Mate Finder

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Board Class Reference

Public Member Functions

- **Board** (const char ∗str)
- Board (const char ∗str, const bool file)
- Piece::Piece **getSquare** (const square< int > pos) const
- bool **isCheck** () const
- bool **blackToMove** () const
- bool **isMate** () const
- bool **isDraw** () const
- moveArray & **getMoves** ()
- void execMove (const move mv)
- void **changeColor** ()
- void **clearBoard** ()
- void **setPiece** (const square< int > sq, const Piece::Piece piece)
- Board **cloneAndExecMove** (const move mv) const
- void printBoard () const
- void printLegalMoves () const

Private Types

- enum **stateFlags** : char {
  checkMask = 0x01, **whiteCastleKingsideMask** = 0x02, **whiteCastleQueensideMask** = 0x04, **black←↩
  CastleKingsideMask** = 0x08,
  blackCastleQueensideMask = 0x10, **blackToMoveMask** = 0x20, **drawMask** = 0x40 }

Private Member Functions

- int fromStr (const char ∗str)
- int fromFile (const char ∗fileName)
- void calcMoves ()
- void getPieceMoves (moveArray &result, const check &kingEnv, const square< int > curPos, const square< int > kingPos)
- void checkDir (moveArray &result, const check &kingEnv, const square< int > basePos, const square< int > dir) const
- check getCheck (const square< int > kingPos)
- bool firstPiece (check &result, const square< int > curPos, const square< int > dir, const int friendlies) const
- bool **isAttacked** (const square< int > piecePos) const
- bool hasAttacker (square< int > pos, const square< int > dir) const
- bool **isFriendly** (const Piece::Piece piece) const
- bool **isFriendly** (const square< int > pos) const
- void flipBoard ()
- Board (const Board &other, const move mv)

Private Attributes

- Piece::Piece **board** [8][8]
- char **state**
- signed char **enPassant**
- moveArray **legalMoves**

2.1.1 Constructor & Destructor Documentation

2.1.1.1 Board() [1/2]

```
Board::Board (
            const Board & other,
            const move mv )   [private]
```

This constructor is used when a move is executed. All the pieces must be copied, but copying all the legal moves is not necessary, as they are recalculated anyway. So, this private constructur is only to be used internally, when calling the *cloneAndExecMove* function. [in] other This is the board that is to be cloned. [in] mv This is the move that is to be executed.

2.1.1.2 Board() [2/2]

```
Board::Board (
            const char * str,
            const bool file )
```

Board constructor creating Board object from either a file or a string

Parameters

str	Either the FEN string or the filename
file	Boolean whether to read file or string (true is file)

2.1.2 Member Function Documentation

2.1.2.1 calcMoves()

```
void Board::calcMoves ( )  [private]
```

This function calculates all possible moves, and stores them in *legalMoves* member. Additionally, it updates the *check* flag, to indicate whether the player that is to move is check. Finally, it also updates the *draw* flag. If there are too few pieces on the board for any mate to occur, then this flag is set.

2.1.2.2 checkDir()

```
void Board::checkDir (
            moveArray & result,
            const check & kingEnv,
            const square< int > basePos,
            const square< int > dir ) const  [inline], [private]
```

This function helps the getPieceMoves function. It checks to which squares a long range piece (i.e. queen, rook or bishop) can move in a given direction, and appends these moves to the array of legal moves.

Parameters

out	*result*	This is the array of moves which the found legal moves are appended to.
in	*kingEnv*	Through this *check* object, the information concerning whether a given move resolves check is supplied to the function.
in	*basePos*	This is the position of the piece whose legal moves are investigated.
in	*dir*	This is the direction in which the legal moves are being checked.

2.1.2.3 execMove()

```
void Board::execMove (
            const move move )
```

Execute a move on a given board and consequently flip the board and recalculate the legal moves. [in] move This is the move that is to be executed.

2.1.2.4 firstPiece()

```
bool Board::firstPiece (
             check & result,
             const square< int > curPos,
             const square< int > dir,
             const int friendlies ) const  [private]
```

This function helps the getCheck function to investigate the influence of one of the 8 directions on the output of the getCheck function.

Parameters

out	*result*	The check struct that getCheck is working on.
in	*curPos*	The current position that is being investigated.
in	*dir*	The direction in which is being checked whether there are attacks going on.
in	*friendlies*	This value determines the number of friendly pieces that were already encountered. If this is 1, and an attacking piece is encountered, then *result* will indicate that this piece is pinned.

Returns

This function returns a boolean. True is returned, if the square indicated by *curPos* is under attack from the direction indicated by *dir*. Otherwise, false is returned.

