AntMazeServerCPP

Generated by Doxygen 1.9.6

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 Constants Namespace Reference

Namespace that contains constants used by the program.

Variables

- int **DIFFICULTY1_MAX_PLAYERS**

  Maximum number of players allowed on difficulty 1.
- int **DIFFICULTY1_SIDE_SIZE**

  Size of the side of the game board on difficulty 1.
- int **DIFFICULTY1_NBFOOD**

  Number of food items on the game board on difficulty 1.
- int **DIFFICULTY1_NESTLINE**

  Line coordinate of the nest on the game board on difficulty 1.
- int **DIFFICULTY1_NESTCOLUMN**

  Column coordinate of the nest on the game board on difficulty 1.
- int **DIFFICULTY2_MAX_PLAYERS**

  Maximum number of players allowed on difficulty 2.
- int **DIFFICULTY2_SIDE_SIZE**

  Size of the side of the game board on difficulty 2.
- int **DIFFICULTY2_NBFOOD**

  Number of food items on the game board on difficulty 2.
- int **DIFFICULTY2_NESTLINE**

  Line coordinate of the nest on the game board on difficulty 2.
- int **DIFFICULTY2_NESTCOLUMN**

  Column coordinate of the nest on the game board on difficulty 2.
- int **DIFFICULTY3_MAX_PLAYERS**

  Maximum number of players allowed on difficulty 3.
- int **DIFFICULTY3_SIDE_SIZE**

  Size of the side of the game board on difficulty 3.
- int **DIFFICULTY3_NBFOOD**

  Number of food items on the game board on difficulty 3.
- int **DIFFICULTY3_NESTLINE**

  Line coordinate of the nest on the game board on difficulty 3.

- int **DIFFICULTY3_NESTCOLUMN**

  Column coordinate of the nest on the game board on difficulty 3.
- float **PHEROMON_DECREASE_AMOUNT**

  Amount by which the pheromone trail decreases over time.
- float **PHEROMON_DROP_AMOUNT**

  Amount of pheromones dropped by an ant when it finds food.
- bool **VERBOSE**

  Whether the program should output verbose messages.
- unsigned short **SERVER_PORT**

  Port used by the server for network communication.

5.1.1 Detailed Description

Namespace that contains constants used by the program.

5.2 JSON Namespace Reference

A namespace containing functions for reading and creating JSON messages.

Functions

- std::string getUUID (const boost::property_tree::ptree &root)

  Returns the UUID from a JSON message.
- std::string getType (const boost::property_tree::ptree &root)

  Returns the type from a JSON message.
- int getDifficultyJoin (const boost::property_tree::ptree &root)

  Returns the difficulty from a JSON join message.
- std::string getMove (const boost::property_tree::ptree &root)

  Returns the move from a JSON move message.
- Maze ∗ getMaze (const boost::property_tree::ptree &root)

  Returns the maze from a JSON maze message.
- std::vector< float > getPheromons (const boost::property_tree::ptree &root)

  Returns the pheromons from a JSON info message.
- void LoadOptionFile (std::string _path)

  Loads the options from a JSON file.
- std::string createGeneric (boost::uuids::uuid _uuid, boost::property_tree::ptree &_root)

  Creates a generic JSON message.
- std::string createJoin (boost::uuids::uuid _uuid, int _difficulty)

  Creates a JSON join message.
- std::string createMove (boost::uuids::uuid _uuid, std::string _move)

  Creates a JSON move message.
- std::string createokMaze (boost::uuids::uuid _uuid, Maze _maze)

  Creates a JSON maze message.
- std::string createInfo (boost::uuids::uuid _uuid, std::vector< float > _pheromons)

  Creates a JSON info message.

5.2.1 Detailed Description

A namespace containing functions for reading and creating JSON messages.

5.2.2 Function Documentation

5.2.2.1 createGeneric()

```
std::string JSON::createGeneric (
            boost::uuids::uuid _uuid,
            boost::property_tree::ptree & _root )
```

Creates a generic JSON message.

Parameters

_uuid	The UUID of the sender.
_root	The root of the JSON message.

Returns

The JSON message as a string.

5.2.2.2 createInfo()

```
std::string JSON::createInfo (
            boost::uuids::uuid _uuid,
            std::vector< float > _pheromons )
```

Creates a JSON info message.

Parameters

_uuid	The UUID of the sender.
_pheromons	The pheromons to be sent.

