**前言**

JJSkin皮肤框架适用于所有iOS产品，如果你是一名iOS开发工程师，希望你读完本文，并且使用JJSkin在你的项目中。

首先定义一下这里皮肤的概念，皮肤即组成屏幕上界面的元素属性及其之间的关系。简单来说就是界面上控件的属性，像背景色，字体颜色等，以及这些控件之间的关系，例如，控件与控件之间的间距，控件相对于另一个控件的位置。做客户端一部分工作就是和皮肤有关，并且写皮肤相关的代码也是一件比较繁琐的事情。

**问题**

我们先来看一下JJSkin皮肤框架诞生的原因，也就是我们在做皮肤时遇见的问题。

1. 适配横屏和竖屏

2. 适配iPhone和iPad

3. 适配不同的屏幕尺寸

4. 适配特定机型，如6Plus

5. 需要对某一特定事件做不同的皮肤，如春节，夜间模式

6. 如果不使用XIB或Storyboard，需要写许多代码对控件属性初始化

对于1，2，3，4点UE都需要做不同的设计，代码需要做许多逻辑判断，你可能会看到许多if/else语句，直接导致维护性和扩展性降低。如果说以上几点对于程序员想办法还是可以处理的，但遇到第5点，如果皮肤都是直接代码编写，那么程序员会哭死的。对于第6点，我一直在想有什么办法将初始化代码放在一个地方。

福音来了，JJSkin皮肤框架很好解决了以上问题，本文就是要介绍如何使用此框架。

框架代码请查看https://github.com/hamilyjing/JJSkin。

**JJSkin框架设计思想**

我们将皮肤相关属性写在文件中，只修改皮肤文件，不修改代码，而达到换肤的目的。

皮肤可以由一组配置文件组成，都有对应iPhone或iPad的通用配置文件，用来定义通用属性，以及针对特殊情况的配置文件，例如针对320\*480屏幕尺寸和6Plus等做特殊处理的配置文件。当用户想要获取某一属性（也有可能是一组属性）时，框架默认先从特定机型的配置文件查找（即5s，6，6p等），如果有的属性值为空，则针对当前屏幕尺寸的配置文件中继续查找（如果存在），然后将本次获取的属性和在上个配置文件获取的属性进行merge，merge的原则是只有当上个文件获取的属性值为空，则才将当前获取的value赋值给此属性，如果有属性值仍为空，继续查找通用配置文件，再次进行merge。

具体配置文件个数和读取顺序可以由用户决定，用户只需要编写配置文件类，实现JJSkinConfig协议，然后加入到JJSkinManager中即可。

在每个皮肤文件中，都会针对横屏和竖屏来对控件属性设值。获取属性值的原则如下：当界面是竖屏状态，首先读取竖屏属性值，没有读到，则选择下一个文件，继续读取竖屏属性值，直到读取到，即竖屏情况下，永远不会去读横屏属性值。反之，当界面是横屏状态，首先读取横屏属性值，没有读到，读取竖屏属性值，仍然没有读到，则读取下一个文件横屏属性值，然后竖屏属性值，直到读取最后一个文件的竖屏属性值（如果属性值仍然为空）。

**如何使用**

首先介绍下JJSkin的使用方法。

1. 下载JJSkin代码，将JJSkin/JJSkin文件夹放到项目中（ARC编译），导入头文件"JJSkin.h”。

2. 编写皮肤文件，根据实际需要来看是否要创建多个配置文件。

{

"portrait": {

"stringValue": "1234567890",

"intergerValue": "890",

"floatValue": "234.9",

"boolValue": "1",

"edgeInsets": "{1,2,3,4}",

"rect": "{{5,6},{7,8}}", // CGRect值的格式{{x,y},{width,height}}

"size": "{9,0}", // CGSize值格式{width,height}

"color": "#FFc58dd0", // UIColor对应值格式#RGB或#ARGB或#RRGGBB或#AARRGGBB

"textLabel": {

"text": "portrait", // 必须和JJLabelStyle类text属性名字一直，下面会详细介绍

"textAlignment": "right",

"lineBreakMode": "clipping"

}

},

"landscape": {

"textLabel": {

"text": "landscape",

"textAlignment": "center",

}

}

}

3. 如果需要，编写配置文件类，实现JJSkinConfig协议，并加入到JJSkinManager对象中。

@protocol JJSkinConfig <NSObject>

- (NSBundle \*)bundle;

- (NSString \*)fileNamePrefix;

- (NSString \*)fileType;

- (NSString \*)landscapeJsonLabel;

- (NSString \*)portraitJsonLabel;

- (NSString \*)iPhoneFileNameSuffix;

- (NSString \*)iPadFileNameSuffix;

- (NSArray \*)fileNames;

