**CIS6930 (07FH, 07F4 & 07GC) & CIS4930 (07G8)  
Mobile Platforms & Development Environments**

**Spring 2013**

**Goban**

**Rajasana Wilhoit, Preston Mueller, Shiraj Sood & Nick Yoon**

**The Idea & Motivation**

*Go* is an ancient Chinese board game that is beloved by many for it’s elegant rules and thought-provoking gameplay. Unfortunately the mobile options for playing *Go* are rather slim. Imagine you want to play *Go* with a friend at a coffee shop. You have to carry the board, which may be several feet, along with a large number of black and white stones for the two players. You set the board up on a table and begin to play. However, a passerby accidently bumps into your table and pieces are jostled about and their positions lost. Now instead imagine you simply pull out your iPad and have access to virtual game of *Go*, where making a move is as simple tapping the spot on the board you would like to play your stone. This is **Goban**. With Goban you can play Go with a friend just about anywhere.

With Goban, we focused on two things: Having a real Go implementation and embracing the “pick up and play” mentality (and the resulting simplicity). There are many subtleties to the rules of Go, so we invested a lot of time making sure your experience playing on iPad would be just like if it were a real board. We also wanted to avoid features that took away from gameplay, so there is no registration or setup required — the app serves a single purpose: to play Go anywhere.

**App Design**

Staying true to our goal of simplicity, our UI is very minimalistic and our UX is easy to follow.

Like all iOS apps, you start Goban by tapping on the easily recognizable app icon.



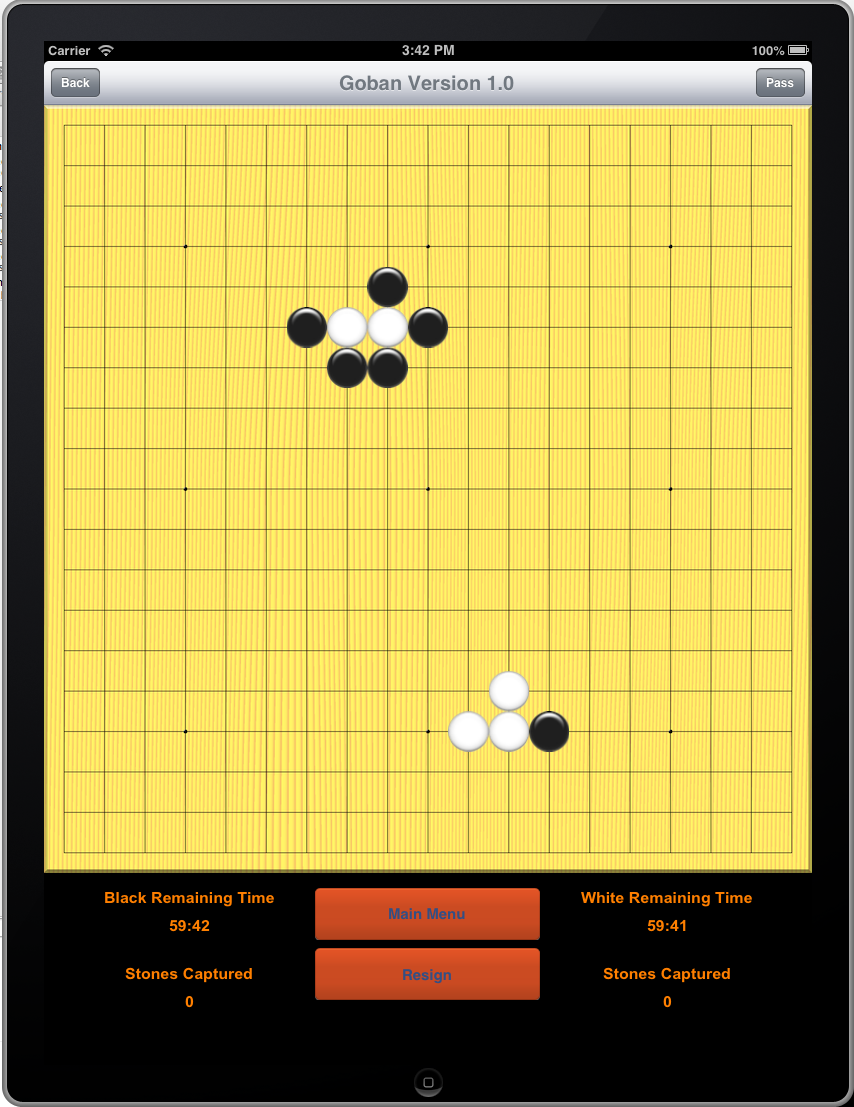
Upon entering the app, you are greeted by a nice looking modal popup with a brief explanation of the app and how to get started.



