 ImagePanel()

```
ImagePanel::ImagePanel (
            wxWindow * parent,
            wxString file,
            wxBitmapType format,
            wxPoint position = wxDefaultPosition,
            wxSize size = wxDefaultSize,
            double rotation = 0.0 )
```

Here is the call graph for this function:

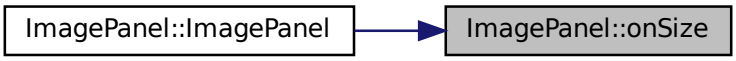


4.21.3 Member Function Documentation

4.21.3.1 onSize()

```
void ImagePanel::onSize (
            wxSizeEvent & event )
```

Here is the caller graph for this function:



4.21.3.2 paintEvent()

```
void ImagePanel::paintEvent (
            wxPaintEvent & event )
```

Here is the caller graph for this function:

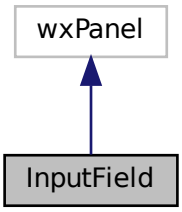
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/ImagePanel.h
- /home/nico/Desktop/battleship/src/client/uiElements/ImagePanel.cpp

4.22 InputField Class Reference

`#include <InputField.h>`

Inheritance diagram for InputField:



Public Member Functions

- InputField (wxWindow *parent, const wxString &labelText, int labelWidth, const wxString &fieldValue, int fieldWidth)
- wxString getValue ()

4.22.1 Constructor & Destructor Documentation

4.22.1.1 InputField()

```
InputField::InputField (
            wxWindow * parent,
            const wxString & labelText,
            int labelWidth,
            const wxString & fieldValue,
            int fieldWidth )
```

4.22.2 Member Function Documentation

4.22.2.1 getValue()

```
wxString InputField::getValue ( )
```

Here is the caller graph for this function:

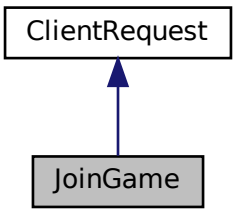
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/InputField.h
- /home/nico/Desktop/battleship/src/client/uiElements/InputField.cpp

4.23 JoinGame Class Reference

```
#include <JoinGame.h>
```

Inheritance diagram for JoinGame:



Public Member Functions

- JoinGame (uuid playerId, std::string playerName)
- auto getPlayerName () const -> std::string

Additional Inherited Members

4.23.1 Constructor & Destructor Documentation

4.23.1.1 JoinGame()

```
JoinGame::JoinGame (
            uuid playerId,
            std::string playerName )
```

4.23.2 Member Function Documentation

4.23.2.1 getPlayerName()

```
auto JoinGame::getPlayerName ( ) const -> std::string
```

Here is the caller graph for this function:

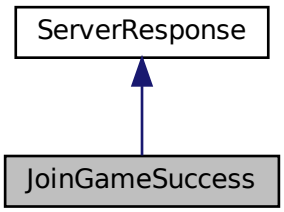
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/network/requests/JoinGame.h
- /home/nico/Desktop/battleship/src/common/network/requests/JoinGame.cpp

4.24 JoinGameSuccess Class Reference

```
#include <JoinGameSuccess.h>
```

Inheritance diagram for JoinGameSuccess:



Public Member Functions

- JoinGameSuccess ()
- bool wasSuccessful () const

Additional Inherited Members

4.24.1 Detailed Description

ServerResponse to a ClientRequest of whether it was successful or not

4.24.2 Constructor & Destructor Documentation

4.24.2.1 JoinGameSuccess()

```
JoinGameSuccess::JoinGameSuccess ( )
```

4.24.3 Member Function Documentation

4.24.3.1 wasSuccessful()

```
bool JoinGameSuccess::wasSuccessful ( ) const
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/JoinGameSuccess.h
- /home/nico/Desktop/battleship/src/common/network/responses/JoinGameSuccess.cpp

4.25 Logger Class Reference

```
#include <Logger.h>
```

Static Public Member Functions

- static void log (const std::string &message, const std::string &function="-")
- static void log (const BattleshipException &exception)
- static void setPrefix (const std::string &s)

4.25.1 Detailed Description

Custom basic logging class. Use the macro LOG(msg) anywhere in the project to log whatever you need.

4.25.2 Member Function Documentation

4.25.2.1 log() [1/2]

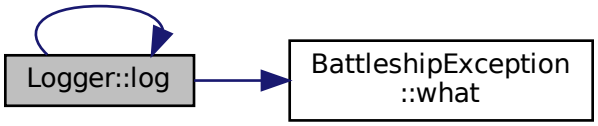
```
void Logger::log (
            const BattleshipException & exception )  [static]
```

overloading of Logger::log to output BattleshipExceptions directly

Parameters

exception	

Here is the call graph for this function:



Here is the caller graph for this function:

4.25.2.2 log() [2/2]

```
void Logger::log (
            const std::string & message,
            const std::string & function = "-" )  [static]
```

outputs a message to std::cout and to the logfile. Use the LOG(msg) macro instead of this function whenever possible

Parameters

message	text to be logged
function	name of the function where this log was called. automatically filled in by the macro

4.25.2.3 setPrefix()

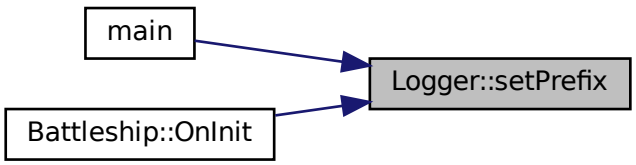
```
void Logger::setPrefix (
            const std::string & s )  [static]
```

changes the prefix used at the very beginning of a log message. use this if you run multiple instances (servers and clients) on the same machine to identify more easily where the message came from

Parameters

s	

Here is the caller graph for this function:



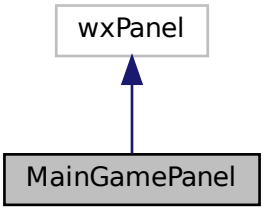
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/Logger.h
- /home/nico/Desktop/battleship/src/common/Logger.cpp

4.26 MainGamePanel Class Reference

`#include <MainGamePanel.h>`

Inheritance diagram for MainGamePanel:



Public Member Functions

- MainGamePanel (wxWindow *parent)

  Constructor for the main game panel. Doesn't display anything, only the background color.
- void buildGameState (GameState *gameState, uuid ownId)

  Builds the game state.
- void displayEmote (EmoteType emote)

  Displays the emote sent by the other player.

4.26.1 Constructor & Destructor Documentation

4.26.1.1 MainGamePanel()