Returns

The JSON message as a string.

5.2.2.3 createJoin()

```
std::string JSON::createJoin (
            boost::uuids::uuid _uuid,
            int _difficulty )
```

Creates a JSON join message.

Parameters

_uuid	The UUID of the sender.
_difficulty	The difficulty of the game.

Returns

The JSON message as a string.

5.2.2.4 createMove()

```
std::string JSON::createMove (
            boost::uuids::uuid _uuid,
            std::string _move )
```

Creates a JSON move message.

Parameters

_uuid	The UUID of the sender.
_move	The move to be sent.

Returns

The JSON message as a string.

5.2.2.5 createokMaze()

```
std::string JSON::createokMaze (
            boost::uuids::uuid _uuid,
            Maze _maze )
```

Creates a JSON maze message.

Parameters

_uuid	The UUID of the sender.
_maze	The maze to be sent.

Returns

The JSON message as a string.

5.2.2.6 getDifficultyJoin()

```
int JSON::getDifficultyJoin (
            const boost::property_tree::ptree & root )
```

Returns the difficulty from a JSON join message.

Parameters

root	The root of the JSON message.

Returns

The difficulty as an integer.

5.2.2.7 getMaze()

```
Maze * JSON::getMaze (
            const boost::property_tree::ptree & root )
```

Returns the maze from a JSON maze message.

Parameters

root	The root of the JSON message.

Returns

A pointer to the maze.

5.2.2.8 getMove()

```
std::string JSON::getMove (
            const boost::property_tree::ptree & root )
```

Returns the move from a JSON move message.

Parameters

root	The root of the JSON message.

Returns

The move as a string.

5.2.2.9 getPheromons()

```
std::vector< float > JSON::getPheromons (
            const boost::property_tree::ptree & root )
```

Returns the pheromons from a JSON info message.

Parameters

root	The root of the JSON message.

Returns

A vector of pheromons.

5.2.2.10 getType()

```
std::string JSON::getType (
            const boost::property_tree::ptree & root )
```

Returns the type from a JSON message.

Parameters

root	The root of the JSON message.

Returns

The type as a string.

5.2.2.11 getUUID()

```
std::string JSON::getUUID (
            const boost::property_tree::ptree & root )
```

Returns the UUID from a JSON message.

Parameters

root	The root of the JSON message.

Returns

The UUID as a string.

5.2.2.12 LoadOptionFile()

```
void JSON::LoadOptionFile (
            std::string _path )
```

Loads the options from a JSON file.

Parameters

_path	The path to the file.

Chapter 6

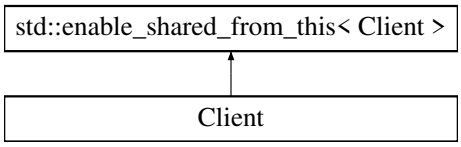
Class Documentation

6.1 Client Class Reference

A class representing a client.

`#include <client.h>`

Inheritance diagram for Client:



Public Member Functions

- Client (boost::asio::io_context &io_context1, std::string _adress, short _port)
  Constructs a new client.
- void join (int _difficulty)
  Sends a join request to the server.
- void move (std::string _move)
  Sends a move request to the server.
- void handleReadClient (const boost::system::error_code &ec, size_t bytes_transferred)
  Handles the completion of a read operation.
- void **listenClient** ()
  Listens for incoming messages.
- void setMaze (Maze *_maze)
  Sets the maze of the client.

6.1.1 Detailed Description

A class representing a client.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Client()

```
Client::Client (
            boost::asio::io_context & io_context1,
            std::string _adress,
            short _port )
```

Constructs a new client.

Parameters

io_context1	The IO context to use for the client.
_adress	The address of the server to connect to.
_port	The port of the server to connect to.

6.1.3 Member Function Documentation

6.1.3.1 handleReadClient()

```
void Client::handleReadClient (
            const boost::system::error_code & ec,
            size_t bytes_transferred )
```

Handles the completion of a read operation.

Parameters

ec	The error code of the operation.
bytes_transferred	The number of bytes transferred.

6.1.3.2 join()

```
void Client::join (
            int _difficulty )
```

Sends a join request to the server.

Parameters

_difficulty	The desired difficulty of the game.

6.1.3.3 move()

```
void Client::move (
            std::string _move )
```

Sends a move request to the server.

