Reversi Game Development

Main Page	Classes	Files	
Class List	Class Members		
sGameS	Status Struc	t Refe	Public Member Functions I Public Attributes I List of all members 'ence

Game status structure that encapsulates the state of the game after the last move. More...

#include <ReversiEngineInterface.h>

Public Member Functions

sGameStatus ()

Struct constructor.

Public Attributes

unsigned int	currentPlayerId Identifier for the current player.		
unsigned int	opposingPlayerId Identifier for the opposing player.		
bool	isGameOver Indicator as to whether game is over.		
bool	isOpponentFinished Indicator as to whether the opponent is finished.		
bool	wasLastMoveValid Indicator as to whether last move was valid.		
unsigned int	gameBoard [REVERSI_BOARD_DIM *REVERSI_BOARD_DIM] Current disc layout on the game board.		
unsigned int	player1Score Score for player 1.		
unsigned int	player2Score Score for player 2.		

Detailed Description

Game status structure that encapsulates the state of the game after the last move.

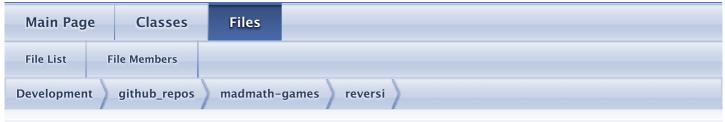
The documentation for this struct was generated from the following file:

• /Users/geoff/Development/github_repos/madmath-games/reversi/ReversiEngineInterface.h

Generated on Wed Nov 5 2014 16:57:35 for Reversi Game Development by



Reversi Game Development



ReversiEngineInterface.h

Go to the documentation of this file.

```
/*!
        \file ReversiEngineInterface.h
 2
        \brief The interface to the Reversi Engine.
 3
 4
        Interface to the 2 player Reversi Engine.
 5
 6
        2014-11-04
                     Geoff Hayes
                                       Initial Release.
 7
 8
    #ifndef REVERSI ENGINE INTERFACE H
10
    #define REVERSI_ENGINE_INTERFACE_H_
11
12
13
    //! Default Reversi board dimension
14
    #define REVERSI BOARD DIM 8
15
16
    //! Game status structure that encapsulates the state of the game after the last
    move.
17
   struct sGameStatus
18
19
        //! Struct constructor.
20
        sGameStatus(): currentPlayerId(0),
21
                         opposingPlayerId(0),
22
                         isGameOver(false),
23
                         isOpponentFinished(false),
24
                         wasLastMoveValid(false),
25
                         player1Score(0),
                         player2Score(0)
26
27
        {
            // intentionally left blank
2.8
29
30
31
        //! Identifier for the current player.
32
        unsigned int currentPlayerId;
33
34
        //! Identifier for the opposing player.
35
        unsigned int opposingPlayerId;
36
37
        //! Indicator as to whether game is over.
38
        bool isGameOver;
39
40
        //! Indicator as to whether the opponent is finished.
41
        bool isOpponentFinished;
42
43
        //! Indicator as to whether last move was valid.
44
        bool wasLastMoveValid;
45
        //! Current disc layout on the game board.
46
47
        unsigned int gameBoard[REVERSI BOARD DIM*REVERSI BOARD DIM];
48
49
        //! Score for player 1.
50
        unsigned int player1Score;
51
52
        //! Score for player 2.
53
        unsigned int player2Score;
54
   };
```

```
55
56
    //! Creates the Reversi Engine instance.
 57
    /**
     * Creates the Reversi Engine instance. If a game is in progress, then it
 58
     * is restarted.
 59
 60
 61
      * @retval true if engine created successfully
 62
     * @retval false if engine not created
 63
 64
    bool createReversiEngine();
 65
    //! Destroys the Reversi Engine instance.
 66
     /**
 67
     * Destroys the Reversi Engine instance. If a game is in progress, then it
 68
     * is terminated.
 69
     */
 70
 71
    void destroyReversiEngine();
 72
 73
    //! Displays the game board on the console (debug only)
 74
 75
     * Displays the game board on the console.
     */
 76
 77
    void displayReversiBoard();
 78
 79
    //! Gets the score for a particular player.
 80
    /**
 81
     * Gets the score for a particular player.
 82
 83
     * @param
                           The identifier for the player.
                 playerId
 84
     */
 85
    unsigned int getPlayerScore(const unsigned int playerId);
 86
 87
     //! Gets the board dimension.
 88
    unsigned int getBoardDim();
 89
 90
    //! Gets the current player identifier.
 91
    unsigned int getCurrentPlayerId();
 92
 93
    //! Gets the game board.
 94
    const unsigned int* const getBoard();
 95
 96
     //! Attempts to apply the player's move on the game board.
 97
    /**
     * Applies a move for the current player.
 98
 99
100
        @param
                 xcoord
                               The x coordinate on the gaming board.
101
                               The y coordinate on the gaming board.
       @param
                 ycoord
102
     * @return The game status.
103
104
105
        @note
                 The x and y coordinates are one-based with coordinate
                 (1,1) the top-left square of the board, (1, REVERSI BOARD DIM)
106
107
     *
                 the top-right square of the board, (REVERSI BOARD DIM, 1) the
108
     *
                 bottom-left square of the board, and (REVERSI_BOARD_DIM,
109
      *
                 REVERSI BOARD DIM) the bottom-right square of the board.
      */
110
    sGameStatus applyMove(const unsigned int xcoord, const unsigned int ycoord);
111
112
113
     //! Indicates whether engine is running or not.
114
    bool isEngineRunning();
115
116
     //! Restarts the game.
117
    void restartGame();
118
    #endif /* REVERSI_ENGINE_INTERFACE_H_ */
119
```