My Project

Generated by Doxygen 1.8.6

Fri Apr 7 2017 11:33:47

Contents

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 documented files with brief descriptions:

Chapter 4

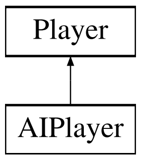
Class Documentation

4.1 AIPlayer Class Reference

the player which will think for himself or herslef

```
#include <aiplayer.h>
```

Inheritance diagram for AIPlayer:



Public Member Functions

- **AIPlayer** (int playerNum, int plyDepth)
- bool **takeTurn** (GUIBoard ∗board)

Additional Inherited Members

4.1.1 Detailed Description

the player which will think for himself or herslef

4.1.2 Description

The AI part of the game that deals with gametrees and interacts with the board based on gametree's decision

4.1.3 License

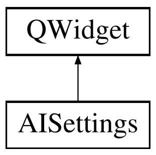
Copyright belongs to Juan du Preez (15189016)

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

- aiplayer.h
- aiplayer.cpp

4.2 AISettings Class Reference

Inheritance diagram for AISettings:



Public Member Functions

- **AISettings** (QWidget ∗parent=0)
- void **setGame** (Game ∗game, QWidget ∗tblWidget)

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

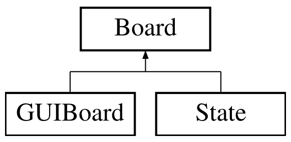
- aisettings.h
- aisettings.cpp

4.3 Board Class Reference

the main area of play in the game

```
#include <board.h>
```

Inheritance diagram for Board:



Public Member Functions

- Board (const Board &other)
- virtual bool enterSeed (int row, int col, bool clockwise)
- virtual bool enterTakasaSeed (int row, int col, bool clockwise)
- virtual bool **makeMtajiMove** (int row, int col, int clockwise)
- bool **isMtajiMove** (int row, int col, int clockwise)
- bool isTakasa ()
- bool isTakasaNyumba ()
- bool isNamua ()
- bool isLosingPosition ()
- bool **isEmpty** (int row, int col)
- virtual void print ()
- virtual void **possibleMoves** (bool player)

Static Public Attributes

- static const bool **CLOCKWISE** = true
- static const bool **ANTICLOCKWISE** = false
- static const bool **PLAYER1** = false
- static const bool **PLAYER2** = true

Protected Member Functions

- virtual bool sow (int &row, int &col, bool clockwise, int hand)
- virtual int take (int row, int col)
- virtual int capture (int row, int col)

Protected Attributes

- int board [4][8]
- int stack1
- int stack2
- bool **isNamuaVar**

4.3.1 Detailed Description

the main area of play in the game

4.3.2 Description

Most of the game functionality is in this class including the sowing of the seeds and seeds themselves.

4.3.3 License

Copyright belongs to Juan du Preez (15189016)

4.3.4 Constructor & Destructor Documentation

4.3.4.1 Board::Board (const Board & *other*)

Copy constructor

4.3.5 Member Function Documentation

4.3.5.1 int Board::capture (int *row,* int *col*) `[protected], [virtual]`

helper function: captures the row opposite row and col

Reimplemented in GUIBoard.

4.3.5.2 bool Board::enterSeed (int *row,* int *col,* bool *clockwise*) `[virtual]`

Namua entering a seed functionality

4.3.5.3 bool Board::enterTakasaSeed (int *row*, int *col*, bool *clockwise*) [virtual]

Namua Takasa situation functionality

4.3.5.4 bool Board::isLosingPosition ( )

returns true if a player has lost the game

4.3.5.5 bool Board::isNamua ( )

returns true if there are still seeds left off the board

4.3.5.6 bool Board::isTakasa ( )

returns true if no captures are possible

4.3.5.7 bool Board::isTakasaNyumba ( )

returns true if in Takasa and landed in house

4.3.5.8 void Board::print ( ) [virtual]

outputs the board and stack variables

Reimplemented in GUIBoard.

4.3.5.9 bool Board::sow (int & *row*, int & *col*, bool *clockwise*, int *hand*) [protected], [virtual]

helper function: has sow around functionality

Reimplemented in GUIBoard.

4.3.5.10 int Board::take (int *row*, int *col*) [protected], [virtual]

helper function: takes seeds

Reimplemented in GUIBoard.

4.3.6 Member Data Documentation

4.3.6.1 int Board::board[4][8] [protected]

the main 4 x 8 board in which seeds are sown

4.3.6.2 int Board::stack1 [protected]

the initial stack of player 1

4.3.6.3 int Board::stack2 `[protected]`

the initial stack of player 2

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

- board.h
- board.cpp

4.4 Game Class Reference

facade to interact with outside world

`#include <game.h>`

Public Member Functions

- void initialize (QWidget ∗w)
- void initPvsP (QWidget ∗w)
- void initPvsAI (QWidget ∗w, int plyDepth)
- void initAIvsAI (QWidget ∗w, int plyDepth1, int plyDepth2)
- void playAIvAI ()
- void setCurData (int curRow, int curCol)
- void sendData (bool direction)
- bool isPvAI ()