Parameters

_move	The move to make.

6.1.3.4 setMaze()

```
void Client::setMaze (
            Maze * _maze )  [inline]
```

Sets the maze of the client.

Parameters

_maze	A pointer to the maze to set.

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

- Client/client.h
- Client/client.cpp

6.2 game Class Reference

A class representing a game.

```
#include <Game.h>
```

Public Member Functions

- game (const int &_difficulty, const int &_max_nb_players, int size_side_maze)

  Constructs a new game.

- void join (const boost::uuids::uuid &_player_uuid, std::shared_ptr< session > _session)

  Adds a player to the game.
- void move (const boost::uuids::uuid &_player, std::string _move)

  Moves a player in the game.
- void **decreasePheromons** ()

  Decreases the value of all pheromons in the game, and send the result to all players connected.
- std::vector< float > getPheromons ()

  Gets the pheromons in the game.
- Maze * getMaze ()

  Gets the maze in the game.
- int getMax_Players ()

  Gets the maximum number of players allowed in the game.
- int getNb_Players ()

  Gets the current number of players in the game.

6.2.1 Detailed Description

A class representing a game.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 game()

```
game::game (
            const int & _difficulty,
            const int & _max_nb_players,
            int size_side_maze )
```

Constructs a new game.

Parameters

_difficulty	The difficulty of the game.
_max_nb_players	The maximum number of players allowed in the game.
size_side_maze	The size of the side of the maze.

6.2.3 Member Function Documentation

6.2.3.1 getMax_Players()

```
int game::getMax_Players ( )  [inline]
```

Gets the maximum number of players allowed in the game.

Returns

The maximum number of players allowed in the game.

6.2.3.2 getMaze()

```
Maze * game::getMaze ( )  [inline]
```

Gets the maze in the game.

Returns

A pointer to the maze in the game.

6.2.3.3 getNb_Players()

```
int game::getNb_Players ( )  [inline]
```

Gets the current number of players in the game.

Returns

The current number of players in the game.

6.2.3.4 getPheromons()

```
std::vector< float > game::getPheromons ( )  [inline]
```

Gets the pheromons in the game.

Returns

A vector of pheromons in the game.

6.2.3.5 join()

```
void game::join (
            const boost::uuids::uuid & _player_uuid,
            std::shared_ptr< session > _session )
```

Adds a player to the game.

Parameters

_player_uuid	The UUID of the player.
_session	The session of the player.

6.2.3.6 move()

```
void game::move (
            const boost::uuids::uuid & _player,
            std::string _move )
```

Moves a player in the game.

Parameters

_player	The UUID of the player.
_move	The move to make.

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

- Server/Game.h
- Server/Game.cpp

6.3 Player Struct Reference

A struct representing a player.

```
#include <player.h>
```

Public Attributes

- boost::uuids::uuid **p_uuid**

  The UUID of the player.
- int **actual_column**

  The current column of the player.
- int **actual_line**

  The current line of the player.
- bool **has_food**

  A flag indicating if the player has food.
- std::shared_ptr< session > **_session**

  A shared pointer to the session of the player.

6.3.1 Detailed Description

A struct representing a player.

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

- Server/player.h

6.4 server Class Reference

A class representing a server that manages games and sessions.

`#include <server.h>`

Public Member Functions

- server (boost::asio::io_context &service, unsigned short port)
  Constructs a new server.
- void **startAccept** ()
  Starts accepting new connections, and send all new connection in new sessions.
- void handleAccept (std::shared_ptr< session > new_session, const boost::system::error_code &ec)
  Handles the acceptance of a new session.
- void findGameWithDifficulty (int _difficulty, boost::uuids::uuid _uuid, std::shared_ptr< session > _session)
  Attempts to matchmake a player with a game of the specified difficulty.
- game * getGame (const boost::uuids::uuid &_uuid)
  Gets a game with the specified UUID.
- std::vector< game > getListofAvailaibleGames ()
  Returns a list of available games.

6.4.1 Detailed Description

A class representing a server that manages games and sessions.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 server()

```
server::server (
            boost::asio::io_context & service,
            unsigned short port )
```

Constructs a new server.

Parameters

service	The IO context to use for the server.
port	The port to listen on.