2.1.2.5 flipBoard()

```
void Board::flipBoard ( )  [private]
```

This function flips the board. More precisely, it mirrors the position of all pieces along the axis between the 4th and 5th rank. So, for example, a piece on e3 is placed on e6, and a piece on a8 is placed on a1. This function is used to ensure that the player that is to move always plays with its pawns advancing to ever higher ranks.

2.1.2.6 fromFile()

```
int Board::fromFile (
             const char * fileName )  [private]
```

Initialize a Board object from a file containing a string in FEN notation.

Parameters

fileName	The name of the file.

Returns

This function returns an error code, according to the following table:

Code	Description
0	Success.
1	The syntax of str is wrong.

2.1.2.7 fromStr()

```
int Board::fromStr (
             const char * str )  [private]
```

Initialize a Board object from a string in FEN notation.

Parameters

in	*str*	The string in FEN notation

Returns

This function returns an error code, according to the following table:

Code	Description
0	Success.
1	The syntax of str is wrong.

2.1.2.8 getCheck()

```
check Board::getCheck (
             const square< int > kingPos )  [private]
```

This function finds out what is going on in relation to the king of the player that is to move. In particular, it returns a check struct containing information about pinned pieces and squares that can resolve check. Moreover, it updates the check flag.

Parameters

in	*kingPos*	The position of the king.

Returns

This function returns a check object, containing information about pinned pieces and, in case of check, how many pieces are attacking the king, and, in case of 1 attacking piece, which squares will resolve the check.

2.1.2.9 getPieceMoves()

```
void Board::getPieceMoves (
            moveArray & result,
            const check & kingEnv,
            const square< int > curPos,
            const square< int > kingPos )  [private]
```

This function helps the calcMoves function. It investigates the legal moves that one particular piece can make, and appends these to the temporary moves array created by calcMoves.

Parameters

out	*result*	This is the array that the newly found moves are being appended to.
in	*kingEnv*	The function uses this data structure to check whether the piece is pinned, or helps resolving a check.
in	*curPos*	This is the square where the piece, whose legal moves are begin investigated, is located.
in	*kingPos*	This is the square where the king of the player who is to move is located. Its value is used to speed up special cases of taking en passant.

2.1.2.10 hasAttacker()

```
bool Board::hasAttacker (
            square< int > pos,
            const square< int > dir ) const  [private]
```

This function helps the isAttacked function to determine whether an attack over a long distance from a fixed direction occurs.

Parameters

in	*pos*	The position of which is determined whether it is under attack.
in	*dir*	The direction in which is being checked whether there is an attacker.

Returns

This function returns a boolean, indicating whether there is an emeny piece attacking the square *pos* from direction *dir*.

2.1.2.11 printBoard()

```
void Board::printBoard ( ) const
```

Print a formatted representation of the board.

2.1.2.12 printLegalMoves()

```
void Board::printLegalMoves ( ) const
```

Print a list of the legal moves.

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

- src/Board.h
- src/Board.cpp

2.2 check Struct Reference

Public Member Functions

- void **display** ()

Public Attributes

- int **len**
- char **heatMap** [8][8]

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

- src/Board.h

2.3 move Struct Reference

Public Attributes

- square< void > **start**
- square< void > **end**
- Piece::Piece **promoteTo**

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

- src/Board.h

2.4 moveArray Struct Reference

Public Member Functions

- **moveArray** (const int n)
- **moveArray** (const moveArray &other)
- moveArray & **operator=** (const moveArray &other)
- moveArray & **operator=** (moveArray &&other)
- move & **operator[ ]** (const int n) const
- int **size** () const
- void **push_back** (const move toAdd)
- moveArray **shrink_to_fit** ()

Public Attributes

- int **num**
- int **ctr**
- move ∗ **moves**

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

- src/Board.h

2.5 square< T > Struct Template Reference

Public Member Functions

- **square** (const T x, const T y)
- template<typename newT >
  square (const square< newT > &other)
- template<typename newT >
  square< typename std::common_type< T, newT >::type > **operator+** (const square< newT > &other) const
- square< T > **operator+** (const square< void > &other) const
- square< T > & **operator+=** (const square< T > &other)
- square< T > & **operator+=** (const square< void > &other)

Public Attributes

- T **x**
- T **y**

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

- src/Board.h

2.6 square< void > Struct Template Reference

Public Member Functions

- **square** (const int x, const int y)
- template<typename newT >
  square (const square< newT > &other)
- template<typename newT >
  square< newT > **operator+** (const square< newT > &other)
- template<typename newT >
  square< void > & **operator+=** (const square< newT > &other)

Public Attributes

- signed char **x**:4
- signed char **y**:4

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

- src/Board.h

Index