```
MainGamePanel::MainGamePanel (
            wxWindow * parent )
```

Constructor for the main game panel. Doesn't display anything, only the background color.

4.26.2 Member Function Documentation

4.26.2.1 buildGameState()

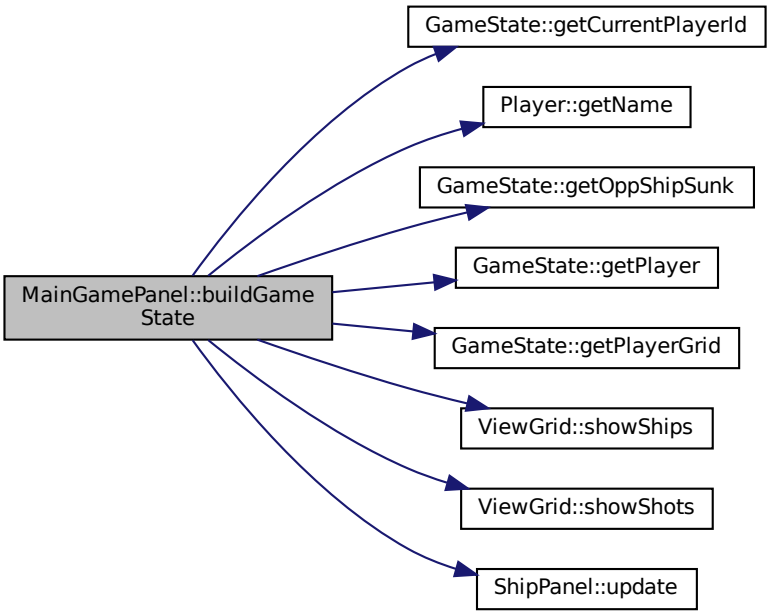
```
void MainGamePanel::buildGameState (
            GameState * gameState,
            uuid ownId )
```

Builds the game state.

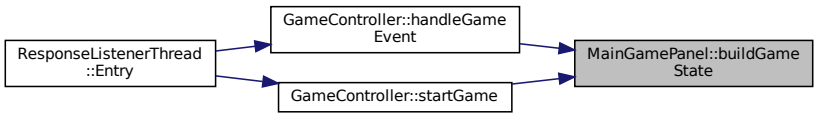
Parameters

gameState	The game state.
ownId	The id of the player who is using this client. Displays the game state using ViewGrid and ShipPanel.

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.2.2 displayEmote()

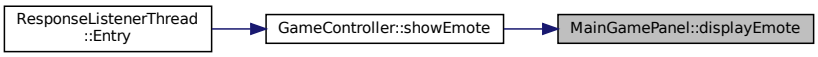
```
void MainGamePanel::displayEmote (
            EmoteType emote )
```

Displays the emote sent by the other player.

Parameters

emote	The emote type that was sent. Not implemented yet.

Here is the caller graph for this function:



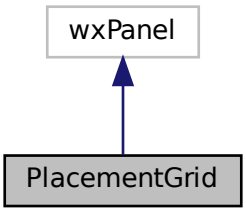
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/panels/MainGamePanel.h
- /home/nico/Desktop/battleship/src/client/panels/MainGamePanel.cpp

4.27 PlacementGrid Class Reference

`#include <PlacementGrid.h>`

Inheritance diagram for PlacementGrid:



Public Member Functions

- PlacementGrid (wxWindow *parent)

  constructor for PlacementGrid. Creates grid of 10x10 tiles and binds mouse events
- void OnMouseMotion (wxMouseEvent &event)

  function is called when mouse hovers over grid and highlights tiles according to mouse position and placed ships
- void OnMouseClick (wxMouseEvent &event)

  function is called when mouse clicks on grid and places ship if possible
- void displayGrid ()

  function displays grid according to data in SetupManager
- void highlightTiles (int CellX, int CellY)

  function highlights tiles according to mouse position and placed ships

Public Attributes

- int cellX_prev = -1
- int cellY_prev = -1

4.27.1 Constructor & Destructor Documentation

4.27.1.1 PlacementGrid()

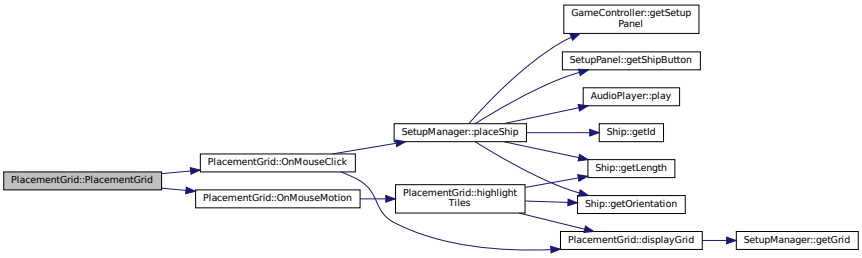
```
PlacementGrid::PlacementGrid (
            wxWindow * parent )
```

constructor for PlacementGrid. Creates grid of 10x10 tiles and binds mouse events

Parameters

parent	

Here is the call graph for this function:



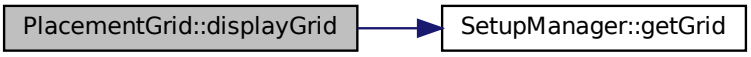
4.27.2 Member Function Documentation

4.27.2.1 displayGrid()

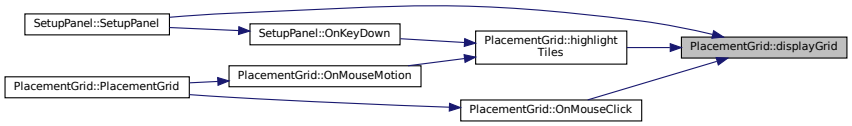
```
void PlacementGrid::displayGrid ( )
```

function displays grid according to data in SetupManager

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.2.2 highlightTiles()

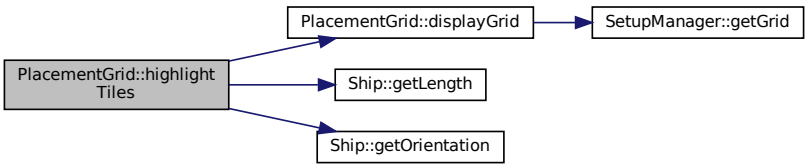
```
void PlacementGrid::highlightTiles (
            int CellX,
            int CellY )
```

function highlights tiles according to mouse position and placed ships

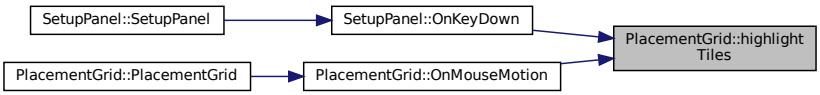
Parameters

CellX	x coordinate of mouse position
CellY	y coordinate of mouse position

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.2.3 OnMouseClick()

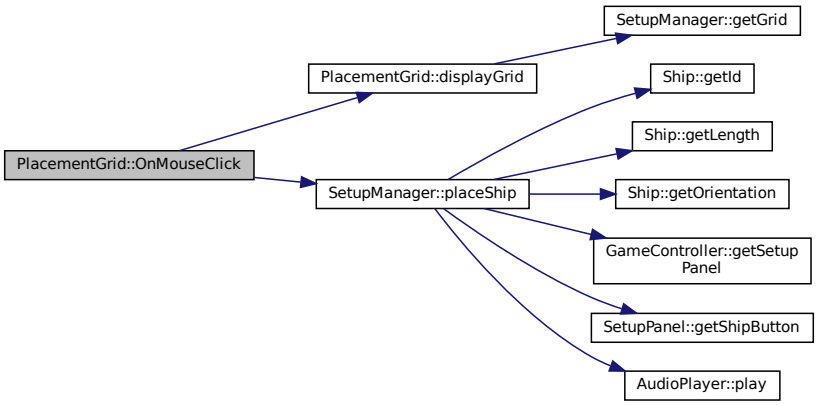
```
void PlacementGrid::OnMouseClick (
            wxMouseEvent & event )
```

function is called when mouse clicks on grid and places ship if possible

Parameters

event	

Here is the call graph for this function:



Here is the caller graph for this function:

4.27.2.4 OnMouseMotion()

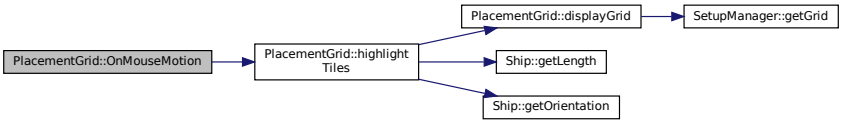
```
void PlacementGrid::OnMouseMotion (
            wxMouseEvent & event )
```

function is called when mouse hovers over grid and highlights tiles according to mouse position and placed ships

Parameters

event	mouse event

Here is the call graph for this function:



Here is the caller graph for this function:

4.27.3 Member Data Documentation

4.27.3.1 cellX_prev

```
int PlacementGrid::cellX_prev = -1
```

4.27.3.2 cellY_prev

```
int PlacementGrid::cellY_prev = -1
```

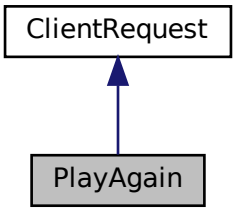
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/PlacementGrid.h
- /home/nico/Desktop/battleship/src/client/uiElements/PlacementGrid.cpp

4.28 PlayAgain Class Reference

`#include <PlayAgain.h>`

Inheritance diagram for PlayAgain:



Public Member Functions

- PlayAgain (uuid playerId)

Additional Inherited Members

4.28.1 Constructor & Destructor Documentation

4.28.1.1 PlayAgain()

```
PlayAgain::PlayAgain (
            uuid playerId )  [explicit]
```

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

- /home/nico/Desktop/battleship/src/common/network/requests/PlayAgain.h
- /home/nico/Desktop/battleship/src/common/network/requests/PlayAgain.cpp

4.29 Player Class Reference

`#include <Player.h>`

Public Member Functions

- Player (uuid playerId, std::string player_name)
- auto getId () const -> uuid
- auto getName () const -> std::string
- bool operator== (const Player &) const =default

4.29.1 Constructor & Destructor Documentation

4.29.1.1 Player()

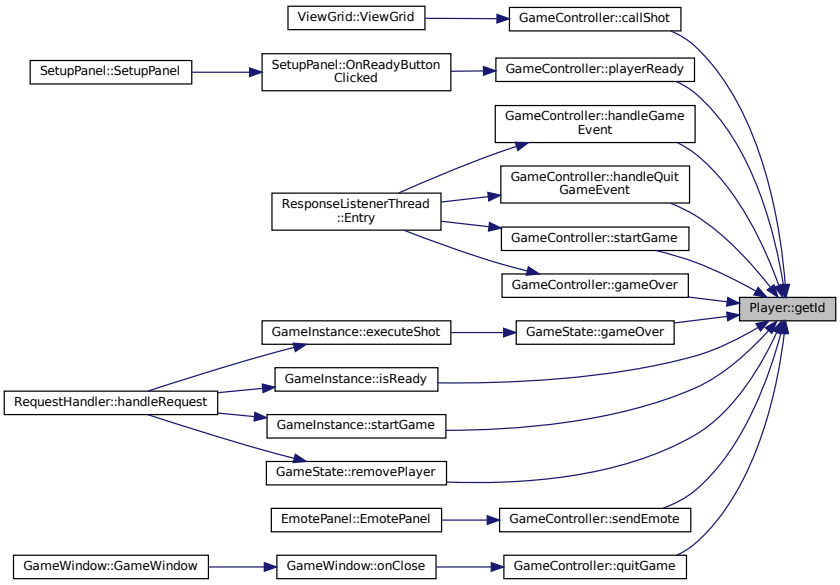
```
Player::Player (
            uuid playerId,
            std::string player_name )
```

4.29.2 Member Function Documentation

4.29.2.1 getId()

```
auto Player::getId ( ) const -> uuid
```

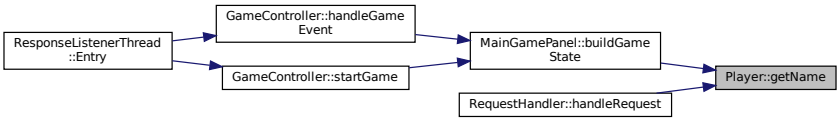
Here is the caller graph for this function:



4.29.2.2 getName()

```
auto Player::getName ( ) const -> std::string
```

Here is the caller graph for this function:



4.29.2.3 operator==()

```
bool Player::operator== (
            const Player &  ) const  [default]
```

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

- /home/nico/Desktop/battleship/src/common/game_state/Player.h
- /home/nico/Desktop/battleship/src/common/game_state/Player.cpp

4.30 PlayerGrid Class Reference

```
#include <PlayerGrid.h>
```

Public Member Functions

- PlayerGrid (uuid playerId, std::vector< Ship > shipsPlacement)

Public Attributes

- uuid playerId

  id of owner of this grid
- std::vector< Ship > shipsPlaced

  vector of all ships placed on this grid
- int shotsReceived [10][10] = {{0}}

  shots received by this grid. 0 = tile not shot, 1 = miss, 2 = hit
- int shotsFired [10][10] = {{0}}

  shots fired by this player. 0 = tile not shot, 1 = miss, 2 = hit

4.30.1 Detailed Description

Data container to store ocean grid info. A grid always belongs to one player (the one who has his ships on this grid)

4.30.2 Constructor & Destructor Documentation

4.30.2.1 PlayerGrid()

```
PlayerGrid::PlayerGrid (
            uuid playerId,
            std::vector< Ship > shipsPlacement )
```

4.30.3 Member Data Documentation

4.30.3.1 playerId

```
uuid PlayerGrid::playerId
```

id of owner of this grid

4.30.3.2 shipsPlaced

```
std::vector<Ship> PlayerGrid::shipsPlaced
```

vector of all ships placed on this grid

4.30.3.3 shotsFired