4.4.1 Detailed Description

facade to interact with outside world

4.4.2 Description

A combination of all other functionality, providing a single point for the external interface to interact with

4.4.3 License

Copyright belongs to Juan du Preez (15189016)

4.4.4 Member Function Documentation

4.4.4.1 void Game::initAIvsAI (QWidget ∗ *w,* int *plyDepth1,* int *plyDepth2*)

Sets up the game for AI vs AI

4.4.4.2 void Game::initialize (QWidget ∗ *w*)

Sets the game to it's initial state

4.4.4.3 void Game::initPvsAI (QWidget ∗ *w,* int *plyDepth*)

Sets up the game for Player vs AI

4.4.4.4 void Game::initPvsP (QWidget ∗ *w*)

Sets up the game for Player vs Player

4.4.4.5 bool Game::isPvAI ( )

used for AISettings. To display two AIs or just one

4.4.4.6 void Game::playAIvAI ( )

starts the game for AI vs AI

4.4.4.7 void Game::sendData (bool *direction*)

sends current row, column, and direction to a player for processing

Parameters

direction	0 for left and 1 for right as indicated in Board's constants

4.4.4.8 void Game::setCurData (int *curRow,* int *curCol*)

a means of setting current row and col

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

- game.h
- game.cpp

4.5 GameTree Class Reference

the "brain" behind the AI functionality

```
#include <gametree.h>
```

Public Member Functions

- **GameTree** (Board ∗board, bool player, int plyDepth)
- void destroy (State ∗root)
- Move makeBestMove (GUIBoard ∗board)
- vector< Move ∗ > ∗ getPossibleMoves (State ∗state)
- State ∗ getNextState (State ∗state, Move ∗move)
- int alphaBetaPruning (State ∗cur, int curDepth)
- void print ()

4.5.1 Detailed Description

the "brain" behind the AI functionality

4.5.2 Description

This is the tree that the AIPlayer will use to look ahead in the game

4.5.3 License

4.5.4 Member Function Documentation

4.5.4.1 int GameTree::alphaBetaPruning (**State** ∗ *cur,* int *curDepth*)

explanations are in comments in the function itself

4.5.4.2 void GameTree::destroy (**State** ∗ *root*)

deletes all nodes to avoid memory leaks

4.5.4.3 **State** ∗ GameTree::getNextState (**State** ∗ *state,* **Move** ∗ *move*)

based on current state and a move, makes next state

4.5.4.4 vector< **Move** ∗ > ∗ GameTree::getPossibleMoves (**State** ∗ *state*)

generates list of possible moves based on current state

4.5.4.5 **Move** GameTree::makeBestMove (**GUIBoard** ∗ *board*)

the umbrella function which is the interface to external classes

4.5.4.6 void GameTree::print ( )

prints the tree as evaluation numbers in a breadth first fashion

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

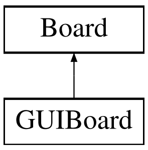
- gametree.h
- gametree.cpp

4.6 GUIBoard Class Reference

gui interface

`#include <guiboard.h>`

Inheritance diagram for GUIBoard:



Public Member Functions

- **GUIBoard** (QWidget ∗w)
- bool sow (int &row, int &col, bool clockwise, int hand)
- void print ()
- void setButton (int row, int col, int val)
- void setHand (int val)
- int capture (int row, int col)
- int take (int row, int col)
- void possibleMoves (bool player)
- void disableAll ()

Additional Inherited Members

4.6.1 Detailed Description

gui interface

4.6.2 Description

Uses functionality of Board class and displays it with the widget

4.6.3 License

Copyright belongs to Juan du Preez (15189016)

4.6.4 Member Function Documentation

4.6.4.1 int GUIBoard::capture (int *row,* int *col*) [virtual]

captures opposite seeds if possible. returns -1 if not

Reimplemented from Board.

4.6.4.2 void GUIBoard::disableAll ( )

disables all buttons

4.6.4.3 void GUIBoard::possibleMoves (bool *player*) [virtual]

makes yellow circles around possible moves

Reimplemented from Board.

4.6.4.4 void GUIBoard::print ( ) [virtual]

updates widget with current values

Reimplemented from Board.

4.6.4.5 void GUIBoard::setButton (int *row,* int *col,* int *val*)

sets single button with single value

4.6.4.6 void GUIBoard::setHand (int *val*)

updates hand label with current hand value

4.6.4.7 bool GUIBoard::sow (int & *row,* int & *col,* bool *clockwise,* int *hand*) [virtual]

helper function: has sow around functionality

Reimplemented from Board.

4.6.4.8 int GUIBoard::take (int *row,* int *col*) [virtual]

removes from single cell in board

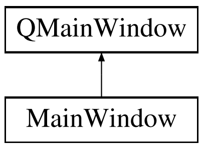
Reimplemented from Board.

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

- guiboard.h
- guiboard.cpp

4.7 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- **MainWindow** (QWidget ∗parent=0)

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

- mainwindow.h
- mainwindow.cpp

4.8 Move Class Reference

a single move that can be made on a current board state

`#include <move.h>`

Public Attributes

- int row
- int col
- bool direction
- bool isTakasaHouse
- bool isNamuaStop

4.8.1 Detailed Description

a single move that can be made on a current board state

4.8.2 Description

This class is used to create the edges of the game tree. One stated leads to another state through a move.

