Jonny Ramos

Math-Blast final Advanced iOS project

**Executive Summary**

Math-Blast is an educational, addition matching game. Much like Bejeweled, where you have to match gems of a certain color, in Math Blast, you have to match gems of different (or same) values that adds up to a determined amount (determined by what level you are playing). For example, in the first level of the game, you have to match gems that add up to 4. If you swipe across the following gems, 2 1 0 1, you successfully captured those gems giving you points. Bonuses are awarded for the number of gems you used, and the number of distinct numbers you used (2,1,1 is better than 2,2). 0 is a very popular gem that you can use to build up combos. For example, in level one, 2 2 will give you a score of 8 (look below for scoring algorithm) while 2 1 1 will give you a score of 24.

Scoring algorithm is as follows:

(Level target number X total gems used) X Distinct gems used

\*Level 1 target number starts at 4, and increments after each level.

Math-Blast is written from scratch, all classes and algorithms used are 100% custom. I used Object orientated designs to help me scope out the different classes. Math-Blast has a single GameController class, which controls every aspect of game play. I have also encapsulated classes for Gems, Timer (clock), GameBoard / GameGrid, Scoring, and Powerups.

The advanced animation are all possible because of Cocos2d. Cocos2d is an open source framework that is now being used by many popular game developers.

Cocos2d implements both layers and scenes. A layer in cocos2d is much like a layer in photoshop, you can draw labels, buttons, sprites on them. When the init method is called, it creates a scene. A scene can have multiple layers. For example, you can call a method [self addChild:sprite z:1] which will take a sprite (an image) and add it to the scene. This layer is now a child of the scene. Cocos2d also implements the concept of a Director. The director has control of all your scenes, you can pause, resume, stop animations, provide transitioning effects and much more. You can also move from scene to scene with the Derector. Calling methods with Director looks like this: [[CCDirector sharedDirector] replaceScene:nextScene].

Cocos2d also has a runAction method, which I feel is where most of the time is spent when coding. Take this line of code for example:

1. [titleLabel runAction:
2. [CCSequence actions:
3. [CCDelayTime actionWithDuration: 1],
4. [CCEaseOut actionWithAction:
5. [CCScaleTo actionWithDuration:1 scale:.5] rate:2.0], nil]];

This line of code displays a label that appears to zoom in on the screen. Line one initiates the action, line two sets up a sequence of actions, line 3 makes these actions wait 1 second before executing, lines 4 and 5 makes the label zoom in at the specified rates. Almost all sprite animations can be executed this way.