```
int PlayerGrid::shotsFired[10][10] = {{0}}
```

shots fired by this player. 0 = tile not shot, 1 = miss, 2 = hit

4.30.3.4 shotsReceived

```
int PlayerGrid::shotsReceived[10][10] = {{0}}
```

shots received by this grid. 0 = tile not shot, 1 = miss, 2 = hit

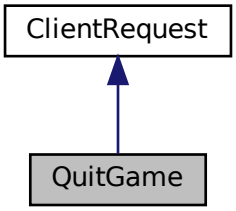
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/game_state/PlayerGrid.h
- /home/nico/Desktop/battleship/src/common/game_state/PlayerGrid.cpp

4.31 QuitGame Class Reference

```
#include <QuitGame.h>
```

Inheritance diagram for QuitGame:



Public Member Functions

- QuitGame (uuid playerId)

Additional Inherited Members

4.31.1 Constructor & Destructor Documentation

4.31.1.1 QuitGame()

```
QuitGame::QuitGame (
            uuid playerId )  [explicit]
```

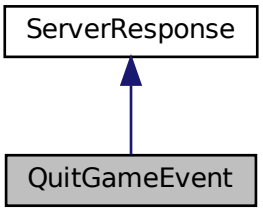
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/network/requests/QuitGame.h
- /home/nico/Desktop/battleship/src/common/network/requests/QuitGame.cpp

4.32 QuitGameEvent Class Reference

`#include <QuitGameEvent.h>`

Inheritance diagram for QuitGameEvent:



Public Member Functions

- QuitGameEvent (uuid quitPlayerId)

Public Attributes

- const uuid quitPlayerId

Additional Inherited Members

4.32.1 Constructor & Destructor Documentation

4.32.1.1 QuitGameEvent()

```
QuitGameEvent::QuitGameEvent (
            uuid quitPlayerId )  [explicit]
```

4.32.2 Member Data Documentation

4.32.2.1 quitPlayerId

```
const uuid QuitGameEvent::quitPlayerId
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/QuitGameEvent.h
- /home/nico/Desktop/battleship/src/common/network/responses/QuitGameEvent.cpp

4.33 RequestHandler Class Reference

```
#include <RequestHandler.h>
```

Static Public Member Functions

- static std::unique_ptr< ServerResponse > handleRequest (GameInstance &gameInstance, const ClientRequest *const req)

4.33.1 Member Function Documentation

4.33.1.1 handleRequest()

```
std::unique_ptr< ServerResponse > RequestHandler::handleRequest (
            GameInstance & gameInstance,
            const ClientRequest *const req )  [static]
```

Handles an incoming request from the ServerNetworkManager and generates a response.

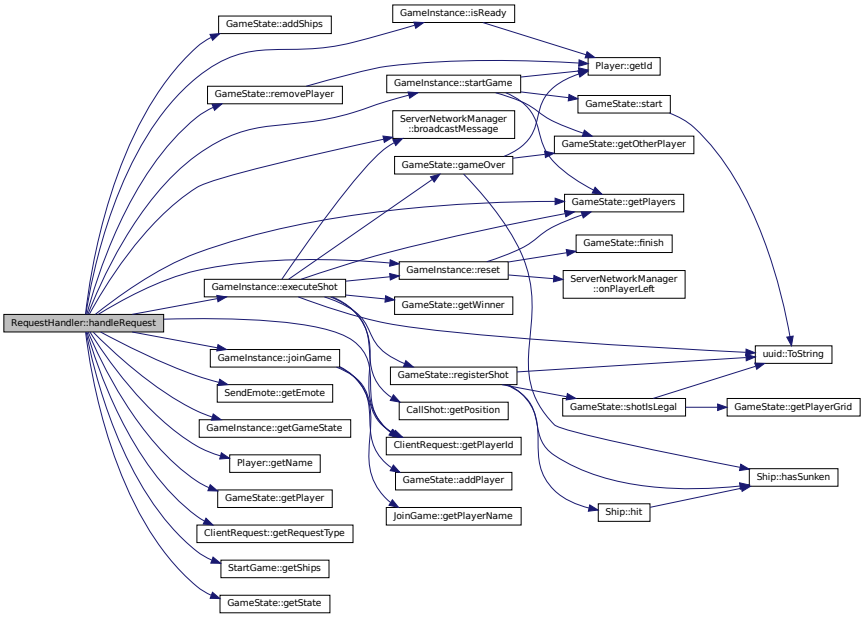
Parameters

gameInstance	
req	

Returns

direct response to send back (only) to the sender of the request

Here is the call graph for this function:



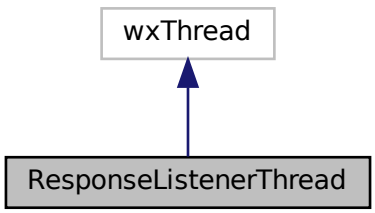
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/server/RequestHandler.h
- /home/nico/Desktop/battleship/src/server/RequestHandler.cpp

4.34 ResponseListenerThread Class Reference

`#include <ResponseListenerThread.h>`

Inheritance diagram for ResponseListenerThread:



Public Member Functions

- ResponseListenerThread (sockpp::tcp_connector ∗connection)

Protected Member Functions

- virtual ExitCode Entry ()

4.34.1 Detailed Description

Thread that is permanently listening for messages from the server

4.34.2 Constructor & Destructor Documentation

4.34.2.1 ResponseListenerThread()

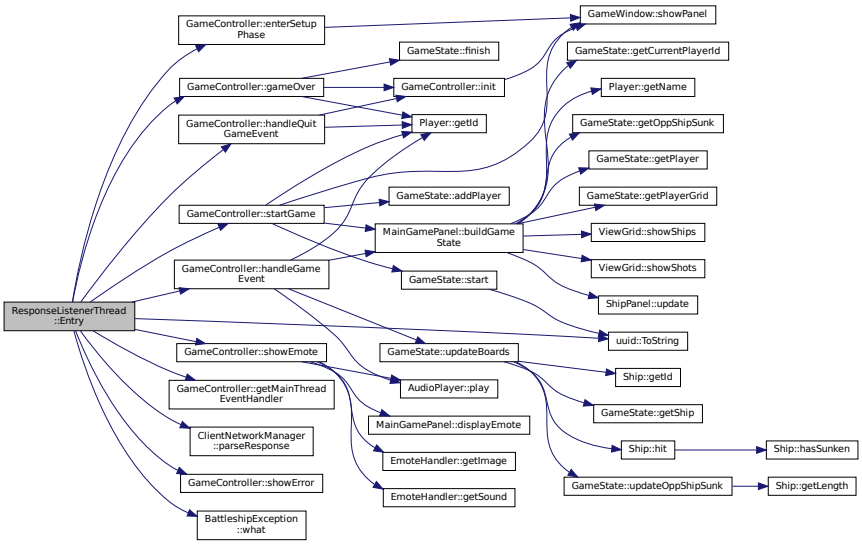
```
ResponseListenerThread::ResponseListenerThread (
            sockpp::tcp_connector * connection )
```

4.34.3 Member Function Documentation

4.34.3.1 Entry()

```
wxThread::ExitCode ResponseListenerThread::Entry ( )  [protected], [virtual]
```

Here is the call graph for this function:



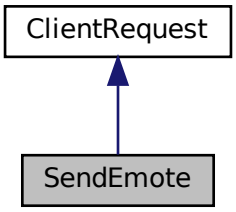
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/ResponseListenerThread.h
- /home/nico/Desktop/battleship/src/client/ResponseListenerThread.cpp

4.35 SendEmote Class Reference

`#include <SendEmote.h>`

Inheritance diagram for SendEmote:



Public Member Functions

- SendEmote (uuid playerId, EmoteType emote)
- auto getEmote () const -> EmoteType

Additional Inherited Members

4.35.1 Constructor & Destructor Documentation

4.35.1.1 SendEmote()

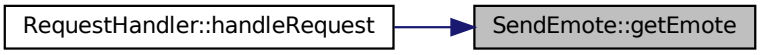
```
SendEmote::SendEmote (
            uuid playerId,
            EmoteType emote )
```

4.35.2 Member Function Documentation

4.35.2.1 getEmote()

```
auto SendEmote::getEmote ( ) const -> EmoteType
```

Here is the caller graph for this function:



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

- /home/nico/Desktop/battleship/src/common/network/requests/SendEmote.h
- /home/nico/Desktop/battleship/src/common/network/requests/SendEmote.cpp

4.36 ServerNetworkManager Class Reference

```
#include <ServerNetworkManager.h>
```

Public Member Functions

- ServerNetworkManager (uint16_t port)
- ~ServerNetworkManager ()
- void listenerLoop ()

Static Public Member Functions

- static void broadcastMessage (ServerResponse &msg, const std::vector< Player > &players, const Player *exclude=nullptr)
- static void onPlayerLeft (uuid player_id)

4.36.1 Constructor & Destructor Documentation

4.36.1.1 ServerNetworkManager()

```
ServerNetworkManager::ServerNetworkManager (
            uint16_t port )
```

4.36.1.2 ~ServerNetworkManager()

```
ServerNetworkManager::~ServerNetworkManager ( )  [default]
```

4.36.2 Member Function Documentation

4.36.2.1 broadcastMessage()

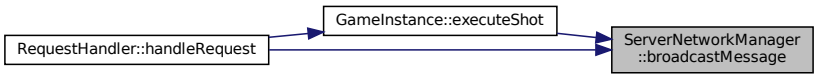
```
void ServerNetworkManager::broadcastMessage (
            ServerResponse & msg,
            const std::vector< Player > & players,
            const Player * exclude = nullptr )  [static]
```

Send out a ServerResponse to everyone

Parameters

msg	Response to be sent
players	Vector of all players/clients to send the message to
exclude	Optional player to exclude from the broadcast

Here is the caller graph for this function:



4.36.2.2 listenerLoop()

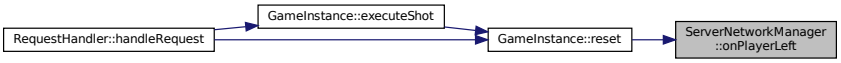
```
void ServerNetworkManager::listenerLoop ( )
```

Here is the caller graph for this function:

4.36.2.3 onPlayerLeft()

```
void ServerNetworkManager::onPlayerLeft (
            uuid player_id )  [static]
```

