Lab Assignment 3

CS 361 – Principles of Programming Languages I

Fall 2017

The goal of this lab is to program the game Connect Four. See [1] if you are unfamiliar with the game.

You are already given a file *main.cpp* (which you may not change). You have to implement the remaining of the game. Next to the *main.cpp*, your program should contain the files *game.h*, *game.cpp*, *players.h*, and *players.cpp*.

The file *game.h* should define an enumeration *GameState* which represents the possible game states, an enumeration *BoardField* which represents the possible state of a field of the game-board, and the class *Game*. The file *players.h* should define an abstract class *Player*, a class *HumanPlayer*, and a class *AiPlayer*. The classes *HumanPlayer* and *AiPlayer* should be subclasses of the class *Player*. The header files should only contain declarations. All class members should only be implemented in the corresponding *.cpp* files.

Player Classes

Player. The abstract class *Player* defines the interface which is used by the *Game* class. It should contain a pure virtual function *getNextTurn* which takes as parameter the current game (to read the game state and board) and return an integer in the range from 1 to 7 representing the column in which the player puts its next stone.

HumanPlayer. Represents a human player. Its function *getNextTurn* should read an integer from the terminal and return it. It may not create any output!

AiPlayer. Represents a non-human player. Its function *getNextTurn* should determine in which column the player can add its next stone. Then, out of all possible columns, one is picked randomly and returned. It also outputs the selected column together with a newline) to the terminal, i. e., if column 3 is selected, the output should be `"3\n"`.

Game Class

The class *Game* contain the logic for the game. It determines if turns of players are valid, keeps track of the board, determines if a player has won, and updates its state accordingly. The class should have the members listed below. Feel free to add additional private members.

`BoardWidth.` A public constant with the value 7.

`BoardHeight.` A public constant with the value 6.

`Game(Player &p1, Player &p2).` Public constructor which takes the players of the game as arguments

`getState().` A public function which returns the current state of the game.

`isRunning().` A public function which returns *true* if the game is still running, or *false* if the game concluded with either a draw or a player winning.

`operator() (int x, int y) const.` Publicly overrides the ()-operator. Returns the state at the board at the given coordinate. The top left element has coordinate $(0, 0)$. Returns *Empty* if the coordinate is out of range of the board.

`nextTurn().` Public function which performs the next turn. To do so, the *Game* class calls the function *getNextTurn* of the current player. If the return value is invalid, nothing should happen. Otherwise, the function should process the move of the player. That is, it should update the board, determine if the player won or if a draw was reached, and update the game state accordingly.

Submission

For your submission, upload a single zip-file to canvas. The zip-file should contain

- the files *game.h* and *game.cpp*, and
- the files *players.h* and *players.cpp*.

This is an individual assignment. Therefore, a submission is required from each student. Your implementation is expected to work with Visual Studio 2012 (the version on the lab computers).

Deadline: Sunday, December 3, 11:59 pm.

References

1. Wikipedia, *Connect Four*, Wikipedia, The Free Encyclopedia, `https://en.wikipedia.org/Connect_Four`.