GG250 Lab 13

We shall play a game of tic-tac-toe

X		
X	O	O
O		

Strategy session

- Objective is to <u>get three in a line first</u>
  - Any row, column, or diagonal will do
- If your next move cannot <u>win</u> the game, make sure you <u>block</u> your opponent from winning
  - What hurts the opponent may be good for you
- Looking ahead to the next move is a must; looking ahead to future moves is much harder and may require a recursive approach. You can score 100% on this lab without recursion.

The Game Board

- Are some places on the board more important than others?

3	2	3
2	4	2
3	2	3

Weighting slots differently

- Since some slots can appear in more winning configurations than others, we may consider giving them different weights:

4	2	4
2	8	2
4	2	4

Lab 13 assignment

- Write a function called name_move.m which simulates how you will play a game of tic-tac-toe.
- You can test your strategy against a person (manually selecting the moves) or against a well-intentioned but clueless monkey.

Definition of name_move.m

```
function [row, col] = name_move (board, mover)
% NAME_MOVE          The strategic moves of NAME
% [row, col] = name_move (board, mover)
% Input:           board     A 3x3 matrix representing the board game.
%                            A value of 0 means unused, whereas -1 and
%                            +1 are cells occupied by players 1 and 2
%                  mover     The ID (-1 or +1) of this player.
% Output:          row, col  The position on the board that you have
%                            decided to move to.  This means that
%                            board(col,row) must equal zero.

% If your ID = mover then the other guy's ID = -mover
% To check the status of the board you may use the function
% [game_over, winner] = tictactoe_game_status (board), where
% game_over is 0 or 1 and winner is -1 or 1 if the game is over.
```

The Random Game

- One of the pre-programmed players in the tictactoe.m game is called 'monkey'.
- This player has no strategy at all!!!
  - The only consideration is to find an unused slot on the board for the next move.
  - No analysis of the current situation is undertaken
  - There is no looking ahead to see what a particular move might accomplish

Implementation of monkey_move.m

```
function [row, col] = monkey_move (board, mover)
% MONKEY_MOVE      Simian Simulation
% The monkey randomly picks one of the available slots on the
% board.  No analysis goes into this choice. The board is only
% consulted to find open slots.  The mover ID is not used.

[rows, cols] = find (board == 0);  % Find all the unoccupied slots
n = length (rows);                 % How many such slots are there
choice = floor (rand (1) * n) + 1;  % Select one of them at random
col = cols(choice);                % Get its column value
row = rows(choice);                % Get its row value
```

Checking the board

- Since Matlab allows you to address rows or columns using indices you can check if making a certain move will produce a win or not.
- You don't have to worry about changing the board since the board you see is a local copy of the game board. The updating of the game board is done by the tictactoe.m function.

Examining a single row

Q: For a certain row to give you a win as a result of your upcoming move, what conditions have to be satisfied?

A: You must already occupy 2 of 3 spots, and the 3rd must be empty

Q: How can you check if, say, the 3rd row satisfies this condition?

A: Perhaps write a subfunction that returns the winning position or 0 if there is no win

```
function pos = checkrow3 (board, mover)

pos = find (board(3,:) == 0);

if length(pos) ~= 1 | sum (board(3,:)) ~= 2*mover

        pos = 0;

end
```

Examining rows and columns

- Seems silly to have a separate function for each row (checkrow1, checkrow2, checkrow3) - why not just pass the row number to the function?
- How about checking columns. Any ideas?
- What to do with those two diagonals?