Blackstorm Labs Gem Swapper Reflection

By: Thomas Deeb

The Game

This Gem Swapper is a classic-style one where you select a gem and swap it with another gem. If three or more match after the swap, you get points, and new gems replace the old ones. Get as many points as you can in the 60 second time limit!

Technical Summary

The game is laid out in four source files:

1. **Application.js** – This is the entry point to the game. We create a StackView scaled to fit the device we're on. The width used is the width of the background image asset. Then we create a new GameScreen, push it onto the stack, and call GameScreen.build().
2. **GameScreen.js** – This is the view you see in the game. It sets up the gem grid and shows and manages the time and score.
3. **Grid.js** – This is the grid of gems you interact with and has a number of rows and columns. It populates the grid with a random set of gems and centers itself at a particular position based on how many rows and columns it has. We subscribe each gem to an input event in their own class to easily see which gem was selected, as the event does not tell you the object the input was started on.
    a. When you click on a gem, the input event fires and sets the heldGem variable to that gem. When clicking on another gem, we call getAdjacentGemsTo() to get an array of adjacent gems and check if the gem is in the adjacent list of heldGem. If so, we simply swap gem numbers with swapGems(), which locks input and goes into check for matches in checkHandleMatches(), which finds all horizontal and vertical chains. If any matches are found, we push the column the match is on to a matchCols array. matchCols is then used in deleteAddGems() to move down all the gems in that column based on how many are missing and create new gems that drop down from above into their spaces.
4. **Gem.js** – This represents a gem, which can be one of 5 types. The gem stores its row and column on the grid in addition to its gem number, which is used to check for matches and correlates to the image displayed for the gem. There is a selection image that shows up when the gem is selected.

Challenges

I have more experience in compiled, statically typed languages such as C#, C++, and Java, so coding in a dynamic language like JavaScript was different. However, the Game Closure devkit environment made it easy to test without having to wait a while for the program to compile, which was much appreciated.

Game Closure, setup was easy and the package system works great. However, it isn't without its faults. The documentation is rather scarce, especially on examples, and there are several features missing from other engines and frameworks, such as lighting and physics. Additionally, I had trouble finding information about the devkit online, as there are no forums or answer boards specifically for the devkit available, unlike many other frameworks. Furthermore, the scaleX and scaleY view properties didn't work for me at all, and I spent quite a while wondering why the UI looked off. I ended up resorting to the scale property. The documentation may be out of date, or those properties might need to be used some other way. Either way, I didn't (and still don't) know, and I feel this should be cleared up.

Closing

I added a bit of visual flare by fading the new blocks in as they appeared so it's clear that those are the ones being added to the grid. The gem selection image was created by me by hand in MSPaint. I do not have any current plans to improve the game.

HTML5 has a lot of potential, but I feel Game Closure is not using it all. Game Closure seems like it's meant only for small, simple games. I cannot see it being used to create a mobile MMO or an RPG game with a sufficient amount of depth.

On game development itself: I feel that game developers as a whole should be pushing boundaries to creating better games designed for the players themselves. Despite how young the mobile landscape is, most mobile games are quite literally clones of other games with very little differentiating factors, used as cash grabs by exploiting people's behavior. If Game Closure can push itself to encourage more in-depth, high-quality games for players to enjoy (instead of be just fun enough for them to purchase goods with real-world money), then I see a bright and enjoyable future for mobile.