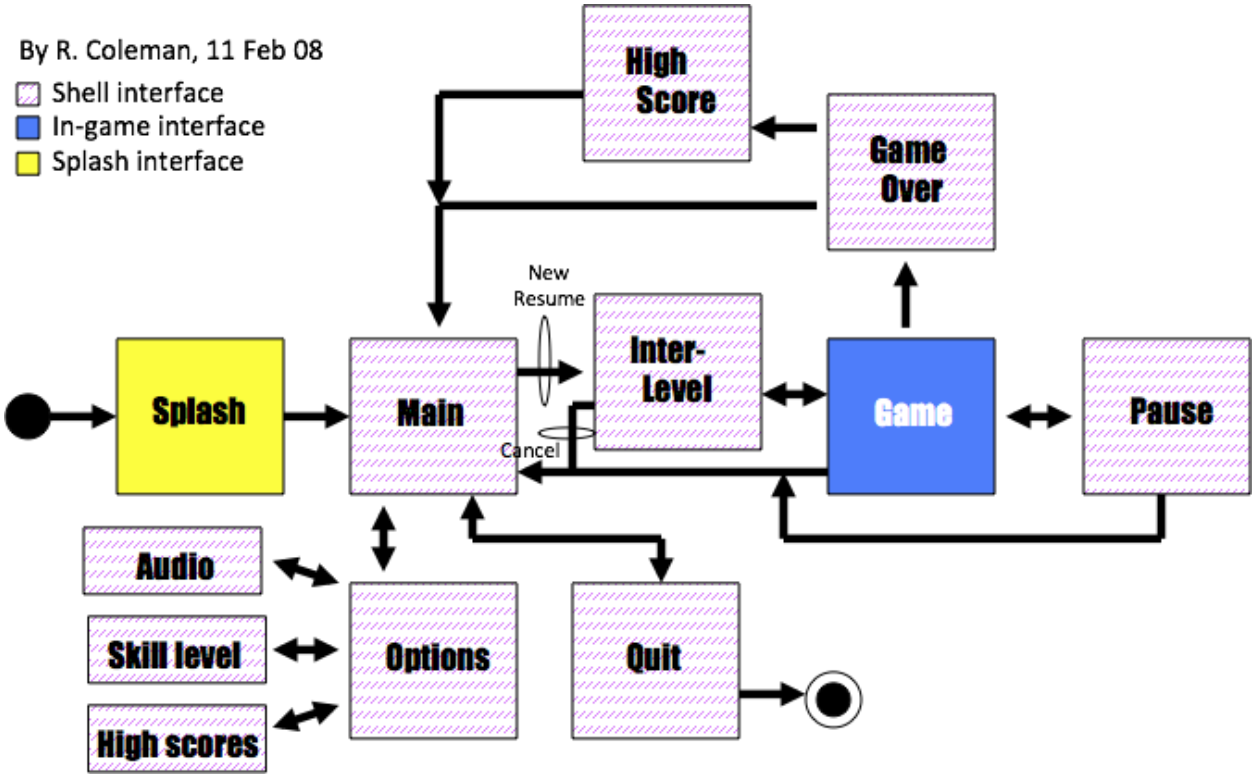
Assignment 2

The goal of this assignment is to implement the shell screens of the screen flow using Xcode, Obj-C, and Cocos2d 2.x.

Note: Please do not use 3.x as it is incompatible with codes I will be giving you.

Background
Figure 1 below shows the screen flow. It shows how the different screens are related to one another. Note the Game screen is where the gameplay occurs while the other screens for a "shell" around this screen.

Figure 1. Game screen state diagram



Prior to coding, review the latest revision of the "Google Objective-C Style Guide" which you can find online.

Tasks
Part I
Install Cocos2d 2.x.

1. Go to http://www.cocos2d-swift.org/download and download "Cocos2D zip archive 2.1.0" to a temporary directory.

2. Double click the `tar.gz` file which creates a folder, `cocos2d-iphone`.

3. Open a terminal and cd to `cocos2d-iphone`. In there run

   ```
   ./install-templates.sh -f
   ```

Part II
Install and test iBall.

1. Make a directory, `games`.

2. Cd to `games` and download iBall from GitHub using the following command:

   ```
   git clone git://github.com/roncoleman125/iBall.git
   ```

3. Cd to iBall, double-click on `iBall4.xcodeproj` to open the project.

4. Run the app as Product | Run.

Part III
Start your game coding.

1. Create a new Cocos2d 2.x project and give it a name relevant to your game.

2. Create a new group, Screens.

3. In the Screens group, create each of the screens in Figure 1 as a subclass of `CCLayer`. Follow the patterns in iBall.

   Do NOT create Splash, Game, and Interlevel. You create these screens in a subsequent assignment. For instance, when the player presses the "New" button, the code does not react.

4. Make each screen go "forward" on button click and "back" with a back button. Otherwise, the screens are stubs to implement the screen flow.

5. In AppDelegate.m, around line 98 comment out,

   ```
   [director runWithScene: [IntroLayer scene]];
   ```

and insert beneath it

```
[[CCDirector sharedDirector] runWithScene: [MainScreen scene]];
```

6. In AppDelegate.m, near the top, comment out the line,

```
#import "IntroLayer.h"
```

and insert beneath it,

```
#import "MainScreen.h"
```

These last two steps are needed to connect the screen shell to the app MVC.

Screen details are in the table below.

Table 1.

Screen	Purpose
MainScreen	This screen contains the main menu with New, Resume (if possible), Options, and Quit buttons. **Note: the Resume button is NOT shown if the game is not being resumed.**
Options	This screen contains three options, Audio, Skill level, and High scores.
Audio	This screen allows the player to toggle all sounds.
Skill level	This screen allows the player to set Beginner, Intermediate, and Advanced levels.
High scores	This screen allows the player to set whether high scores are recorded locally or online. **Note: this screen is different from the High Score screen (see below).**
Game over	This screen presents "game over" to the player with the game score.
High score	This screen is shown only if the game over score is greater than the lowest score on the leaderboard. In this case, the screen allows the user to enter their name and the screen adds this info to the leaderboard. We'll study how to persist score in a subsequent assignment.
Pause	This screen allows the player to pause and resume or quit and go back to the main screen.
Quit	This screen allows the player to confirm quit, that is, go back to the main screen or terminate the app.

Part IV

1. Go over the code style. See “Google Objective-C Style Guide”.

2. Remove all compiler warnings and errors.

3. Run your project and test the screen flow.

4. Zip the project folder and upload it into the assignment shell.

5. Go to another machine, download the project, and test it.

Deliverables

Upload the zip of the project into the assignment shell. Make sure you add the team member names into the comment section of the shell.

Evaluation

I evaluate the project on the basis of it compiling without errors or warnings, running successfully, correctness according to the specs, thoroughness, and style.

The project will be penalized 10% if it needs to be resubmitted for compile errors, runtime errors, etc.

The project will be penalized 10% if delivered after the deadline.