Here is the caller graph for this function:



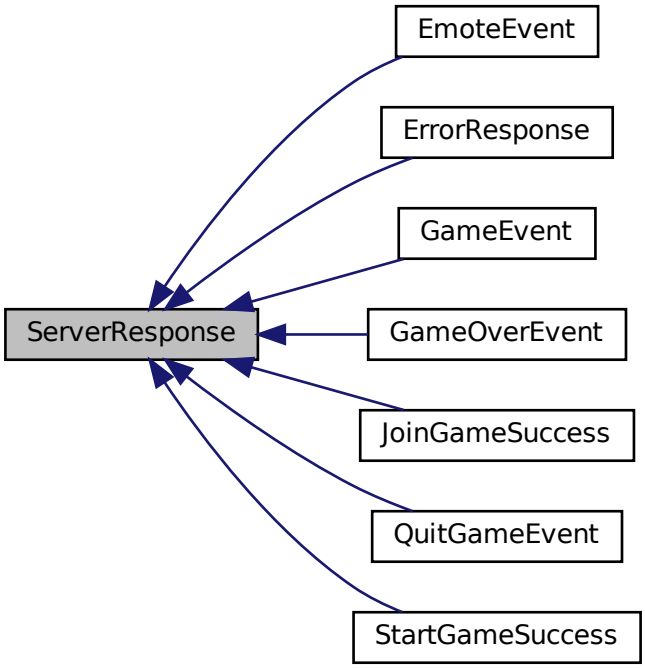
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/server/ServerNetworkManager.h
- /home/nico/Desktop/battleship/src/server/ServerNetworkManager.cpp

4.37 ServerResponse Class Reference

```
#include <ServerResponse.h>
```

Inheritance diagram for ServerResponse:



Public Member Functions

- auto operator<=> (const ServerResponse &) const =default
- virtual ~ServerResponse ()=default

Public Attributes

- const ResponseType responseType

Protected Member Functions

- ServerResponse (ResponseType responseType)

4.37.1 Constructor & Destructor Documentation

4.37.1.1 ~ServerResponse()

```
virtual ServerResponse::~ServerResponse ( )  [virtual], [default]
```

4.37.1.2 ServerResponse()

```
ServerResponse::ServerResponse (
            ResponseType responseType )  [explicit], [protected]
```

4.37.2 Member Function Documentation

4.37.2.1 operator<=>()

```
auto ServerResponse::operator<=> (
            const ServerResponse &  ) const  [default]
```

4.37.3 Member Data Documentation

4.37.3.1 responseType

```
const ResponseType ServerResponse::responseType
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/ServerResponse.h
- /home/nico/Desktop/battleship/src/common/network/responses/ServerResponse.cpp

4.38 SetupManager Class Reference

```
#include <SetupManager.h>
```

Public Member Functions

- SetupManager ()
  constructor for SetupManager. Initializes ships and grid.

Static Public Member Functions

- static bool placeShip (wxPoint &position, Ship *ship)
  function places ship on grid and updates ship position
- static int * getGrid ()
  getter for _grid
- static bool placedAllShips ()
  checks if all ships have been placed (= no longer at initial position)

Static Public Attributes

- static std::vector< Ship > _ships_placed
- static Ship * _selectedShip

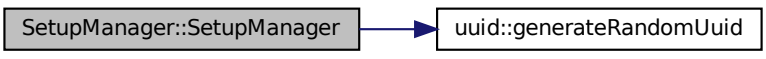
4.38.1 Constructor & Destructor Documentation

4.38.1.1 SetupManager()

```
SetupManager::SetupManager ( )
```

constructor for SetupManager. Initializes ships and grid.

Here is the call graph for this function:



4.38.2 Member Function Documentation

4.38.2.1 getGrid()

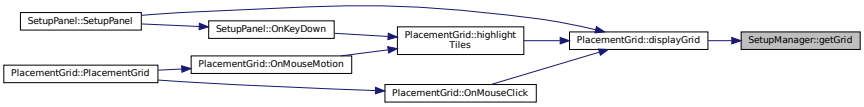
```
int * SetupManager::getGrid ( )  [static]
```

getter for _grid

Returns

int pointer to _grid array

Here is the caller graph for this function:



4.38.2.2 placedAllShips()

```
bool SetupManager::placedAllShips ( )  [static]
```

checks if all ships have been placed (= no longer at initial position)

Returns

true if all ships have been placed, false otherwise

Here is the caller graph for this function:

4.38.2.3 placeShip()

```
bool SetupManager::placeShip (
            wxPoint & position,
            Ship * ship )  [static]
```

function places ship on grid and updates ship position

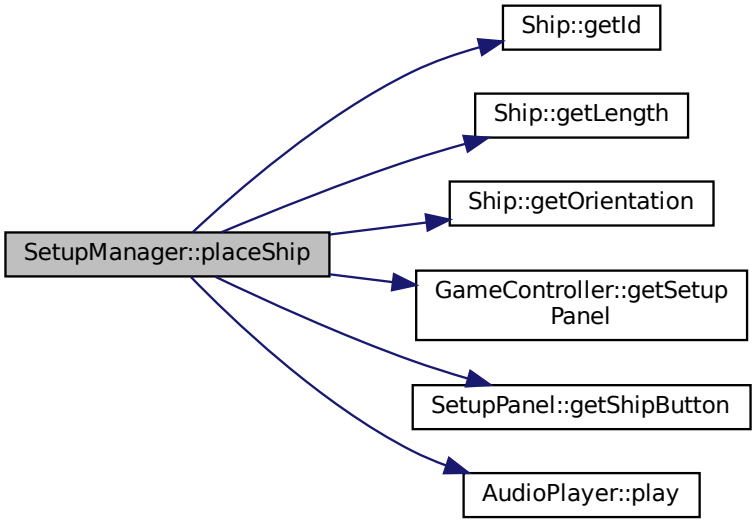
Parameters

position	of left-most or top-most cell of ship (depending on orientation)
ship	pointer to ship that should be placed

Returns

true if ship was placed successfully and false if ship couldn't be placed

Here is the call graph for this function:



Here is the caller graph for this function:

4.38.3 Member Data Documentation

4.38.3.1 _selectedShip

```
Ship * SetupManager::_selectedShip  [static]
```

4.38.3.2 _ships_placed

```
std::vector< Ship > SetupManager::_ships_placed  [static]
```

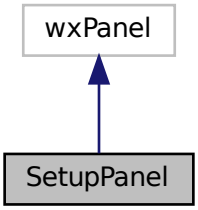
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/SetupManager.h
- /home/nico/Desktop/battleship/src/client/panels/SetupPanel.cpp
- /home/nico/Desktop/battleship/src/client/SetupManager.cpp

4.39 SetupPanel Class Reference

```
#include <SetupPanel.h>
```

Inheritance diagram for SetupPanel:



Public Member Functions

- SetupPanel (wxWindow *parent)

  Constructor of SetupPanel. Creates the panel and all its components.
- void OnReadyButtonClicked (wxCommandEvent &event)

  event handler for when the "Ready" button is clicked. Checks if all ships have been placed, and if so, notifies the GameController that the player is ready.
- void OnKeyDown (wxKeyEvent &event)

  Key Event handler for rotating ship. Rotates selected ship if 'R' is pressed.
- wxStaticBitmap * getShipButton (int idx)

  helper function used in SetupManager::placeShip() to disable ship button after it has been placed
- wxButton * getReadyButton () const

  getter for the ready button (for GameController to disable after it has been clicked)
- wxStaticText * getReadyText () const

4.39.1 Constructor & Destructor Documentation

4.39.1.1 SetupPanel()

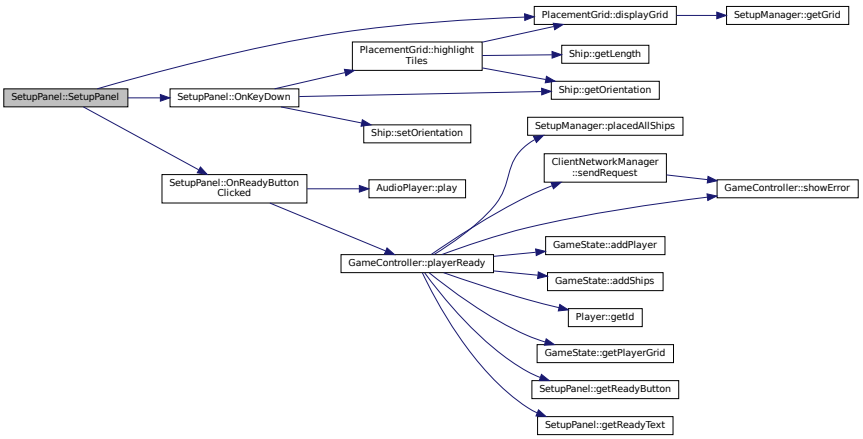
```
SetupPanel::SetupPanel (
            wxWindow * parent )
```

Constructor of SetupPanel. Creates the panel and all its components.

Parameters

parent	

Here is the call graph for this function:



4.39.2 Member Function Documentation

4.39.2.1 getReadyButton()

```
wxButton * SetupPanel::getReadyButton ( ) const
```

getter for the ready button (for GameController to disable after it has been clicked)

Returns

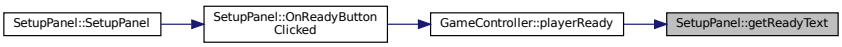
wxButton∗ pointing to the ready button

Here is the caller graph for this function:

4.39.2.2 getReadyText()

```
wxStaticText * SetupPanel::getReadyText ( ) const
```

Here is the caller graph for this function:



4.39.2.3 getShipButton()

