Notes on the Reversi Implementation

The implementation for this 2-player game consists of three parts:

1. The C++ Reversi Engine which manages the rules and state of the game (the latter includes, but is not limited to, the layout of the player discs on the board, the player scores, and the player "turn");

2. The C++ Reversi Engine Interface which allows for the creation, communication, and destruction of a Reversi Engine instance.

3. The MATLAB user interface.

Each of the above parts will be briefly discussed.

This software was built and tested on OS X 10.8.5.

Reversi Engine

The ReversiEngine class is implemented in the ReversiEngine.h and ReversiEngine.cpp files. As already stated, it manages the rules and state of the game. It allows for a variable sized square board (height/width dimension is assumed even and at least four).

See docs/ReversiEngineClass.pdf for more details.

Reversi Engine Interface

The interface is implemented in the ReversiEngineInterface.h and ReversiEngineInterface.cpp files. It acts as the gateway between a user interface and the Reversi engine. It provides functions to create the engine, pass player moves (with response), and destroy the engine.

See docs/ReversiEngineInterface.pdf for more details.

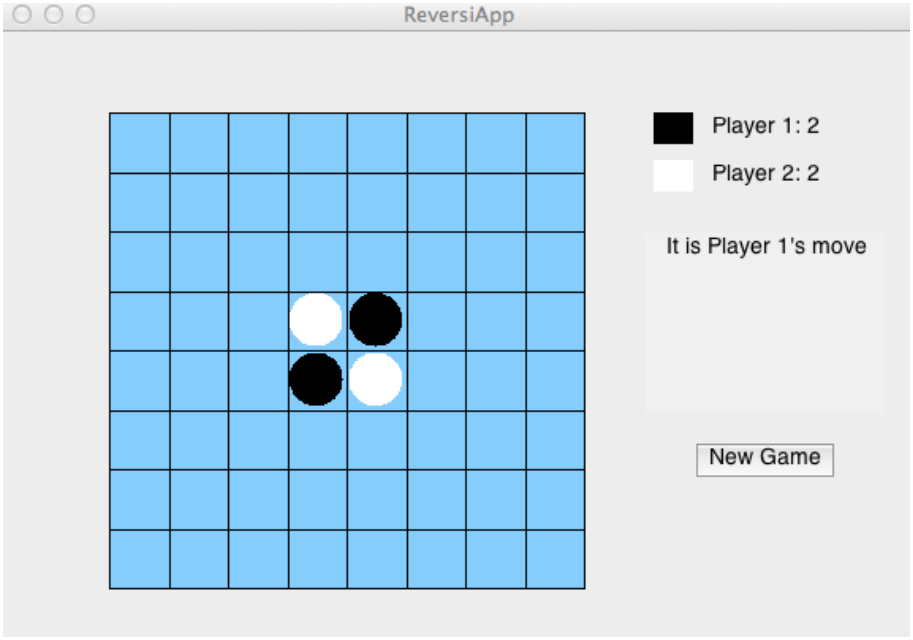
The interface makes use of the struct sGameStatus which is used to provide the response from a players' move back to the caller (UI).

See docs/GameStatusStruct.pdf for more details.

MATLAB User Interface

The MATLAB user interface (UI) allows two players to play a game of Reversi. The UI and code to communicate with the Reversi Engine Interface is found in ReversiApp.fig and ReversiApp.m respectively. As the UI is interfacing with C++ code, it requires its own "gateway" to the interface using a MEX function, mexReversiEngineInterface.cpp.

The (simple) UI is



Building Software

To validate the software, the code is compiled with **gcc** (version 4.2.1) to create a simple executable that validates some of the functionality as

```
gcc -o reversi reversi_dev_main.cpp ReversiEngine.cpp ReversiEngineInterface.cpp -lstdc++
```

(the above is assumed to have been executed in the directory that contains all source files). The **reverse** executable uses the interface to create

the engine, send requests, and finally destroys the engine.

A similar "executable" exists for MATLAB (version R2014) which builds the MEX function and runs through a similar suite of tests. In the MATLAB Command Window, in the directory that contains all source code, the function to test the code is called as

```
reversi_dev_mex_main
```

This function builds the MEX function which can also be built separately as

```
mex mexReversiEngineInterface.cpp ReversiEngine.cpp ReversiEngineInterface.cpp -lstdc++
```

A file similar to mexReversiEngineInterface.mexmaci64, will be built.

Running the UI

Assuming that the MEX function has been built (either separately or with reversi_dev_mex_main), the UI can be launched (from the directory with all source code) as

```
ReversiApp
```