6.4.3 Member Function Documentation

6.4.3.1 findGameWithDifficulty()

```
void server::findGameWithDifficulty (
            int _difficulty,
            boost::uuids::uuid _uuid,
            std::shared_ptr< session > _session )
```

Attempts to matchmake a player with a game of the specified difficulty.

Parameters

_difficulty	The desired difficulty of the game.
_uuid	The UUID of the player.
_session	The session of the player.

6.4.3.2 getGame()

```
game * server::getGame (
            const boost::uuids::uuid & _uuid )
```

Gets a game with the specified UUID.

Parameters

_uuid	The UUID of the game to get.

Returns

A pointer to the game with the specified UUID, or `nullptr` if no such game exists.

6.4.3.3 getListofAvailaibleGames()

```
std::vector< game > server::getListofAvailaibleGames ( )  [inline]
```

Returns a list of available games.

Returns

A vector of `game` objects.

6.4.3.4 handleAccept()

```
void server::handleAccept (
            std::shared_ptr< session > new_session,
            const boost::system::error_code & ec )
```

Handles the acceptance of a new session.

Parameters

new_session	The new session.
ec	The error code of the operation.

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

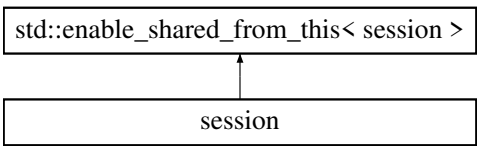
- Server/server.h
- Server/Server.cpp

6.5 session Class Reference

A class representing a session between a client and the server.

`#include <session.h>`

Inheritance diagram for session:



Public Member Functions

- session (boost::asio::io_context &service, server *_server)

  Constructs a new session.
- void **listen** ()

  Listens for incoming messages.
- void handle_write (const std::error_code &ec)

  Handles the completion of a write operation.
- void handle_read (const boost::system::error_code &ec, size_t bytes_transferred)

  Handles the completion of a read operation.

- void sendMaze (boost::uuids::uuid _uuid, Maze *_maze)

  Sends the maze to the client.
- void sendPheromons (boost::uuids::uuid _uuid, const std::vector< float > &_pheromons)

  Sends the pheromons to the client.
- void sendString (std::string _message)

  Sends a string message to the client.
- socket_t & socket ()

  Gets the socket of the session.
- void setGame (game *_game)

  Set the game object for the current instance.

6.5.1 Detailed Description

A class representing a session between a client and the server.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 session()

```
session::session (
            boost::asio::io_context & service,
            server * _server )
```

Constructs a new session.

Parameters

service	The IO context to use for the session.
_server	A pointer to the server that created the session.

6.5.3 Member Function Documentation

6.5.3.1 handle_read()

```
void session::handle_read (
            const boost::system::error_code & ec,
            size_t bytes_transferred )
```

Handles the completion of a read operation.

Parameters

ec	The error code of the operation.
bytes_transferred	The number of bytes transferred.

6.5.3.2 handle_write()

```
void session::handle_write (
            const std::error_code & ec )
```

Handles the completion of a write operation.

Parameters

ec	The error code of the operation.

6.5.3.3 sendMaze()

```
void session::sendMaze (
            boost::uuids::uuid _uuid,
            Maze * _maze )
```

Sends the maze to the client.

Parameters

_uuid	The UUID of the player.
_maze	A pointer to the maze to send.

6.5.3.4 sendPheromons()

```
void session::sendPheromons (
            boost::uuids::uuid _uuid,
            const std::vector< float > & _pheromons )
```

Sends the pheromons to the client.

Parameters

_uuid	The UUID of the player.
_pheromons	A vector of pheromons to send.

6.5.3.5 sendString()

```
void session::sendString (
            std::string _message )
```

Sends a string message to the client.

Parameters

_message	The message to send.

6.5.3.6 setGame()

```
void session::setGame (
            game * _game )  [inline]
```

Set the game object for the current instance.

Parameters

_game	Pointer to the game object to be set

6.5.3.7 socket()

```
socket_t & session::socket ( )  [inline]
```

Gets the socket of the session.

Returns

The socket of the session.

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

- Server/session.h
- Server/Session.cpp

Chapter 7

File Documentation

7.1 Client/client.h File Reference

Header file for the `Client` class.

```
#include <boost/asio.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/property_tree/ptree.hpp>
#include <iostream>
#include <memory>
#include <boost/uuid/uuid.hpp>
#include "../Maze/libAntMaze.h"
#include "../JSON.h"
```

Classes

- class Client

  A class representing a client.