```
wxStaticBitmap * SetupPanel::getShipButton (
            int idx )
```

helper function used in SetupManager::placeShip() to disable ship button after it has been placed

Parameters

idx	int in [0, 4] representing the index of the ship button to disable

Returns

wxStaticBitmap* representing the ship button that should be disabled. Returns NULL if idx is invalid.

Here is the caller graph for this function:

4.39.2.4 OnKeyDown()

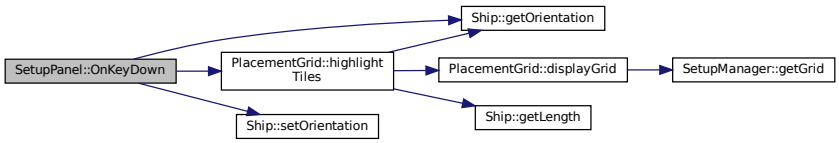
```
void SetupPanel::OnKeyDown (
            wxKeyEvent & event )
```

Key Event handler for rotating ship. Rotates selected ship if 'R' is pressed.

Parameters

event	

Here is the call graph for this function:



Here is the caller graph for this function:

4.39.2.5 OnReadyButtonClicked()

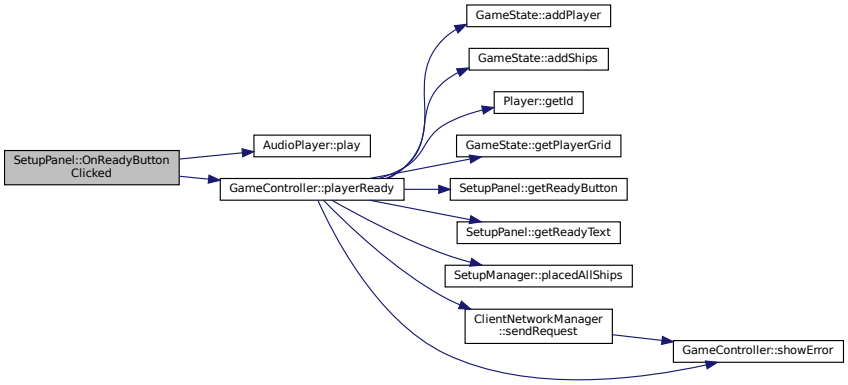
```
void SetupPanel::OnReadyButtonClicked (
            wxCommandEvent & event )
```

event handler for when the "Ready" button is clicked. Checks if all ships have been placed, and if so, notifies the GameController that the player is ready.

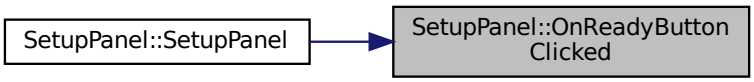
Parameters

event	wxCommandEvent

Here is the call graph for this function:



Here is the caller graph for this function:



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

- /home/nico/Desktop/battleship/src/client/panels/SetupPanel.h
- /home/nico/Desktop/battleship/src/client/panels/SetupPanel.cpp

4.40 Ship Class Reference

`#include <Ship.h>`

Public Types

- enum class Orientation { Vertical , Horizontal }

Public Member Functions

- Ship (int length, Coordinate position, Orientation orientation, uuid id)
- auto getLength () const -> int
- auto getPosition () const -> Coordinate
- auto getOrientation () const -> Orientation
- auto getId () const -> uuid
- auto setOrientation (Orientation orientation) -> void
- auto setPosition (Coordinate position) -> void
- bool hit (Coordinate shot)
- bool hasSunken () const

4.40.1 Member Enumeration Documentation

4.40.1.1 Orientation

```
enum Ship::Orientation  [strong]
```

Enumerator

Vertical	
Horizontal	

4.40.2 Constructor & Destructor Documentation

4.40.2.1 Ship()

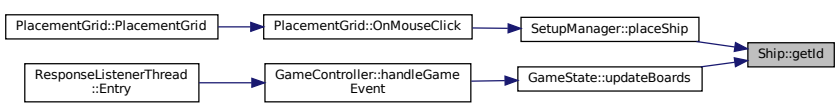
```
Ship::Ship (
            int length,
            Coordinate position,
            Orientation orientation,
            uuid id )
```

4.40.3 Member Function Documentation

4.40.3.1 getId()

```
auto Ship::getId ( ) const -> uuid
```

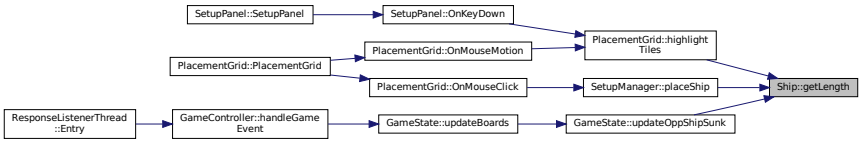
Here is the caller graph for this function:



4.40.3.2 getLength()

```
auto Ship::getLength ( ) const -> int
```

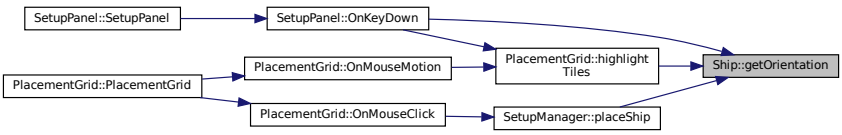
Here is the caller graph for this function:



4.40.3.3 getOrientation()

```
auto Ship::getOrientation ( ) const -> Orientation
```

Here is the caller graph for this function:



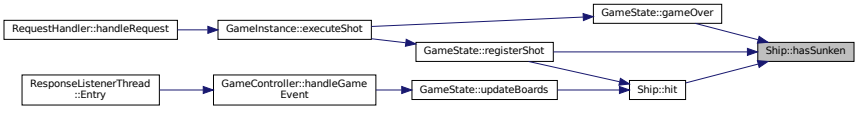
4.40.3.4 getPosition()

```
auto Ship::getPosition ( ) const -> Coordinate
```

4.40.3.5 hasSunken()

```
bool Ship::hasSunken ( ) const
```

Here is the caller graph for this function:



4.40.3.6 hit()

```
bool Ship::hit (
            Coordinate shot )
```

Hit detection. Processes a shot for an individual ship. Will update m_sunk and m_hits

Parameters

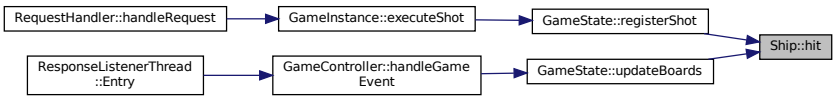
shot	coordinates of the shot called

Returns

true if ship was hit, false otherwise

Here is the call graph for this function:

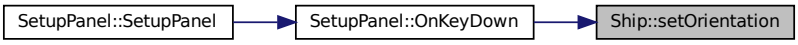
Here is the caller graph for this function:



4.40.3.7 setOrientation()

```
auto Ship::setOrientation (
            Ship::Orientation orientation ) -> void
```

Here is the caller graph for this function:



4.40.3.8 setPosition()

```
auto Ship::setPosition (
            Coordinate position ) -> void
```

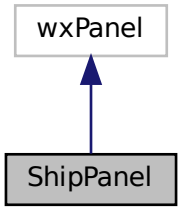
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/game_state/Ship.h
- /home/nico/Desktop/battleship/src/common/game_state/Ship.cpp

4.41 ShipPanel Class Reference

```
#include <ShipPanel.h>
```

Inheritance diagram for ShipPanel:



Public Member Functions

- ShipPanel (wxWindow *parent, wxPoint pos, const std::array< bool, 5 > sunk)
- void update (const std::array< bool, 5 > sunk)

  Updates the panel to show the ships that have been sunk.

4.41.1 Constructor & Destructor Documentation

4.41.1.1 ShipPanel()

```
ShipPanel::ShipPanel (
            wxWindow * parent,
            wxPoint pos,
            const std::array< bool, 5 > sunk )
```

4.41.2 Member Function Documentation

4.41.2.1 update()

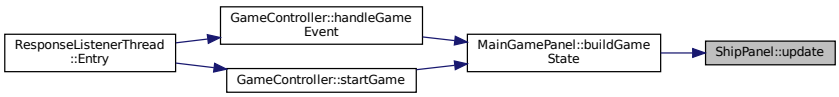
```
void ShipPanel::update (
            const std::array< bool, 5 > sunk )
```

Updates the panel to show the ships that have been sunk.

Parameters

sunk	An array of booleans, where true means that the ship has been sunk. The ships are sorted by length, the ship at index 0 is the longest one with length 5.

Here is the caller graph for this function:



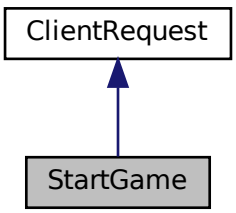
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/client/uiElements/ShipPanel.h
- /home/nico/Desktop/battleship/src/client/uiElements/ShipPanel.cpp

4.42 StartGame Class Reference

```
#include <StartGame.h>
```

Inheritance diagram for StartGame:



Public Member Functions

- StartGame (uuid playerId, std::vector< Ship > ships)
- auto getShips () const -> std::vector< Ship >

Additional Inherited Members

4.42.1 Constructor & Destructor Documentation

4.42.1.1 StartGame()

```
StartGame::StartGame (
            uuid playerId,
            std::vector< Ship > ships )
```

4.42.2 Member Function Documentation

4.42.2.1 getShips()

```
auto StartGame::getShips ( ) const -> std::vector<Ship>
```

Here is the caller graph for this function:

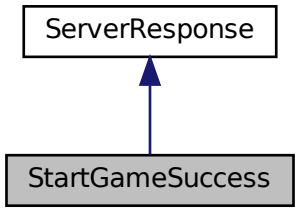
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/network/requests/StartGame.h
- /home/nico/Desktop/battleship/src/common/network/requests/StartGame.cpp

4.43 StartGameSuccess Class Reference

`#include <StartGameSuccess.h>`

Inheritance diagram for StartGameSuccess:



Public Member Functions

- StartGameSuccess (std::vector< Player > players, uuid startingPlayerId)

Public Attributes

- const std::vector< Player > players
- const uuid startingPlayerId

Additional Inherited Members

4.43.1 Constructor & Destructor Documentation

4.43.1.1 StartGameSuccess()

```
StartGameSuccess::StartGameSuccess (
            std::vector< Player > players,
            uuid startingPlayerId )
```

4.43.2 Member Data Documentation

4.43.2.1 players

```
const std::vector<Player> StartGameSuccess::players
```

4.43.2.2 startingPlayerId

```
const uuid StartGameSuccess::startingPlayerId
```

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

- /home/nico/Desktop/battleship/src/common/network/responses/StartGameSuccess.h
- /home/nico/Desktop/battleship/src/common/network/responses/StartGameSuccess.cpp

4.44 uuid Class Reference

