CMPT 212 (2008-1) Assignment 4*
Due online Monday, April 7, 2008, 11:00pm

Ján Maňuch
jmanuch@sfu.ca

1 Programming Assignment

Write an MFC GUI program that allows two players to play an extended version of Tic-Tac-Toe game.

Your code can be located in any number of files (that means you can have several .cpp, .h, .rc and image files) and does not have to be located in one directory. When submitting your assignment include the whole directory containing your project, but excluding `Debug` subdirectories and the .ncb file. This will simplify the evaluating process significantly, as to compile your program, it will be enough to open your .sln file. (Make sure that you submit only one .sln file!)

You must submit a .zip file such that the TA can follow these steps to run your program:

1. Decompress the contents of the .zip file to any folder.
2. Go to that folder and open the .sln file in MS Visual Studio C++ 2005.
3. Compile the program (F7).
4. Run the program (Ctrl-F5).

It is recommended that you test the above four steps using your .zip file before handing it in to ensure that it works. Specifically, when you .zip your project folder, make sure that the structure of subdirectories within your project folder is preserved in the .zip file, so that files are placed in their proper folders when the .zip file is decompressed. If your program cannot be run by following the above four steps, you will be penalized up to 10% of the accuracy part.

Information about evaluating and grading the assignment can be found on the course website.

If you have any questions about the assignment, do not hesitate to send me an email. I might revise the text of the assignment based on your questions if I find out that some parts are difficult to understand or insufficiently specified. If you have troubles with implementing the assignment, you are welcome to see me or TA during office hours.

Remark: *Your program should implement all controls specified here and they have to act exactly as specified here to get full marks for this assignment.* You can add additional controls and graphics, if you want (additional functionality can help you to win the contest for the "best design", see Section 3).

2 Problem Specification

Your task is to implement an MFC GUI program that allows two players to play an extended version of Tic-Tac-Toe game.

* *Revision* : 1.2 (Wednesday 2nd April, 2008,16:45); Details in this document may change before the date of the submission of the assignment. Please make sure you have the latest version.

2.1 The rules of Tic-Tac-Toe

- The game board is a N by M grid of symbols, which is initially empty. The N and M are set by the user of the program in a simple `Dialog` windows. The dialog should not allow user to set N and M to smaller value than 5 and larger value than 20.
- There are two players, named Player X and Player O. Player O goes first.
- A player moves by selecting an empty square on the board.
- A symbol representing that player (i.e., an `'X'` or an `'O'`) is placed on the selected square, and it becomes the other player's turn.
- The game is won if one player gets **five** symbols in a row (horizontally, vertically, or diagonally).
- The game is a draw if all the squares become occupied, yet no player has won.

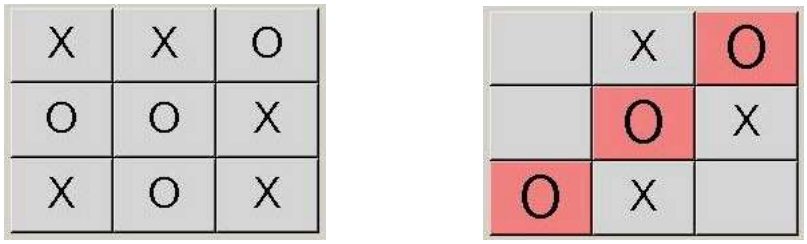

Figure 1: Examples of the display at the end of the game: a draw (left), Player O wins (right) assuming that the user selected $N = M = 3$ when the program started.

2.2 Description of required windows and controls

- Before the main window is open, the program should open the dialog allowing the user to specify the size of the grid of cells (the numbers of rows and columns). It should not allow the user to specify more less than 5 or more than 20 rows (columns).
- The main window of the application should contain a two-dimensional grid (the size is specified by the user of your program) of cells. Each cells can have 3 states: empty, marked with an `'X'` or marked with an `'O'`. Each empty cell should react to the left button mouse clicks on which it will change its state to `'X'` or `'O'` depending on whose player turn it was.
  You can implement cells with ordinary buttons, or you can use drawing functions of the display context and the mouse message handler. If you implement cells with `CButton`-s, you can use its method `SetWindowText("X")` to change the text displayed by the button.
- Your program should issue a warning whenever player tries to play a square which is already occupied with an `'X'` or `'O'`.
- The main window should clearly show which player's turn it is.
- The main window should have a button `Restart`, which would reset the game to initial state (all cells are empty). It does not need to allow the user to change the size of the grid, but it can, if you want.
- Your program should check after each move if a player who just marked a cell has won or not and if he/she won, it should announce the winner, and if all cells have been filled, it should announce the draw.
- Hitting the "Escape" key should bring up a message box with three buttons: "Restart", "Exit" and "Cancel" and the message:

```
Please click ''Restart'' to start again,
''Exit'' to quit the application and
''Cancel'' to continue.
[Restart] [Exit] [Cancel]
```

and clicking on any of these buttons should perform the corresponding task.

2.3 Hints

In the implementation where each cell of the grid is implemented as a **CButton**, you need to assign a different ids to those **CButton**-s. Now, adding 20 times 20, that is 400 message handlers for each button would be no fun. Luckily, there is a macro **ON_COMMAND_RANGE**, which can be used in the message map to associate the whole interval of messages (**int**-s) to one message handler. So, let's see how it works.

- Assume that we have 100 **CButtons**-s in the main window and their ids are $5000 \ldots 5099$.
- We add the following line to the message map of the main window:

```
ON_COMMAND_RANGE(5000,5099,OnButton)
```

- Next, we need to provide code for member function **OnButton()**:

```
CMainWindow : public CFrameWnd {
//...
  afx_msg void OnButton(UINT id)
  {
    // a button with id 'id' was pressed
    // do something about it
  }
};
```

that is, the message handler takes one parameter: the id of on of the 100 buttons which was pressed.

You can use the following start-up project to start your application: a4.zip.

3 Programming Contest

Assignment 4 will have an associated programming contest. Only assignments that are considered to have adequate implementations, as outlined in Section 2 will be entered into the contest.

The winners will be entries that result in the *best design of the graphic interface — the game display*. Note that the specification of the assignment does not exactly specify the position of controls in the windows, or the way how to display cells. These are all parts of the design and will be evaluated in the contest. In addition you can implement some additional features, for instance: the menu, ability of resizing the grid, scrolling, changing the speed, recording the statistics, etc. to improve your chances in winning the contest. The design will be judged independently by instructor and TA (each of us will assign points between 0 and 10 to each program and the final contest score will be the sum of those points).

Up to three entries will be selected. In case of ties that result in more than three top entries of equal size, none of the tied entries will be considered.

The winners will be publicly acknowledged on the course website and during lecture. The source code of winning assignments will be uploaded to and linked from the course web site. Students who are not comfortable with these conditions should indicate that through email at time of submission of their assignment and will not be considered for the contest.