Macros

- #define **NULL_UUID** { 00000000 - 0000 - 0000 - 0000 - 000000000000 }

7.1.1 Detailed Description

Header file for the `Client` class.

7.2 client.h

Go to the documentation of this file.

```
00001
00006 #include <boost/asio.hpp>
00007 #include <boost/property_tree/json_parser.hpp>
00008 #include <boost/property_tree/ptree.hpp>
00009 #include <iostream>
00010 #include <memory>
00011 #include <boost/uuid/uuid.hpp>
00012 #include "../Maze/libAntMaze.h"
00013 #include "../JSON.h"
00014
00015 #define NULL_UUID    { 00000000 - 0000 - 0000 - 0000 - 000000000000 }
00016
00017 using boost::asio::ip::tcp;
00018
00019
00020
00025 class Client : public std::enable_shared_from_this<Client> {
00026 private:
00030     boost::asio::io_context& p_io_context;
00031
00035     tcp::socket p_socket_client;
00036
00040     boost::uuids::uuid p_uuid;
00041
00045     boost::asio::streambuf p_buffer{ 2048 };
00046
00050     Maze* p_maze;
00051
00052 public:
00059     Client(boost::asio::io_context& io_context1, std::string _adress, short _port);
00060
00065     void join(int _difficulty);
00066
00067
00068
00073     void move(std::string _move);
00074
00080     void handleReadClient(const boost::system::error_code& ec,
00081         size_t bytes_transferred);
00082
00086     void listenClient();
00087
00092     void setMaze(Maze* _maze) { p_maze = _maze; }
00093 };
```

7.3 JSON.h File Reference

Header file for the JSON namespace.

```
#include <boost/property_tree/ptree.hpp>
#include <boost/uuid/uuid.hpp>
#include <vector>
#include <iostream>
#include "Maze/libAntMaze.h"
```

Namespaces

- namespace JSON

  A namespace containing functions for reading and creating JSON messages.

Functions

- std::string JSON::getUUID (const boost::property_tree::ptree &root)

  Returns the UUID from a JSON message.
- std::string JSON::getType (const boost::property_tree::ptree &root)

  Returns the type from a JSON message.
- int JSON::getDifficultyJoin (const boost::property_tree::ptree &root)

  Returns the difficulty from a JSON join message.
- std::string JSON::getMove (const boost::property_tree::ptree &root)

  Returns the move from a JSON move message.
- Maze * JSON::getMaze (const boost::property_tree::ptree &root)

  Returns the maze from a JSON maze message.
- std::vector< float > JSON::getPheromons (const boost::property_tree::ptree &root)

  Returns the pheromons from a JSON info message.
- void JSON::LoadOptionFile (std::string _path)

  Loads the options from a JSON file.
- std::string JSON::createGeneric (boost::uuids::uuid _uuid, boost::property_tree::ptree &_root)

  Creates a generic JSON message.
- std::string JSON::createJoin (boost::uuids::uuid _uuid, int _difficulty)

  Creates a JSON join message.
- std::string JSON::createMove (boost::uuids::uuid _uuid, std::string _move)

  Creates a JSON move message.
- std::string JSON::createokMaze (boost::uuids::uuid _uuid, Maze _maze)

  Creates a JSON maze message.
- std::string JSON::createInfo (boost::uuids::uuid _uuid, std::vector< float > _pheromons)

  Creates a JSON info message.

7.3.1 Detailed Description

Header file for the JSON namespace.

7.4 JSON.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <boost/property_tree/ptree.hpp>
00009 #include <boost/uuid/uuid.hpp>
00010 #include <vector>
00011 #include <iostream>
00012 #include "Maze/libAntMaze.h"
00013
00018 namespace JSON {
00019
00025     std::string getUUID(const boost::property_tree::ptree& root);
00026
00032     std::string getType(const boost::property_tree::ptree& root);
00033
00039     int getDifficultyJoin(const boost::property_tree::ptree& root);
00040
00046     std::string getMove(const boost::property_tree::ptree& root);
00047
00053     Maze* getMaze(const boost::property_tree::ptree& root);
00054
00060     std::vector <float> getPheromons(const boost::property_tree::ptree& root);
00061
00066     void LoadOptionFile(std::string _path);
```