```
#include <uuid.h>
```

Public Member Functions

- uuid ()=default
- uuid (const std::string &uuid)
- auto ToString () const -> std::string
- bool operator== (const uuid &) const =default

Static Public Member Functions

- static auto generateRandomUuid () -> uuid

4.44.1 Constructor & Destructor Documentation

4.44.1.1 uuid() [1/2]

```
uuid::uuid ( )  [default]
```

4.44.1.2 uuid() [2/2]

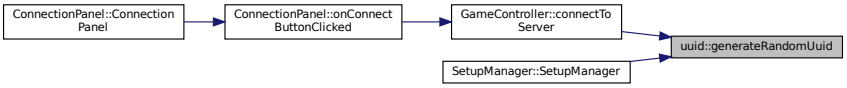
```
uuid::uuid (
            const std::string & uuid )  [explicit]
```

4.44.2 Member Function Documentation

4.44.2.1 generateRandomUuid()

```
auto uuid::generateRandomUuid ( ) -> uuid  [static]
```

Here is the caller graph for this function:



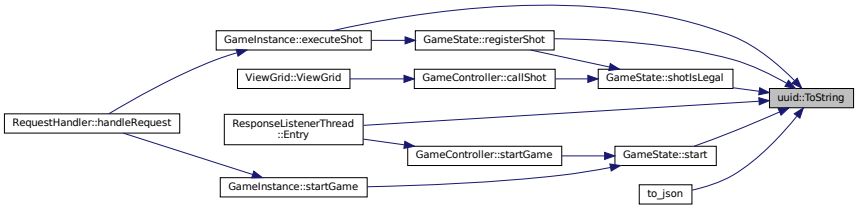
4.44.2.2 operator==()

```
bool uuid::operator== (
            const uuid &  ) const  [default]
```

4.44.2.3 ToString()

```
auto uuid::ToString ( ) const -> std::string
```

Here is the caller graph for this function:



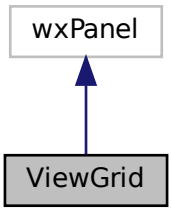
The documentation for this class was generated from the following files:

- /home/nico/Desktop/battleship/src/common/uuid.h
- /home/nico/Desktop/battleship/src/common/uuid.cpp

4.45 ViewGrid Class Reference

`#include <ViewGrid.h>`

Inheritance diagram for ViewGrid:



Public Types

- enum GridType { own , opp }

Public Member Functions

- ViewGrid (wxWindow *parent, GridType type)
- void showShips (const std::vector< Ship > &ships)

  Displays the ships on the grid.
- void showShots (const int shots[10][10])

  Displays the shots on the grid.

4.45.1 Member Enumeration Documentation

4.45.1.1 GridType

`enum ViewGrid::GridType`

Enumerator

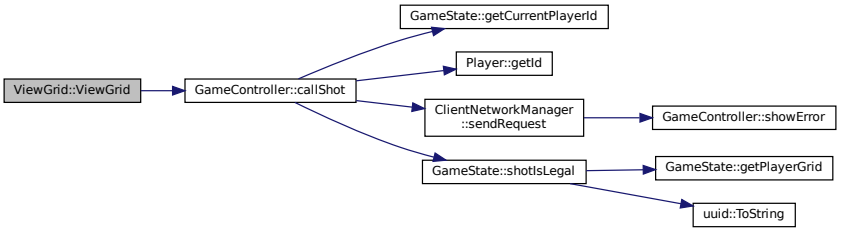
own	
opp	

4.45.2 Constructor & Destructor Documentation

4.45.2.1 ViewGrid()

```
ViewGrid::ViewGrid (
            wxWindow * parent,
            ViewGrid::GridType type )
```

Here is the call graph for this function:



4.45.3 Member Function Documentation

4.45.3.1 showShips()

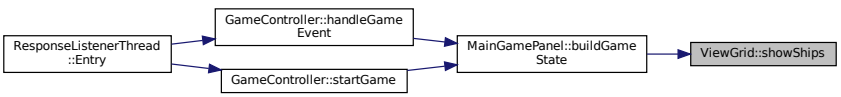
```
void ViewGrid::showShips (
            const std::vector< Ship > & ships )
```

Displays the ships on the grid.

Parameters

ships	A vector of ships that will be displayed. Displays the ships on the grid using the ship's position and orientation. This function should only be called for the own grid.

Here is the caller graph for this function:



4.45.3.2 showShots()

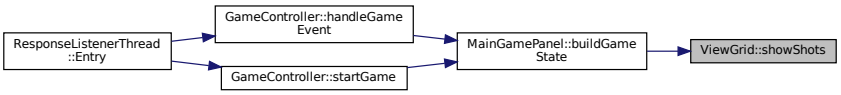
```
void ViewGrid::showShots (
            const int shots[10][10] )
```

Displays the shots on the grid.

Parameters

shots	A 2D array of integers, where 0 means no shot, 1 means a miss and 2 means a hit. A miss is marked with a white dot and a hit is marked with a red dot.

Here is the caller graph for this function:



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

- /home/nico/Desktop/battleship/src/client/uiElements/ViewGrid.h
- /home/nico/Desktop/battleship/src/client/uiElements/ViewGrid.cpp

Chapter 5

File Documentation

5.1 /home/nico/Desktop/battleship/src/client/AudioPlayer.cpp File Reference

```
#include "AudioPlayer.h"
#include "Logger.h"
```

5.2 /home/nico/Desktop/battleship/src/client/AudioPlayer.h File Reference

```
#include "EmoteHandler.h"
#include "wx/sound.h"
#include <map>
#include <string>
#include <wx/wx.h>
```

Classes

- class AudioPlayer

5.3 /home/nico/Desktop/battleship/src/client/Battleship.cpp File Reference

```
#include "Battleship.h"
#include "GameController.h"
#include "GameWindow.h"
#include "Logger.h"
```

5.4 /home/nico/Desktop/battleship/src/client/Battleship.h File Reference

```
#include <wx/wx.h>
```

Classes

- class Battleship

5.5 /home/nico/Desktop/battleship/src/client/ClientNetworkManager.cpp File Reference

```
#include "ClientNetworkManager.h"
#include "GameController.h"
#include "Logger.h"
#include "ResponseListenerThread.h"
#include "serialization/serialization.h"
#include <nlohmann/json.hpp>
#include <sockpp/exception.h>
#include <sockpp/tcp_connector.h>
#include <sstream>
```

5.6 /home/nico/Desktop/battleship/src/client/ClientNetworkManager.h File Reference

```
#include "ResponseListenerThread.h"
#include "network/requests/ClientRequest.h"
#include "network/responses/ServerResponse.h"
#include "sockpp/tcp_connector.h"
#include <memory>
#include <string>
```

Classes

- class ClientNetworkManager

5.7 /home/nico/Desktop/battleship/src/client/EmoteHandler.cpp File Reference

```
#include "EmoteHandler.h"
```

5.8 /home/nico/Desktop/battleship/src/client/EmoteHandler.h File Reference

```
#include "network/responses/EmoteEvent.h"
#include <map>
#include <string>
```

Classes

- class EmoteHandler

5.9 /home/nico/Desktop/battleship/src/client/GameController.cpp File Reference

```
#include "GameController.h"
#include "AudioPlayer.h"
#include "ClientNetworkManager.h"
#include "Logger.h"
#include "network/requests/CallShot.h"
#include "network/requests/JoinGame.h"
#include "network/requests/QuitGame.h"
#include "network/requests/SendEmote.h"
#include "network/requests/StartGame.h"
```

5.10 /home/nico/Desktop/battleship/src/client/GameController.h File Reference

```
#include "EmoteHandler.h"
#include "GameWindow.h"
#include "SetupManager.h"
#include "game_state/Player.h"
#include "network/responses/EmoteEvent.h"
#include "network/responses/GameEvent.h"
#include "network/responses/QuitGameEvent.h"
#include "network/responses/StartGameSuccess.h"
#include "panels/ConnectionPanel.h"
#include "panels/MainGamePanel.h"
#include "panels/SetupPanel.h"
#include <chrono>
#include <wx/wx.h>
```

Classes

- class GameController

5.11 /home/nico/Desktop/battleship/src/client/GameWindow.cpp File Reference

```
#include "GameWindow.h"
#include "../common/Logger.h"
#include "GameController.h"
#include <wx/wx.h>
```

5.12 /home/nico/Desktop/battleship/src/client/GameWindow.h File Reference

```
#include <wx/wx.h>
```

Classes

- class GameWindow

5.13 /home/nico/Desktop/battleship/src/client/main.cpp File Reference

```
#include "Battleship.h"
#include <wx/wx.h>
```

Functions

- wxIMPLEMENT_APP (Battleship)

5.13.1 Function Documentation

5.13.1.1 wxIMPLEMENT_APP()

```
wxIMPLEMENT_APP (
            Battleship  )
```

5.14 /home/nico/Desktop/battleship/src/server/main.cpp File Reference

```
#include "Logger.h"
#include "ServerNetworkManager.h"
```

Functions

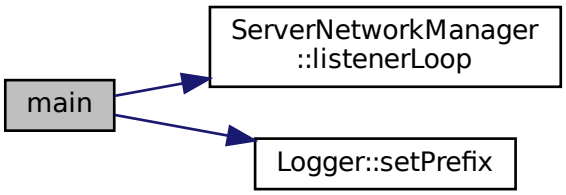
- int main ()

5.14.1 Function Documentation

5.14.1.1 main()

```
int main ( )
```

Here is the call graph for this function:



5.15 /home/nico/Desktop/battleship/src/client/panels/ConnectionPanel.cpp File Reference

```
#include "ConnectionPanel.h"
#include "../AudioPlayer.h"
#include "../GameController.h"
#include "../uiElements/ImagePanel.h"
```

5.16 /home/nico/Desktop/battleship/src/client/panels/ConnectionPanel.h File Reference

```
#include "../uiElements/InputField.h"
#include <wx/wx.h>
```

Classes

- class ConnectionPanel

5.17 /home/nico/Desktop/battleship/src/client/panels/MainGame↩ Panel.cpp File Reference

```
#include "MainGamePanel.h"
#include "../GameController.h"
#include "Logger.h"
```

5.18 /home/nico/Desktop/battleship/src/client/panels/MainGamePanel.h File Reference