@end

皮肤文件默认放在当前bundle里，并且文件名前缀是"jjSkin-"，文件类型是"json"，例如通用皮肤文件名"jjSkin-iPhone.json"，375x667尺寸皮肤文件名"jjSkin-iPhone375x667.json"，针对iPhone 5C的皮肤文件名"jjSkin-iPhone 5C.json"。其中fileNames决定框架读取皮肤文件的顺序。

4. 直接获取或更新控件属性值

获取和更新控件属性值，都是通过ID得到，ID是配置文件标签的路径，以点号分割，如@"R.floatValue"，@"R.textLabel.text"等，ID以"R"开头，并且不包含横屏和竖屏标签，框架会根据界面方向自动加入。

NSString \*stringValue = [JJSkinManager getStringByID:@"R.stringValue"]; // "1234567890"

CGFloat floatValue = [JJSkinManager getFloatByID:@"R.floatValue"]; // 234.9

CGRect rect = [JJSkinManager getRectByID:@"R.rect"]; // {{5,6},{7,8}}

/\*\*

直接获取UILabel对象。

竖屏下，label内容是"portrait"，并且文字右对齐。

横屏下，label内容是"landscape"，并且文字居中对齐。

\*/

// 神奇的地方，只需要一行代码，label的属性在不同的情况下如iPhone，iPad，横屏，竖屏等等，有不同的值。

UILabel \*label = [JJSkinManager getLabelByID:@"R.textLabel"];

/\*\*

更新已存在UILabel对象属性。

label1和label的text和textAlignment属性值相同。

\*/

UILabel \*label1 = [[UILabel alloc] init];

[JJSkinManager updateLabel:label1 withID:@"R.textLabel"];

**皮肤Style**

皮肤Style定义的是控件属性的集合。皮肤配置文件中每个key都对应一种Style。当key对应的value是string类型，则它是JJCommonStyle，例如"R.stringValue"；如果value是dictionary类型，则它是一种属性集合style，如JJLabelStyle，JJButtonStyle等，以及用户自定义的style，像"R.textLabel"是JJLabelStyle。一种style由一个或多个style组成。

当用户获取某一ID对应的值时，JJSkin将文件中ID对应的value转化成某一种Style（由用户决定，即调用JJSkinManager不同的API），然后由Style实现对象的生成。

皮肤Style有一个公共基类JJSkinStyle，框架默认提供常用的Style，如JJLabelStyle，JJImageStyle等，用户可以定义自己的Style，需要继承JJSkinStyle，并实现如下两个方法：

+ (id)objectFromStyle:(id)style;

- (void)updateObject:(id)object;

然后使用JJSkinManager下面两个API获取或更新控件：

+ (id)getObjectByID:(NSString \*)id withStyleClass:(Class)styleClass;

+ (void)updateObject:(id)object withID:(NSString \*)id withStyleClass:(Class)styleClass;

当配置文件属性使用JJCommonStyle类型，此类型的key可以是任何名字。

"portrait": {

"stringValue": "1234567890”, // “stringValue”可以是任意名字

"intergerValue": "890",

"floatValue": "234.9",

"boolValue": "1",

"edgeInsets": "{1,2,3,4}",

"rect": "{{5,6},{7,8}}", // CGRect值的格式{{x,y},{width,height}}

"size": "{9,0}", // CGSize值格式{width,height}

"color": "#FFc58dd0", // UIColor对应值格式#RGB或#ARGB或#RRGGBB或#AARRGGBB

}

当配置文件属性使用非JJCommonStyle类型，此类型的key必须和对应Style类属性名字一致。因为框架通过运行时方法，匹配类属性名字和key，然后进行赋值。

"landscape": {

"textLabel": {// textLabel是JJLabelStyle类型

"text": "landscape", // "text"必须和JJLabelStyle中text属性名字一直，否则赋值失败

"textAlignment": "center",

}

}

皮肤文件中有一个status的key，当值为"finish"时，即便有部分属性值没有值，也不在继续继续查找，但不包括子Style；当值为"finishIncludeSon"时，其子Style的属性值没取完整，也不再继续查找。

"landscape": {

"textLabel": { // textLabel的属性只有text和textAlignment，因为没有子style，所以不会查找portrait和其他配置文件。

"status": "finish",

"text": "landscape",

"textAlignment": "center",

}

}

**通知**

用户可以调用JJSkinManager中changeSkin方法进行换肤，此方法是发出名为”JJSkinChangedNotification”通知。

JJSkin框架实现了JJSkinView和JJSkinViewController两个基类，进行监听换肤事件，用户只需继承这两个类，并实现layoutViewsWhenSkinChanged方法。

综述，我们只需将皮肤属性写在配置文件里，只需一行代码就可以适应不同状态下属性值的变化

最后，欢迎您使用JJSkin，并提出宝贵意见。

Thanks!

JJ（宫健）