Reversi Game Development

sGameStatus Struct Reference

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.

Definition at line **17** of file **ReversiEngineInterface.h**.

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

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

Reversi Game Development

ReversiEngineInterface.h

Go to the documentation of this file.

```
/*! \file ReversiEngineInterface.h
    \brief The interface to the Reversi Engine.

    Interface to the 2 player Reversi Engine.

    2014-11-04   Geoff Hayes     Initial Release.
*/

#ifndef REVERSI_ENGINE_INTERFACE_H_
#define REVERSI_ENGINE_INTERFACE_H_


//! Default Reversi board dimension
#define REVERSI_BOARD_DIM 8

//! Game status structure that encapsulates the state of the game after the last move.
struct sGameStatus
{
    //! Struct constructor.
    sGameStatus() : currentPlayerId(0),
                    opposingPlayerId(0),
                    isGameOver(false),
                    isOpponentFinished(false),
                    wasLastMoveValid(false),
                    player1Score(0),
                    player2Score(0)
    {
        // intentionally left blank
    }

    //! Identifier for the current player.
    unsigned int currentPlayerId;

    //! Identifier for the opposing player.
    unsigned int opposingPlayerId;

    //! Indicator as to whether game is over.
    bool isGameOver;

    //! Indicator as to whether the opponent is finished.
    bool isOpponentFinished;

    //! Indicator as to whether last move was valid.
    bool wasLastMoveValid;

    //! Current disc layout on the game board.
    unsigned int gameBoard[REVERSI_BOARD_DIM*REVERSI_BOARD_DIM];

    //! Score for player 1.
    unsigned int player1Score;

    //! Score for player 2.
    unsigned int player2Score;
};
```

```

//! Creates the Reversi Engine instance.
/**
 * Creates the Reversi Engine instance.  If a game is in progress, then it
 * is restarted.
 *
 * @retval  true if engine created successfully
 * @retval  false if engine not created
 */
bool createReversiEngine();

//! Destroys the Reversi Engine instance.
/**
 * Destroys the Reversi Engine instance.  If a game is in progress, then it
 * is terminated.
 */
void destroyReversiEngine();

//! Displays the game board on the console (debug only)
/**
 * Displays the game board on the console.
 */
void displayReversiBoard();

//! Gets the score for a particular player.
/**
 * Gets the score for a particular player.
 *
 * @param   playerId   The identifier for the player.
 */
unsigned int getPlayerScore(const unsigned int playerId);

//! Gets the board dimension.
unsigned int getBoardDim();

//! Gets the current player identifier.
unsigned int getCurrentPlayerId();

//! Gets the game board.
const unsigned int* const getBoard();

//! Attempts to apply the player's move on the game board.
/**
 * Applies a move for the current player.
 *
 * @param   xcoord        The x coordinate on the gaming board.
 * @param   ycoord        The y coordinate on the gaming board.
 *
 * @return  The game status.
 *
 * @note    The x and y coordinates are one-based with coordinate
 *          (1,1) the top-left square of the board, (1,REVERSI_BOARD_DIM)
 *          the top-right square of the board, (REVERSI_BOARD_DIM,1) the
 *          bottom-left square of the board, and (REVERSI_BOARD_DIM,
 *          REVERSI_BOARD_DIM) the bottom-right square of the board.
 */
sGameStatus applyMove(const unsigned int xcoord, const unsigned int ycoord);

//! Indicates whether engine is running or not.
bool isEngineRunning();

//! Restarts the game.
void restartGame();

#endif /* REVERSI_ENGINE_INTERFACE_H_ */
```