```
#include "../../common/game_state/Player.h"
#include "../../common/network/responses/EmoteEvent.h"
#include "../uiElements/EmotePanel.h"
#include "../uiElements/EmotePopup.h"
#include "../uiElements/ShipPanel.h"
#include "../uiElements/ViewGrid.h"
#include "game_state/GameState.h"
#include <wx/wx.h>
```

Classes

- class MainGamePanel

5.19 /home/nico/Desktop/battleship/src/client/panels/SetupPanel.cpp File Reference

```
#include "SetupPanel.h"
#include "../AudioPlayer.h"
#include "../GameController.h"
#include "Logger.h"
```

5.20 /home/nico/Desktop/battleship/src/client/panels/SetupPanel.h File Reference

```
#include "../SetupManager.h"
#include "../uiElements/PlacementGrid.h"
#include <wx/wx.h>
```

Classes

- class SetupPanel

5.21 /home/nico/Desktop/battleship/src/client/ResponseListenerThread.cpp File Reference

```
#include "ResponseListenerThread.h"
#include "ClientNetworkManager.h"
#include "GameController.h"
#include "Logger.h"
#include "network/responses/ErrorResponse.h"
#include "network/responses/GameOverEvent.h"
#include <sstream>
```

5.22 /home/nico/Desktop/battleship/src/client/ResponseListenerThread.h File Reference

```
#include <sockpp/tcp_connector.h>
#include <wx/wx.h>
```

Classes

- class ResponseListenerThread

5.23 /home/nico/Desktop/battleship/src/client/SetupManager.cpp File Reference

```
#include "SetupManager.h"
#include "AudioPlayer.h"
#include "GameController.h"
#include "Logger.h"
#include "game_state/Coordinate.h"
```

5.24 /home/nico/Desktop/battleship/src/client/SetupManager.h File Reference

```
#include "../common/game_state/Ship.h"
#include <vector>
#include <wx/gdicmn.h>
```

Classes

- class SetupManager

5.25 /home/nico/Desktop/battleship/src/client/uiElements/EmotePanel.cpp File Reference

```
#include "EmotePanel.h"
#include "../GameController.h"
#include "Logger.h"
```

5.26 /home/nico/Desktop/battleship/src/client/uiElements/EmotePanel.h File Reference

```
#include "../../common/network/responses/EmoteEvent.h"
#include "../EmoteHandler.h"
#include <chrono>
#include <wx/wx.h>
```

Classes

- class EmotePanel

5.27 /home/nico/Desktop/battleship/src/client/uiElements/EmotePopup.cpp File Reference

```
#include "EmotePopup.h"
#include "Logger.h"
```

5.28 /home/nico/Desktop/battleship/src/client/uiElements/EmotePopup.h File Reference

```
#include "../EmoteHandler.h"
#include <wx/popupwin.h>
#include <wx/wx.h>
```

Classes

- class EmotePopup

5.29 /home/nico/Desktop/battleship/src/client/uiElements/ImagePanel.cpp File Reference

```
#include "ImagePanel.h"
```

5.30 /home/nico/Desktop/battleship/src/client/uiElements/ImagePanel.h File Reference

```
#include <wx/sizer.h>
#include <wx/wx.h>
```

Classes

- class ImagePanel

5.31 /home/nico/Desktop/battleship/src/client/uiElements/InputField.cpp File Reference

```
#include "InputField.h"
```

5.32 /home/nico/Desktop/battleship/src/client/uiElements/InputField.h File Reference

```
#include <wx/wx.h>
```

Classes

- class InputField

5.33 /home/nico/Desktop/battleship/src/client/uiElements/PlacementGrid.cpp File Reference

```
#include "PlacementGrid.h"
#include "../SetupManager.h"
#include "Logger.h"
#include <wx/wx.h>
```

5.34 /home/nico/Desktop/battleship/src/client/uiElements/PlacementGrid.h File Reference

```
#include <wx/wx.h>
```

Classes

- class PlacementGrid

5.35 /home/nico/Desktop/battleship/src/client/uiElements/ShipPanel.cpp File Reference

```
#include "ShipPanel.h"
```

5.36 /home/nico/Desktop/battleship/src/client/uiElements/ShipPanel.h File Reference

```
#include "game_state/Ship.h"
#include <string>
#include <wx/wx.h>
```

Classes

- class ShipPanel

5.37 /home/nico/Desktop/battleship/src/client/uiElements/ViewGrid.cpp File Reference

```
#include "ViewGrid.h"
#include "../GameController.h"
#include "Logger.h"
```

5.38 /home/nico/Desktop/battleship/src/client/uiElements/ViewGrid.h File Reference

```
#include "../../common/game_state/Ship.h"
#include <wx/wx.h>
```

Classes

- class ViewGrid

5.39 /home/nico/Desktop/battleship/src/common/exceptions/BattleshipException.h File Reference

```
#include <string>
```

Classes

- class BattleshipException

5.40 /home/nico/Desktop/battleship/src/common/game_state/Coordinate.h File Reference

```
#include <compare>
```

Classes

- struct Coordinate

5.41 /home/nico/Desktop/battleship/src/common/game_state/GameState.cpp File Reference

```
#include "GameState.h"
#include "Coordinate.h"
#include "Logger.h"
#include "Player.h"
#include <cassert>
#include <stdexcept>
#include <utility>
```

5.42 /home/nico/Desktop/battleship/src/common/game_state/GameState.h File Reference

```
#include "Coordinate.h"
#include "game_state/Player.h"
#include "game_state/PlayerGrid.h"
#include "network/responses/GameEvent.h"
#include <vector>
```

Classes

- class GameState

5.43 /home/nico/Desktop/battleship/src/common/game_state/Player.cpp File Reference

```
#include "Player.h"
#include <utility>
```

5.44 /home/nico/Desktop/battleship/src/common/game_state/Player.h File Reference

```
#include "uuid.h"
#include <string>
```

Classes

- class Player

5.45 /home/nico/Desktop/battleship/src/common/game_state/PlayerGrid.cpp File Reference

```
#include "game_state/PlayerGrid.h"
#include "game_state/Ship.h"
#include "uuid.h"
#include <utility>
#include <vector>
```

5.46 /home/nico/Desktop/battleship/src/common/game_state/PlayerGrid.h File Reference

```
#include "game_state/Ship.h"
#include "uuid.h"
#include <vector>
```

Classes

- class PlayerGrid

5.47 /home/nico/Desktop/battleship/src/common/game_state/Ship.cpp File Reference

```
#include "Ship.h"
```

5.48 /home/nico/Desktop/battleship/src/common/game_state/Ship.h File Reference

```
#include "../uuid.h"
#include "Coordinate.h"
#include <vector>
```

Classes

- class Ship

5.49 /home/nico/Desktop/battleship/src/common/Logger.cpp File Reference

```
#include "Logger.h"
#include <sstream>
```

5.50 /home/nico/Desktop/battleship/src/common/Logger.h File Reference

```
#include "exceptions/BattleshipException.h"
#include <ctime>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <string>
```

Classes

- class Logger

Macros

- #define LOG(message) Logger::log(message, __func__)

5.50.1 Macro Definition Documentation

5.50.1.1 LOG

```
#define LOG(
              message ) Logger::log(message, __func__)
```

5.51 /home/nico/Desktop/battleship/src/common/network/requests/CallShot.cpp File Reference

```
#include "CallShot.h"
#include "../../game_state/Coordinate.h"
#include "../../uuid.h"
#include "ClientRequest.h"
```

5.52 /home/nico/Desktop/battleship/src/common/network/requests/CallShot.h File Reference

```
#include "ClientRequest.h"
#include "game_state/Coordinate.h"
```

Classes

- class CallShot

5.53 /home/nico/Desktop/battleship/src/common/network/requests/ClientRequest.cpp File Reference

```
#include "ClientRequest.h"
#include "uuid.h"
```

5.54 /home/nico/Desktop/battleship/src/common/network/requests/ClientRequest.h File Reference

```
#include "uuid.h"
#include <compare>
```

Classes

- class ClientRequest

Enumerations

- enum class RequestType {
  JoinGame , StartGame , CallShot , SendEmote ,
  QuitGame , PlayAgain }

5.54.1 Enumeration Type Documentation

5.54.1.1 RequestType

```
enum RequestType  [strong]
```

Enumerator

JoinGame	
StartGame	
CallShot	
SendEmote	
QuitGame	
PlayAgain	

5.55 /home/nico/Desktop/battleship/src/common/network/requests/↩ JoinGame.cpp File Reference

```
#include "JoinGame.h"
#include "../../uuid.h"
#include "ClientRequest.h"
#include <string>
#include <utility>
```

5.56 /home/nico/Desktop/battleship/src/common/network/requests/↩ JoinGame.h File Reference

```
#include "ClientRequest.h"
#include <string>
```

Classes

- class JoinGame

5.57 /home/nico/Desktop/battleship/src/common/network/requests/↩ PlayAgain.cpp File Reference

```
#include "PlayAgain.h"
#include "../../uuid.h"
#include "ClientRequest.h"
```

5.58 /home/nico/Desktop/battleship/src/common/network/requests/↩ PlayAgain.h File Reference

```
#include "ClientRequest.h"
```

Classes

- class PlayAgain

5.59 /home/nico/Desktop/battleship/src/common/network/requests/↩ QuitGame.cpp File Reference

```
#include "QuitGame.h"
#include "../../uuid.h"
#include "ClientRequest.h"
```

5.60 /home/nico/Desktop/battleship/src/common/network/requests/↩ QuitGame.h File Reference

```
#include "ClientRequest.h"
```

Classes

- class QuitGame

5.61 /home/nico/Desktop/battleship/src/common/network/requests/↩ SendEmote.cpp File Reference

```
#include "SendEmote.h"
#include "../../uuid.h"
#include "ClientRequest.h"
#include <string>
#include <utility>
```

5.62 /home/nico/Desktop/battleship/src/common/network/requests/SendEmote.h File Reference

```
#include "ClientRequest.h"
#include "network/responses/EmoteEvent.h"
#include <string>
```

Classes

- class SendEmote

5.63 /home/nico/Desktop/battleship/src/common/network/requests/StartGame.cpp File Reference

