Reversi Game Development

ReversiEngineInterface.h File Reference

The interface to the Reversi Engine. More...

Go to the source code of this file.

Classes

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

Macros

#define **REVERSI_BOARD_DIM** 8
Default Reversi board dimension.

Functions

bool **createReversiEngine** ()
Creates the Reversi Engine instance. More...

void **destroyReversiEngine** ()
Destroys the Reversi Engine instance. More...

void **displayReversiBoard** ()
Displays the game board on the console (debug only) More...

unsigned int **getPlayerScore** (const unsigned int playerId)
Gets the score for a particular player. More...

unsigned int **getBoardDim** ()
Gets the board dimension.

unsigned int **getCurrentPlayerId** ()
Gets the current player identifier.

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

sGameStatus	**applyMove** (const unsigned int xcoord, const unsigned int ycoord) Attempts to apply the player's move on the game board. More...
bool	**isEngineRunning** () Indicates whether engine is running or not.
void	**restartGame** () Restarts the game.

Detailed Description

The interface to the Reversi Engine.

Interface to the 2 player Reversi Engine.

2014-11-04 Geoff Hayes Initial Release.

Definition in file **ReversiEngineInterface.h**.

Function Documentation

sGameStatus applyMove (const unsigned int xcoord,
const unsigned int ycoord
)

Attempts to apply the player's move on the game board.

Applies a move for the current player.

Parameters

xcoord The x coordinate on the gaming board.

ycoord The y coordinate on the gaming board.

Returns

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.

Definition at line **126** of file **ReversiEngineInterface.cpp**.

bool createReversiEngine ( )

Creates the Reversi Engine instance.

Creates the Reversi Engine instance. If a game is in progress, then it is restarted.

Return values

true if engine created successfully

false if engine not created

Definition at line 14 of file ReversiEngineInterface.cpp.

void destroyReversiEngine ( )

Destroys the Reversi Engine instance.

Destroys the Reversi Engine instance. If a game is in progress, then it is terminated.

Definition at line 36 of file ReversiEngineInterface.cpp.

void displayReversiBoard ( )

Displays the game board on the console (debug only)

Displays the game board on the console.

Definition at line 54 of file ReversiEngineInterface.cpp.

unsigned int getPlayerScore (const unsigned int playerId)

Gets the score for a particular player.

Gets the score for a particular player.

Parameters

playerId The identifier for the player.

Definition at line 69 of file ReversiEngineInterface.cpp.

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;
};
```

```
 55
 56 //! Creates the Reversi Engine instance.
 57 /**
 58  * Creates the Reversi Engine instance.  If a game is in progress, then it
 59  * is restarted.
 60  *
 61  * @retval  true if engine created successfully
 62  * @retval  false if engine not created
 63  */
 64 bool createReversiEngine();
 65
 66 //! Destroys the Reversi Engine instance.
 67 /**
 68  * Destroys the Reversi Engine instance.  If a game is in progress, then it
 69  * is terminated.
 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   playerId   The identifier for the player.
 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 /**
 98  * Applies a move for the current player.
 99  *
100  * @param   xcoord        The x coordinate on the gaming board.
101  * @param   ycoord        The y coordinate on the gaming board.
102  *
103  * @return  The game status.
104  *
105  * @note    The x and y coordinates are one-based with coordinate
106  *          (1,1) the top-left square of the board, (1,REVERSI_BOARD_DIM)
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  */
111 sGameStatus applyMove(const unsigned int xcoord, const unsigned int ycoord);
112
113 //! Indicates whether engine is running or not.
114 bool isEngineRunning();
115
116 //! Restarts the game.
117 void restartGame();
118
119 #endif /* REVERSI_ENGINE_INTERFACE_H_ */
```

Reversi Game Development

ReversiEngineInterface.cpp

