Teaching Announcements 2011/12

< Previous Next >

COMP327 - PRACTICAL ASSIGNMENT 1 2011 28/10/2011

The aim of the first assignment is to demonstrate an understanding of the Model View Controller design paradigm, and apply it by creating an iOS iPhone app. The app should be a Noughts and Crosses (*Tic Tac Toe*) application using the iPhone API. Examples of the final version are given below. If you are unfamiliar with the game (which is also known as "*Noughts and Crosses*", then visit the Wikipedia web page.

Your solution should utilise a ***ViewController*** to manage the main view, with an associated .xib file in which you design your interface (i.e do not add your interface elements to the MainWindow.xib file). The *game-tokens*, or, *counters*. which indicate each play, should be generated from the following images:

- nought.png
- cross.png
- button.png

The last image can be (optionally) used as a background to the "Reset Game" button (size 76x76 pixels). These images should be included as part of your project.

The Model for the game should be realised by creating an instance of a `NCGameModel` model class, which can be obtained from the two files described here.

The game should allow a player to play against the device, i.e. to take it in turn to select a space on the 3x3 playing area, and position a *game-token or counter*. The game should proceed until one player wins, or there are no spaces left.

NOTE: You should not develop your own AI engine; either create a random more for the device, or use the `getOptimalMoveForPlayer:` method provided by the Model.

Detailed Requirements

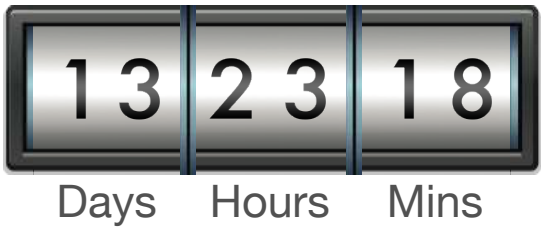
The full requirements are given (below):

1. Each game should alternate between starting with either noughts or crosses. A game can be reset by pressing a "Restart Game" button (which can also be used to alternate the starting player). This should be irrespective of who won the previous game.

2. A status label beneath the main playing area should provide information to the players. In particular, it should:
   - Indicate when the human player should start the game.

Deadline:

4pm Friday 11th Nov '11



A PDF (printable) version of the assignment is also available

Example Output

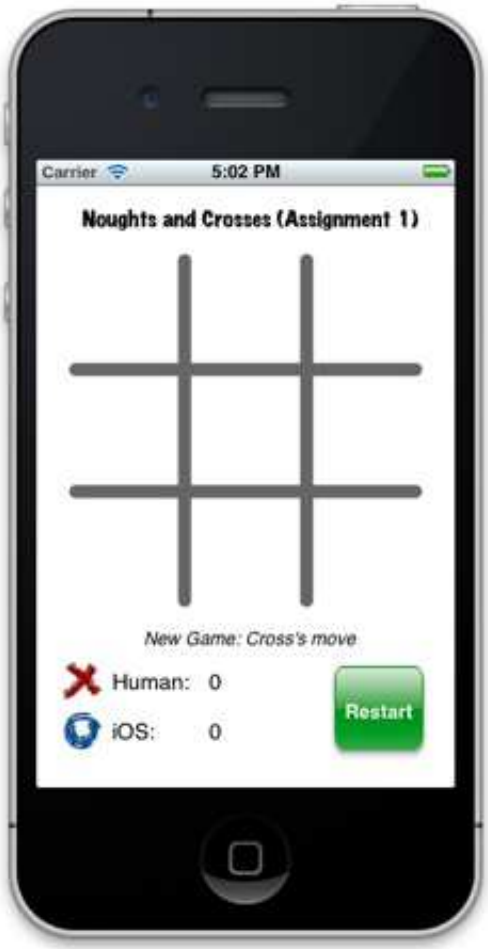
- Indicate the winner when the game has been won.
- Warn that an illegal move has been attempted (such as the user selecting a space already occupied by a *game-token.*
- Indicate if a stalemate condition has occurred.

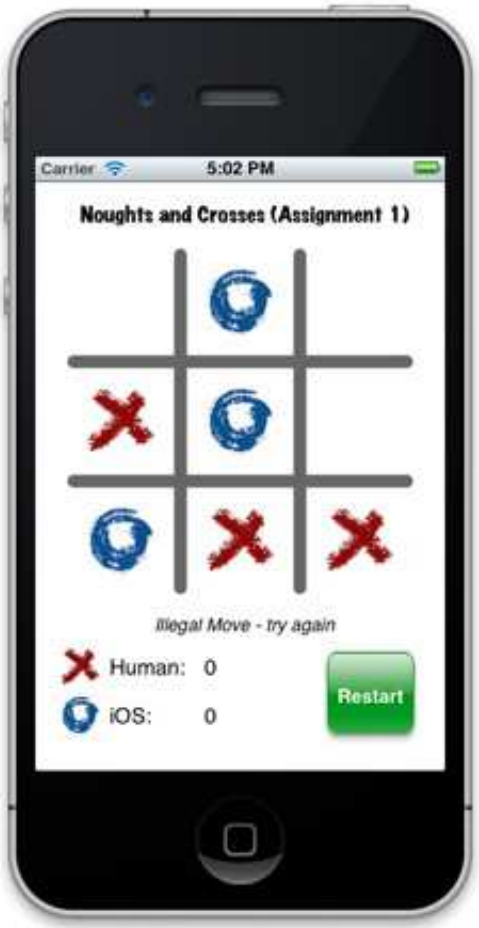
3. A *game-token* should appear on the board at the position where the finger was lifted. This should be either the nought or cross image, depending on which player made the move. This image should be position the images manually, using a **CGContext** (see below) onto a view. **Do not use** UIImages on the board.