```
00067
00074     std::string createGeneric(boost::uuids::uuid _uuid,boost::property_tree::ptree& _root);
00075
00082     std::string createJoin(boost::uuids::uuid _uuid, int _difficulty);
00083
00090     std::string createMove(boost::uuids::uuid _uuid, std::string _move);
00091
00098     std::string createokMaze(boost::uuids::uuid _uuid, Maze _maze);
00099
00106     std::string createInfo(boost::uuids::uuid _uuid, std::vector<float> _pheromons);
00107
00108 }
```

7.5 resource.h

```
00001 //{{NO_DEPENDENCIES}}
00002 // Microsoft Visual C++ generated include file.
00003 // Used by Serveur_test_2.rc
00004
00005 // Valeurs par défaut suivantes des nouveaux objets
00006 //
00007 #ifdef APSTUDIO_INVOKED
00008 #ifndef APSTUDIO_READONLY_SYMBOLS
00009 #define _APS_NEXT_RESOURCE_VALUE        101
00010 #define _APS_NEXT_COMMAND_VALUE         40001
00011 #define _APS_NEXT_CONTROL_VALUE         1001
00012 #define _APS_NEXT_SYMED_VALUE           101
00013 #endif
00014 #endif
```

7.6 constants.h

```
00001 #pragma once
00002
00006 namespace Constants {
00007
00008     // Difficulty 1
00009
00013     extern int DIFFICULTY1_MAX_PLAYERS;
00014
00018     extern int DIFFICULTY1_SIDE_SIZE;
00019
00023     extern int DIFFICULTY1_NBFOOD;
00024
00028     extern int DIFFICULTY1_NESTLINE;
00029
00033     extern int DIFFICULTY1_NESTCOLUMN;
00034
00035     // Difficulty 2
00036
00040     extern int DIFFICULTY2_MAX_PLAYERS;
00041
00045     extern int DIFFICULTY2_SIDE_SIZE;
00046
00050     extern int DIFFICULTY2_NBFOOD;
00051
00055     extern int DIFFICULTY2_NESTLINE;
00056
00060     extern int DIFFICULTY2_NESTCOLUMN;
00061
00062     // Difficulty 3
00063
00067     extern int DIFFICULTY3_MAX_PLAYERS;
00068
00072     extern int DIFFICULTY3_SIDE_SIZE;
00073
00077     extern int DIFFICULTY3_NBFOOD;
00078
00082     extern int DIFFICULTY3_NESTLINE;
00083
00087     extern int DIFFICULTY3_NESTCOLUMN;
00088
00092     extern float PHEROMON_DECREASE_AMOUNT;
00093
00097     extern float PHEROMON_DROP_AMOUNT;
00098
00102     extern bool VERBOSE;
00103
00107     extern unsigned short SERVER_PORT;
00108 }
```

7.7 Server/Game.h File Reference

Header file for the `game` class.

```
#include <functional>
#include <iostream>
#include <vector>
#include <boost/uuid/uuid.hpp>
#include <memory>
#include "../JSON.h"
#include "../Maze/libAntMaze.h"
#include "session.h"
#include "player.h"
#include "constants.h"
```

Classes

- class game

  A class representing a game.

7.7.1 Detailed Description

Header file for the `game` class.

7.8 Game.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <functional>
00009 #include <iostream>
00010 #include <vector>
00011
00012 #include <boost/uuid/uuid.hpp>
00013 #include <memory>
00014 #include "../JSON.h"
00015 #include "../Maze/libAntMaze.h"
00016 #include "session.h"
00017 #include "player.h"
00018 #include "constants.h"
00019
00024 class game
00025 {
00026 private:
00030     int difficulty;
00031
00035     std::vector<Player> p_players;
00036
00040     int MAX_PLAYERS;
00041
00045     int p_actual_players;
00046
00050     std::vector<float> p_pheromons;
00051
00055     int numberOfTiles;
00056
00060     Maze* p_Maze;
00061
00062 public:
00069     game(const int& _difficulty, const int& _max_nb_players, int size_side_maze);
00070
```

```
00076     void join(const boost::uuids::uuid& _player_uuid, std::shared_ptr<session> _session);
00077
00078
00079
00085     void move(const boost::uuids::uuid& _player, std::string _move);
00086
00087
00091     void decreasePheromons();
00092
00097     std::vector<float> getPheromons() { return p_pheromons; };
00098
00103     Maze* getMaze() { return p_Maze; };
00104
00109     int getMax_Players() { return MAX_PLAYERS; };
00110
00115     int getNb_Players() { return p_actual_players; };
00116
00117
00118 };
```

7.9 Server/player.h File Reference

Header file for the `Player` struct.

```
#include <boost/uuid/uuid.hpp>
#include "session.h"
```

Classes

- struct Player

  A struct representing a player.