```
1  /**
2   * Interface to the 2 player Reversi Engine.
3   *
4   * 2014-11-04   Geoff Hayes     Initial Release.
5   */
6
7  #include "ReversiEngineInterface.h"
8  #include "ReversiEngine.h"
9  #include <exception>
10 #include <cstring>
11
12 ReversiEngine* pReversiEngine = 0;
13
14 bool createReversiEngine()
15 {
16     try
17     {
18         if (pReversiEngine)
19         {
20             delete pReversiEngine;
21             pReversiEngine = 0;
22         }
23
24         pReversiEngine = new ReversiEngine(REVERSI_BOARD_DIM);
25     }
26     catch(const std::exception& ex)
27     {
28         pReversiEngine = 0;
29
30         // log an error
31     }
32
33     return pReversiEngine;
34 }
35
36 void destroyReversiEngine()
37 {
38     try
39     {
40         if (pReversiEngine)
41         {
42             delete pReversiEngine;
43             pReversiEngine = 0;
44         }
45     }
46     catch(const std::exception& ex)
47     {
48         pReversiEngine = 0;
49
50         // log an error
51     }
52 }
53
54 void displayReversiBoard()
55 {
56     try
57     {
```

```
58          if (pReversiEngine)
59          {
60              pReversiEngine->displayBoard();
61          }
62      }
63      catch(const std::exception& ex)
64      {
65          // intentionally left blank
66      }
67  }
68
69  unsigned int getPlayerScore(const unsigned int playerId)
70  {
71      unsigned int score = 0;
72
73      try
74      {
75          if (pReversiEngine)
76          {
77              score =  pReversiEngine->getPlayerScore(playerId);
78          }
79      }
80      catch(const std::exception& ex)
81      {
82          // intentionally left blank
83      }
84
85      return score;
86  }
87
88  unsigned int getBoardDim()
89  {
90      unsigned int boardDim = 0;
91
92      try
93      {
94          if (pReversiEngine)
95          {
96              boardDim =  pReversiEngine->getBoardDim();
97          }
98      }
99      catch(const std::exception& ex)
100     {
101         // intentionally left blank
102     }
103
104     return boardDim;
105 }
106
107 const unsigned int* const getBoard()
108 {
109     if (pReversiEngine)
110     {
111         try
112         {
113             return pReversiEngine->getBoard();
114         }
115         catch(const std::exception& ex)
116         {
117             return 0;
118         }
119     }
120     else
121     {
122         return 0;
123     }
124 }
125
126 sGameStatus applyMove(const unsigned int xcoord, const unsigned int ycoord)
127 {
128     ReversiEngine::GAME_STATUS_TYPE moveStatus = ReversiEngine::INVALID_MOVE;
129
130     sGameStatus gameStatus;
131
132     try
133     {
134         if (pReversiEngine)
135         {
```

```
136             moveStatus = pReversiEngine->applyMove(xcoord,ycoord);
137
138             gameStatus.wasLastMoveValid =
    (moveStatus==ReversiEngine::NEXT_PLAYER ||
139                 moveStatus==ReversiEngine::OPP_FINISHED ||
140                 moveStatus==ReversiEngine::GAME_OVER);
141
142             gameStatus.isGameOver         =
    (moveStatus==ReversiEngine::GAME_OVER);
143             gameStatus.isOpponentFinished =
    (moveStatus==ReversiEngine::OPP_FINISHED);
144             gameStatus.currentPlayerId    = pReversiEngine-
    >getCurrentPlayerId();
145             gameStatus.opposingPlayerId   = pReversiEngine-
    >getOpponentId(gameStatus.currentPlayerId);
146             gameStatus.player1Score       = pReversiEngine->getPlayerScore(1);
147             gameStatus.player2Score       = pReversiEngine->getPlayerScore(2);
148             memcpy(gameStatus.gameBoard,pReversiEngine->getBoard(),
149                    sizeof(unsigned int)*REVERSI_BOARD_DIM*REVERSI_BOARD_DIM);
150         }
151     }
152     catch(const std::exception& ex)
153     {
154         // intentionally left blank
155     }
156
157     return gameStatus;
158 }
159
160 bool isEngineRunning()
161 {
162     return pReversiEngine;
163 }
164
165 void restartGame()
166 {
167     try
168     {
169         if (pReversiEngine)
170         {
171             pReversiEngine->restartGame();
172         }
173     }
174     catch(const std::exception& ex)
175     {
176         // intentionally left blank
177     }
178 }
179
180 unsigned int getCurrentPlayerId()
181 {
182     unsigned int playerId = 0;
183
184     try
185     {
186         if (pReversiEngine)
187         {
188             playerId = pReversiEngine->getCurrentPlayerId();
189         }
190     }
191     catch(const std::exception& ex)
192     {
193         // intentionally left blank
194     }
195
196     return playerId;
197 }
```