4. A score should be kept to indicate how many games each player has played. This should be incremented after each game where one player won.

You should implement a UIView class, called **GameAreaView** that is responsible for displaying the board and counters. The game itself should be managed using a ViewController instance, and the game model should be handled by the (supplied) Game model.

You can use any version of Xcode 4 (including the recently released XCode4.2) running under any version of iOS 5. **However, you should not use any of the new features introduced in iOS5,** as the code should still run on an iOS 4.3 device.





Marking Scheme

The purpose of this assignment is to demonstrate that you know how to develop native iPhone app using the ***Model-View-Controller*** design pattern. You do not need to hand in any design documentation but your code **MUST** be well commented so that it explains what is happening in the code. The following marking scheme will be used to assess each submission:

- **View Controller:** (40 marks total)
  - ***Demonstrate an understanding of using IBOutlets and IBActions, by drawing the main view, including an instance of the GameAreaView, status label, and "Restart Game" button, and managing this using a view-controller (10 marks)***
  - ***Capturing touch events and positioning the correct game-tokens in the Game Model, including detecting illegal moves (10 marks)***
  - ***Detecting the end of a winning game, and congratulating the winner (10 marks)***
  - ***Managing the device to make a move (10 marks)***

- **GameAreaView:** (30 marks total)
  - ***Draw the game lines, and counters on the board based on the status of the Game Model. (20 marks)***
  - ***Draw a line representing the winning line if a game is won (10 marks)***

- **General:** (30 marks total)
  - ***Manage both event locations and drawing the view by calculating positions relative to the view size (10 marks)***
  - ***Maintaining scores for each player and alternating the starting player(10 marks)***
  - ***Layout and Design (10 marks)***

Hints

Most of what you need has already been covered in the labs or lecture notes. The following provides a few additional hints for this assignment. Other hints are also included in the Game Model header file.

1. Images can be stored as resources within a project, and then accessed locally by the project without using a file system (which can be problematic on the iPhone) or accessing them from the web. The following code fragment illustrates how an image could be loaded from a resource, and then displayed on a view:

```
UIImage *crossImage = [UIImage imageNamed:@"cross.png"];
CGPoint myPoint;
...
[crossImage drawAtPoint:myPoint];
```

2. A good solution will always avoid hard-coding locations in views. Think about how the board is partitioned with respect to the bounds of the view. Write methods that convert touch locations into positions, and counter positions into locations, that determine the size of the view by querying its bounds. Test this by resizing your view and seeing if your code still works.

This assignment contributes 10% to your overall mark for COMP327. The code should compile without warning or errors. In addition, memory should be managed appropriately.

SUBMISSION INSTRUCTIONS

Firstly, check that you have adhered to the following list:

1. Your project should be self contained within a single zip file containing all of the files in the XCode project. The file's name MUST be 'NoughtsCorsses.zip'. Ensure that all of your source files are included in the project (hint, unpack it in a different directory and check it still builds).
2. Your project is developed within XCode, not some other language or environment.
3. Your program compiles and runs on a machine within the computer science department's Mac Lab, either under Xcode 4.2, using the iOS SDK 5 (recently released) or an earlier version. If you have developed your code elsewhere (e.g. your own mac), ensure that it also works on our system before submission. It is your responsibility to check that you can log onto the department's system well in advance of the submission deadline.
4. Your program does not bear undue resemblance to anybody else's! Electronic checks for code similarity will be performed on all submissions and instances of plagiarism will be severely dealt with. The rules on plagiarism and collusion are explicit: do not copy anything from anyone else's code, do not let anyone else copy from your code and do not hand in 'jointly developed' solutions.

To submit your solution you must **SUBMIT IT ELECTRONICALLY**, and adhere to the following instructions:

Electronic submission:

- Your code must be submitted to the departmental electronic

submission system at: http://cgi.csc.liv.ac.uk/cgi-bin/submit.pl?module=comp327

- You need to login in to the above system and select ‘Assignment 1’ from the drop-down menu. You then locate the file containing your program that you wish to submit, check the box stating that you have read and understood the university’s policy on plagiarism and collusion, then click the ‘Upload File’ button.

Work will be accepted only if it is submitted electronically following the above instructions.

Finally, please remember that it is always better to hand in an incomplete piece of work, which will result in some marks being awarded, as opposed to handing in nothing, which will guarantee a mark of 0 being awarded. Demonstrators will be on hand during the COMP327 practical sessions to provide assistance, should you need it.