7.9.1 Detailed Description

Header file for the `Player` struct.

7.10 player.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <boost/uuid/uuid.hpp>
00009 #include "session.h"
00010
00015 struct Player {
00016
00020     boost::uuids::uuid p_uuid;
00021
00025     int actual_column;
00026
00030     int actual_line;
00031
00035     bool has_food;
00036
00040     std::shared_ptr<session> _session;
00041 };
00042
```

7.11 Server/server.h File Reference

Header file for the `server` class.

```
#include <vector>
#include <map>
#include <memory>
#include <boost/asio.hpp>
#include <boost/uuid/uuid.hpp>
#include <boost/uuid/uuid_io.hpp>
#include "game.h"
#include "constants.h"
```

Classes

- class server

  A class representing a server that manages games and sessions.

7.11.1 Detailed Description

Header file for the `server` class.

7.12 server.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <vector>
00009 #include <map>
00010 #include <memory>
00011
00012 #include <boost/asio.hpp>
00013 #include <boost/uuid/uuid.hpp>
00014 #include <boost/uuid/uuid_io.hpp>
00015
00016
00017 #include "game.h"
00018 #include "constants.h"
00019
00020
00025 class server {
00026     using acceptor_t = boost::asio::ip::tcp::acceptor;
00027     using endpoint_t = boost::asio::ip::tcp::endpoint;
00028     using socket_t = boost::asio::ip::tcp::socket;
00029
00030
00031 public:
00037     server(boost::asio::io_context& service, unsigned short port);
00038
00042     void startAccept();
00043
00044
00050     void handleAccept(std::shared_ptr<session> new_session,
00051         const boost::system::error_code& ec);
00052
00059     void findGameWithDifficulty(int _difficulty, boost::uuids::uuid _uuid, std::shared_ptr<session>
      _session);
00060
00066     game* getGame(const boost::uuids::uuid& _uuid);
00067
00072     std::vector<game> getListofAvailaibleGames() { return p_games; };
00073
```

```
00074 private:
00078     std::vector<game> p_games;
00079
00083     std::vector <std::pair<boost::uuids::uuid, game» p_players_game;
00084
00088     boost::asio::io_context& p_context;
00089
00093     acceptor_t p_acceptor;
00094 };
```

7.13 Server/session.h File Reference

Header file for the `session` class.

```
#include <array>
#include <memory>
#include <boost/asio.hpp>
#include <boost/uuid/uuid.hpp>
```

Classes

- class session

  A class representing a session between a client and the server.

7.13.1 Detailed Description

Header file for the `session` class.

7.14 session.h

Go to the documentation of this file.

```
00001
00006 #pragma once
00007
00008 #include <array>
00009 #include <memory>
00010
00011 #include <boost/asio.hpp>
00012 #include <boost/uuid/uuid.hpp>
00013
00014 class game;
00015 class server;
00016 class Maze;
00017
00022 class session : public std::enable_shared_from_this<session> {
00023
00024     using endpoint_t = boost::asio::ip::tcp::endpoint;
00025     using socket_t = boost::asio::ip::tcp::socket;
00026
00027 public:
00033     session(boost::asio::io_context& service, server* _server);
00034
00038     void listen();
00039
00044     void handle_write(const std::error_code& ec);
00045
00051     void handle_read(const boost::system::error_code& ec,
00052         size_t bytes_transferred);
00053
00059     void sendMaze(boost::uuids::uuid _uuid, Maze* _maze);
00060
```

```
00066     void sendPheromons(boost::uuids::uuid _uuid, const std::vector<float>& _pheromons);
00067
00072     void sendString(std::string _message);
00073
00078     socket_t& socket() { return p_socket; };
00079
00080
00081
00087     void setGame(game* _game) { p_game = _game; };
00088
00089 private:
00093     socket_t p_socket;
00094
00098     server* p_origin;
00099
00103     game* p_game;
00104
00108     boost::asio::streambuf buffer{ 2048 };
00109 };
```

Index