4.8.3 License

Copyright belongs to Juan du Preez (15189016)

4.8.4 Member Data Documentation

4.8.4.1 int Move::col

the current column which is chosen in the move

4.8.4.2 bool Move::direction

the direction row which is chosen in the move

4.8.4.3 bool Move::isNamuaStop

true if the one chooses to stop at the house

4.8.4.4 bool Move::isTakasaHouse

true if the move ends in a takas house situation

4.8.4.5 int Move::row

the current row which is chosen in the move

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

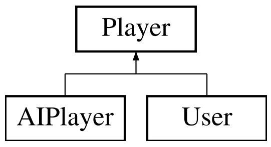
- move.h
- move.cpp

4.9 Player Class Reference

base class for User and AIPlayer

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

Inheritance diagram for Player:



Public Member Functions

- **Player** (int playerNumber)
- virtual bool play (int row, int col, bool direction, Board ∗board)

Protected Attributes

- int playerNumber

4.9.1 Detailed Description

base class for User and AIPlayer

4.9.2 Description

Base class which can be used to implement either the user's or the computer's part in the game

4.9.3 License

Copyright belongs to Juan du Preez (15189016)

4.9.4 Member Function Documentation

4.9.4.1 bool Player::play (int *row,* int *col,* bool *direction,* Board ∗ *board*) `[virtual]`

This provides the functionality of a "User". I should have moved this function to the User class.

4.9.5 Member Data Documentation

4.9.5.1 int Player::playerNumber `[protected]`

1 or 2 based on where one sits with regards to facing the board

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

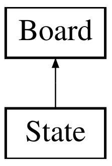
- player.h
- player.cpp

4.10 State Class Reference

Node in Game tree.

`#include <state.h>`

Inheritance diagram for State:



Public Member Functions

- **State** (const Board &other, bool player)
- int evaluate (bool player)
- bool getPlayer ()

Public Attributes

- vector< State ∗ > children
- int evaluation
- bool isMaxNode

Additional Inherited Members

4.10.1 Detailed Description

Node in Game tree.

4.10.2 Description

The node in the game tree displaying the current state of the game

4.10.3 License

Copyright belongs to Juan du Preez (15189016)

4.10.4 Member Function Documentation

4.10.4.1 int State::evaluate (bool *player*)

evaluates the state of the board with regards to a player

4.10.4.2 bool State::getPlayer ( )

returns the favouredPlayer variable

4.10.5 Member Data Documentation

4.10.5.1 vector<**State**∗> State::children

reference to further states

4.10.5.2 int State::evaluation

a variable that keeps the evaluated value. Also used as alpha/beta value of node

4.10.5.3 bool State::isMaxNode

shows whether node is min or max node

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

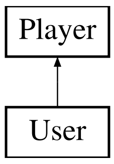
- state.h
- state.cpp

4.11 User Class Reference

the human based part of the game

`#include <user.h>`

Inheritance diagram for User:



Public Member Functions

- **User** (int)
- bool takeTurn (Board ∗)
- int assessState (Board ∗)
- void sowLeftRight (int)
- void capture (int)

Additional Inherited Members

4.11.1 Detailed Description

the human based part of the game

4.11.2 Description

In charge of managing user’s decisions

4.11.3 License

Copyright belongs to Juan du Preez (15189016)

4.11.4 Member Function Documentation

4.11.4.1 int User::assessState (**Board** ∗ *board*)

currently not in use

4.11.4.2 void User::capture (int *x*)

currently not in use

4.11.4.3 void User::sowLeftRight (int *x*)

currently not in use

4.11.4.4 bool User::takeTurn (Board ∗ *board*)

currently not in use

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

- user.h
- user.cpp

Chapter 5

File Documentation

5.1 aiplayer.h File Reference

```
#include "player.h"
#include "gametree.h"
#include "guiboard.h"
```

Classes

- class AIPlayer

  the player which will think for himself or herslef

5.2 game.h File Reference

```
#include "aiplayer.h"
#include "user.h"
#include "guiboard.h"
#include <QWidget>
```

Classes

- class Game

  facade to interact with outside world

5.3 gametree.h File Reference

```
#include "state.h"
#include "board.h"
#include "guiboard.h"
#include "move.h"
#include <vector>
```

Classes

- class GameTree

  the "brain" behind the AI functionality

5.4 guiboard.h File Reference

```
#include "board.h"
#include <QWidget>
#include <QPushButton>
#include <QLabel>
#include <sstream>
#include <unistd.h>
```

Classes

- class GUIBoard

  gui interface

5.5 move.h File Reference

Classes

- class Move

  a single move that can be made on a current board state

5.6 player.h File Reference

```
#include "board.h"
```

Classes

- class Player

  base class for User and AIPlayer

5.7 state.h File Reference

```
#include "board.h"
#include <vector>
```

Classes

- class State

  Node in Game tree.

5.8 user.h File Reference

```
#include "player.h"
```

Classes

- class User

  the human based part of the game

Index