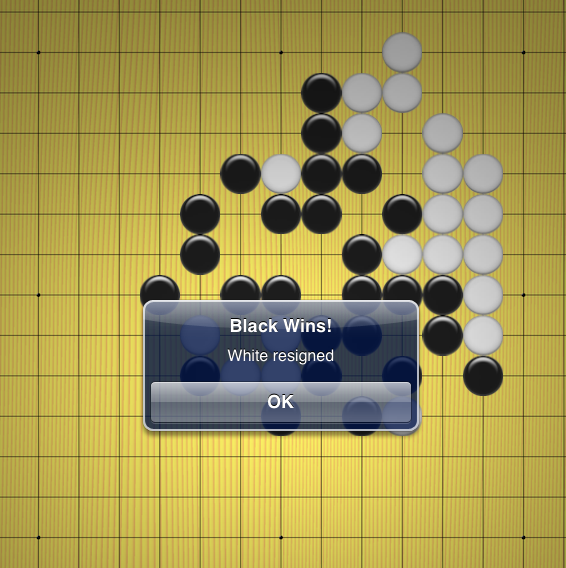
Just as the modal says, there are two ways to get started playing. New Game speaks for itself, and Load Game will pull the last game you played from the server to be resumed.



Upon making your main menu selection, you are transitioned into the game itself. A live game board is now displayed, along with the stats and option buttons at the bottom. To make a turn, a player simply taps on the game board grid where they would like to play. If the move is valid, a stone for their color (either black or white) will appear at the grid intersection they tapped. Turns alternate between players with black getting the first turn.



Because touch interaction is not always highly accurate, we have built into the app the ability to take back the previous move. So if you accidently place a stone in the wrong place, you can simply tap the back button in the top left. To pass on your turn, you tap on the button in the top right. If a player feels they cannot win and the game is over, they may resign with the option button at the bottom of the screen. After which the game is over and they can return to the main menu to start a new one.



**Implementation**

The app was developed starting with the backend. I tested the backend functionality mainly by using log statements. I developed a BFS algorithm to search through the stones adjacent to new moves to check if any stones died. In addition, I wrote numerous algorithms to check if moves were legal.

The board and stones are all drawn on using PNG images. The board is drawn on start and the stones are drawn when the user touches the location on the board. At first I tried putting buttons underneath the board to get the location that the user touched, but that turned out to be too much overhead for the editor, and I instead developed an algorithm that calculated the location that the user touched on the board and then converted that to graph coordinates. In addition, I had the board and all the stones being redrawn each time the user made a move; however, I changed this to only redraw the board when any stones died, since it was much easier to redraw all of the layers of the board instead of deleting specific layers. When there are many stones on the board it can take up to a second to redraw everything, but since the redraw only occurs when something dies, it doesn’t feel like there is much lag to the player, and I was satisfied with this implementation.

Other features such as the game clock, the resign button, and the counters for dead/captured stones were added only after the back end and most of the UI was done. I also used some button templates to draw the buttons.

**Network Resources Used**

To facilitate game saves we utilized the client-server model with the iPad app as the client and a custom Node.js app deployed to the cloud as our server. The basic idea is that the iPad app serializes the object that stores the state of the game when there is a significant change. It will then make an HTTP request to the Node.js server to send this serialized form to be stored under a unique id. The iPad app saves this unique id so that the game can later be requested from the server and loaded.

The Node.js server makes use of the Express library, which gives a basic framework for request routes and responses. For the database, MongoDB, which is a schema-less data store, is used. MongoDB’s flexibility in what can be stored helps tremendously for when we need to store different data, and it also creates the unique id’s mentioned above for retrieval of saved games. The actual data saved are all the values needed to re-load a previously played game. This list includes values such as the state of the board pieces, the number of captures the players have made, the time each player have left, etc.

For server hosting we deployed the Node.js app to the cloud using Heroku (heroku.com).

**Project Status**

The project is in a very workable state. The rules of Go are implemented, so you can actually play the real game with a friend (or play both sides yourself). The player timers are working (if they run out, that player loses), and if a player feels that they up against an impossible situation they can resign. The back button (in case of touch mistakes) is also implemented. As well the automatic saving to the cloud server is implemented, and the re-loading of the last game state (from the main menu) so that games can be played in multiple sittings. The main things not implemented are the win conditions, so the pass button currently does nothing and there is no end game scoring algorithm to determine the winner.

**Materialized Team Effort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Idea | Design | Implementation | Reporting |
| Rajasana Wilhoit | 70% | 65% | 65% | 50% |
| Preston Mueller | 30% | 25% | 35% | 50% |
| Shiraj Sood | 0% | 0% | 0% | 0% |
| Nick Yoon | 0% | 10% | 0% | 0% |