SRRI Code Standards

June 28 2013

Abstract

This document outlines the standards decided upon for coding the SRRI Datagames project. If you update this document at any time, be sure to change the 'date' tag to the date of your update.

1 Variable Definitions

- Variable defintions should be prefixed with either m, or k, depending on whether they represent a variable, or constant respectively.
- The prefix character should always be lowercase. Start each word after that with uppercase.
- Variables should be named according to the value that they represent. Look at other variables in document if you need help with naming.
- Instance variables should be private.

2 Function Definitions

- The opening brace should be on the same line as the function name.
- Functions are named in camel case, i.e. likeThisOneIsNamed.

3 Commenting

- Comments for a function should be written on the line above the function definition. If the line ends up being very long, break it into two lines around 70 characters.
- In-line comments may be added within function definitions. Insert comments after the end of the line. If inline comments end up too long, select them, and press CMD-i to block them together.
- There are never enough comments. Program as literately as possible.

4 Packages

- Packages will be named in all lowercase, and with the _ character representing a space. Use of the default package is prohibited.
- Each game will have a 'main' package containing the .MXML and related source. This package will always be one word chosen based on the game title.
- If you have created several classes that together serve a purpose, create a package for them inside of the main package.

5 Whitespace

- Leave one blank line between each method/function that you define. If there is a comment above a function, leave a line between the end of your function, and the comment.
- Generally, leave one space character between parameters. This includes arguments to a function/method, as well as anywhere you have a comma between items in a list. i.e.

```
private var mArrayTest:Array = [1, 2, 3, 4];

private function testFunc(arg1:int, arg2:int):void{
    ...
}
```

6 Code Example

```
package test_package{

    private var mTestVar:Number = 1.618;

    public static const kTestConst:Number = 1.618;

    //Here is the comment explaining what testFunc does
    private function testFunc(arg1:String, arg2:Number):void{
        ...
    }

    //Here is the comment explaining what testFunc does
    private function testFunc(arg1:String, arg2:Number):void{
        ... //here is an example of an inline comment
    }
}
```