```
#include "StartGame.h"
#include "../../game_state/Ship.h"
#include "../../uuid.h"
#include "ClientRequest.h"
#include <utility>
#include <vector>
```

5.64 /home/nico/Desktop/battleship/src/common/network/requests/StartGame.h File Reference

```
#include "../../game_state/Ship.h"
#include "ClientRequest.h"
#include <vector>
```

Classes

- class StartGame

5.65 /home/nico/Desktop/battleship/src/common/network/responses/EmoteEvent.cpp File Reference

```
#include "EmoteEvent.h"
```

5.66 /home/nico/Desktop/battleship/src/common/network/responses/EmoteEvent.h File Reference

```
#include "game_state/Player.h"
#include "network/responses/ServerResponse.h"
#include <array>
```

Classes

- class EmoteEvent

Enumerations

- enum class EmoteType {
  MiddleFinger , RussianWarshipGoFuckYourself , Mocking , BestPirate ,
  Panic , Clown }

5.66.1 Enumeration Type Documentation

5.66.1.1 EmoteType

```
enum EmoteType  [strong]
```

Enumerator

MiddleFinger	
RussianWarshipGoFuckYourself	
Mocking	
BestPirate	
Panic	
Clown	

5.67 /home/nico/Desktop/battleship/src/common/network/responses/ErrorResponse.cpp File Reference

```
#include "ErrorResponse.h"
#include <utility>
```

5.68 /home/nico/Desktop/battleship/src/common/network/responses/ErrorResponse.h File Reference

```
#include "exceptions/BattleshipException.h"
#include "network/responses/ServerResponse.h"
```

Classes

- class ErrorResponse

5.69 /home/nico/Desktop/battleship/src/common/network/responses/GameEvent.cpp File Reference

```
#include "GameEvent.h"
#include "game_state/Coordinate.h"
#include "game_state/Ship.h"
#include "uuid.h"
```

5.70 /home/nico/Desktop/battleship/src/common/network/responses/GameEvent.h File Reference

```
#include "game_state/Coordinate.h"
#include "game_state/Ship.h"
#include "network/responses/ServerResponse.h"
#include "uuid.h"
```

Classes

- class GameEvent

5.71 /home/nico/Desktop/battleship/src/common/network/responses/GameOverEvent.cpp File Reference

```
#include "GameOverEvent.h"
#include "uuid.h"
```

5.72 /home/nico/Desktop/battleship/src/common/network/responses/GameOverEvent.h File Reference

```
#include "network/responses/ServerResponse.h"
#include "uuid.h"
```

Classes

- class GameOverEvent

5.73 /home/nico/Desktop/battleship/src/common/network/responses/JoinGameSuccess.cpp File Reference

```
#include "JoinGameSuccess.h"
```

5.74 /home/nico/Desktop/battleship/src/common/network/responses/JoinGameSuccess.h File Reference

```
#include "network/requests/ClientRequest.h"
#include "network/responses/ServerResponse.h"
#include "uuid.h"
```

Classes

- class JoinGameSuccess

5.75 /home/nico/Desktop/battleship/src/common/network/responses/QuitGameEvent.cpp File Reference

```
#include "QuitGameEvent.h"
#include "ServerResponse.h"
```

5.76 /home/nico/Desktop/battleship/src/common/network/responses/QuitGameEvent.h File Reference

```
#include "ServerResponse.h"
#include "uuid.h"
```

Classes

- class QuitGameEvent

5.77 /home/nico/Desktop/battleship/src/common/network/responses/ServerResponse.cpp File Reference

```
#include "ServerResponse.h"
```

5.78 /home/nico/Desktop/battleship/src/common/network/responses/ServerResponse.h File Reference

```
#include <compare>
```

Classes

- class ServerResponse

Enumerations

- enum class ResponseType {
  GameEvent , EmoteEvent , JoinGameSuccess , StartGameSuccess ,
  GameOverEvent , ErrorResponse , QuitGameEvent }

5.78.1 Enumeration Type Documentation

5.78.1.1 ResponseType

```
enum ResponseType  [strong]
```

Enumerator

GameEvent	
EmoteEvent	
JoinGameSuccess	
StartGameSuccess	
GameOverEvent	
ErrorResponse	
QuitGameEvent	

5.79 /home/nico/Desktop/battleship/src/common/network/responses/StartGameSuccess.cpp File Reference

```
#include "StartGameSuccess.h"
#include <utility>
```

5.80 /home/nico/Desktop/battleship/src/common/network/responses/StartGameSuccess.h File Reference

```
#include "game_state/Player.h"
#include "network/responses/ServerResponse.h"
#include <vector>
```

Classes

- class StartGameSuccess

5.81 /home/nico/Desktop/battleship/src/common/serialization/serialization.h File Reference

```
#include "game_state/Coordinate.h"
#include "game_state/Ship.h"
#include "network/requests/CallShot.h"
#include "network/requests/ClientRequest.h"
#include "network/requests/JoinGame.h"
#include "network/requests/PlayAgain.h"
#include "network/requests/QuitGame.h"
#include "network/requests/SendEmote.h"
#include "network/requests/StartGame.h"
#include "network/responses/EmoteEvent.h"
#include "network/responses/ErrorResponse.h"
#include "network/responses/GameEvent.h"
#include "network/responses/GameOverEvent.h"
#include "network/responses/JoinGameSuccess.h"
#include "network/responses/QuitGameEvent.h"
#include "network/responses/ServerResponse.h"
#include "network/responses/StartGameSuccess.h"
#include <memory>
#include <nlohmann/json.hpp>
```

Functions

- void to_json (nlohmann::json &json, const Coordinate &position)
- void from_json (const nlohmann::json &json, Coordinate &position)
- void to_json (nlohmann::json &json, const uuid &uuid)
- void from_json (const nlohmann::json &json, uuid &uuid_v)
- NLOHMANN_JSON_SERIALIZE_ENUM (Ship::Orientation, {{Ship::Orientation::Vertical, "v"}, {Ship::Orientation::Horizontal, "h"}}) NLOHMANN_JSON_SERIALIZE_ENUM(RequestType

5.81.1 Function Documentation

5.81.1.1 from_json() [1/2]

```
void from_json (
            const nlohmann::json & json,
            Coordinate & position )  [inline]
```

5.81.1.2 from_json() [2/2]

```
void from_json (
            const nlohmann::json & json,
            uuid & uuid_v )  [inline]
```

5.81.1.3 NLOHMANN_JSON_SERIALIZE_ENUM()

```
NLOHMANN_JSON_SERIALIZE_ENUM (
            Ship::Orientation ,
            {{Ship::Orientation::Vertical, "v"}, {Ship::Orientation::Horizontal, "h"}}  )
```

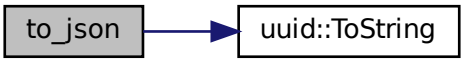
5.81.1.4 to_json() [1/2]

```
void to_json (
            nlohmann::json & json,
            const Coordinate & position )  [inline]
```

5.81.1.5 to_json() [2/2]

```
void to_json (
            nlohmann::json & json,
            const uuid & uuid )  [inline]
```

Here is the call graph for this function:



5.82 /home/nico/Desktop/battleship/src/common/uuid.cpp File Reference

```
#include "uuid.h"
#include <algorithm>
#include <cassert>
#include <iomanip>
#include <ios>
#include <random>
#include <sstream>
#include <string>
```

5.83 /home/nico/Desktop/battleship/src/common/uuid.h File Reference

```
#include <array>
#include <cstdint>
#include <string>
```

Classes

- class uuid
- struct std::hash< uuid >

5.84 /home/nico/Desktop/battleship/src/server/GameInstance.cpp File Reference

```
#include "GameInstance.h"
#include "Logger.h"
#include "ServerNetworkManager.h"
#include "network/responses/ErrorResponse.h"
#include "network/responses/GameEvent.h"
#include "network/responses/GameOverEvent.h"
#include "network/responses/JoinGameSuccess.h"
#include "network/responses/QuitGameEvent.h"
#include "network/responses/ServerResponse.h"
#include <cassert>
```

5.85 /home/nico/Desktop/battleship/src/server/GameInstance.h File Reference

```
#include "game_state/GameState.h"
#include "game_state/Player.h"
#include "network/requests/CallShot.h"
```

```
#include "network/requests/JoinGame.h"
#include "network/requests/QuitGame.h"
#include <mutex>
#include <string>
#include <unordered_map>
#include <vector>
```

Classes

- class GameInstance

5.86 /home/nico/Desktop/battleship/src/server/RequestHandler.cpp File Reference

```
#include "RequestHandler.h"
#include "GameInstance.h"
#include "Logger.h"
#include "ServerNetworkManager.h"
#include "network/requests/CallShot.h"
#include "network/requests/JoinGame.h"
#include "network/requests/QuitGame.h"
#include "network/requests/SendEmote.h"
#include "network/requests/StartGame.h"
#include "network/responses/ErrorResponse.h"
#include "network/responses/JoinGameSuccess.h"
#include "network/responses/QuitGameEvent.h"
#include "network/responses/ServerResponse.h"
#include "network/responses/StartGameSuccess.h"
```

5.87 /home/nico/Desktop/battleship/src/server/RequestHandler.h File Reference

```
#include "GameInstance.h"
#include "network/requests/ClientRequest.h"
#include "network/responses/ServerResponse.h"
#include <memory>
```

Classes

- class RequestHandler

5.88 /home/nico/Desktop/battleship/src/server/ServerNetwork↩ Manager.cpp File Reference

```
#include "ServerNetworkManager.h"
#include "Logger.h"
#include "RequestHandler.h"
#include "network/responses/ServerResponse.h"
#include "serialization/serialization.h"
#include <nlohmann/json.hpp>
#include <sstream>
#include <string>
```

5.89 /home/nico/Desktop/battleship/src/server/ServerNetworkManager.h File Reference

```
#include "GameInstance.h"
#include "game_state/Player.h"
#include "network/responses/ServerResponse.h"
#include "sockpp/tcp_acceptor.h"
#include "sockpp/tcp_connector.h"
#include "sockpp/tcp_socket.h"
#include <functional>
#include <shared_mutex>
#include <thread>
#include <unordered_map>
```

Classes

- class ServerNetworkManager

Index