作为一个写了三年多Cocos的(cai)大(bi)神,从C++到lquick到Cocos-js再到现在的CocosCreator，也算是各种语言都写了一遭。对于现在的CocosCreator(下文简称CCC),从刚开始出现之时的质疑，“这玩意不是完全照抄Unity吗”。到用了一段时间之后，发现开发效率确实比以前提高了不少。总的来说，觉得CCC的未来一定是大有可为的。但是这其中有个比较致命的问题，就是CCC把底层封装成JS和组件暴露给开发者之后，能理解cocos底层的人将会越来越少，将来很多没有从C++走过来的人可能对cocos只能知其然，而不能知其所以然。更不要说Hank一些代码来满足自己的需求啦。比如说，今天这篇文章将要讲述的如何切换横竖屏的问题。

本教程所使用的版本为CCC的1.4.2版本，不同版本的代码可能会有所差异，所以，本次教程不会机械的上去贴代码，而是一步步深入分析问题，从而引导出我们想要的结果。阅读本文之前希望读者有开发过cocos-2dx 和 cocos-js的基础，但是即使没有，也不会影响接下来阅读。本文章是完全面向新手的，我会尽量的将每个知识点都阐述的简明易懂。

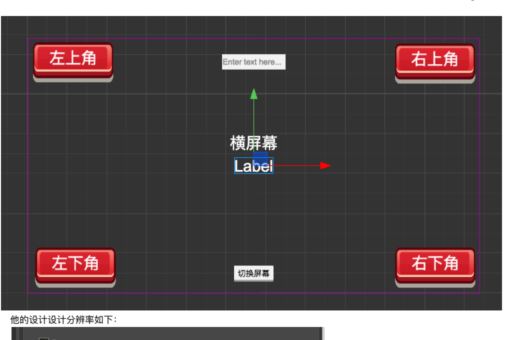
首先我们先阐述一下几个基本概念.

1.屏幕的画布分辨率： frameSize

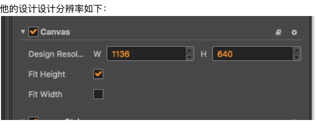
frameSize是屏幕的真实的分辨率，比如说一个iphone5s，在横屏的时候那么它的画布分辨率就是 1136 x 640,而在竖直的情况下
一定是 640 x 1136。这是固定的永远都不会变的。这点很关键。在cocos中得到frameSize的 api 一般是
xxx.getFrameSize(); 这个
xxx到底是个啥，不同的语言有不同的调用方法，但是看到 getFrameSize()这个东西，我们就要意识到这个是得到屏幕的画布分辨率，
同理 xxxx.setFrameSize()就是设置咯。

2.在CCC中创建2个Scene

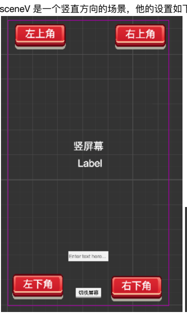
sceneH 是一个横向的屏幕，我给它的四个角都加了一张精灵来标示屏幕的位置，屏幕上所有的元素都挂载了widget组件



他的设计设计分辨率如下：



sceneV 是一个竖直方向的场景，他的设置如下。



每次点击“切换屏幕”按钮之后，我们就会首先会调用一个函数，把屏幕设置成竖屏还是横屏，接下，转跳到对面的Scene上

我们可以看到每个屏幕上都有label，这里，我们会显示一下屏幕的基本信息。

```
...
start:function () {
    var string1  = 'visibleSize: '+ cc.visibleRect.width.toFixed(2) + '-' + cc.visibleRect.height.toFixed(2);
    var string2  = 'designSize:' + cc.view.getDesignResolutionSize().width + '-'+ cc.view.getDesignResolutionSize().height;
    var string3  = 'frameSize:'+ cc.view.getFrameSize().width+'-'+cc.view.getFrameSize().height;
    this.viewLabel.string = string1 + '\n' + string2+'\n'+string3;    cc.log(this.viewLabel.string);
},
...
```

代码如上，我们在Start函数里显示了当前屏幕的的基本信息。

3.开始写实现屏幕反转的函数，创建JsNativeBridget.js文件 并且设置这个文件为插件

```
window.JsNativeBridge = {
   // true 是切换到横屏， false 是切换到竖屏
   changeOrientationH:function (val) {
      if(cc.sys.isBrowser){
         var frameSize = cc.view.getFrameSize();
         cc.view.setFrameSize(frameSize.height,frameSize.width);
      }
      else{
         if(cc.sys.os == cc.sys.OS_IOS){
            jsb.reflection.callStaticMethod(
              "AppController",
              "changeOrientationH:",
              val );
         }
         else if(cc.sys.os == cc.sys.OS_ANDROID){
            jsb.reflection.callStaticMethod(
              "org/cocos2dx/javascript/AppActivity",
              "changeOrientationH",
              "(Z)V",
              val );
        }
      }
   },
```

web:在web环境下为了方便，我们日常的开发，我们可以直接获取到当前的frameSize，然后让他们互换一下就好啦
代码如下，实现起来真是美滋滋呢。

```
var frameSize = cc.view.getFrameSize();
cc.view.setFrameSize(frameSize.height,frameSize.width);
```

ios: 因为IOS平台上OC和C++ 可以混编，而且还没有不同线程之间通信的问题，所以我们先来搞定IOS吧。记得要先构建发布Native端工程才行啊

写过C++代码的同学应该对于cocos的渲染有所了解, cocos的Director持有一个openGlView对象，cocos的所有渲染都是靠着这个openGL来渲染,
相信大家对于这段代码不会陌生。

```
bool AppDelegate::applicationDidFinishLaunching()
{
#if CC_TARGET_PLATFORM == CC_PLATFORM_IOS && PACKAGE_AS
    SDKManager::getInstance()->loadAllPlugins();
#endif
    // initialize director
    auto director = Director::getInstance();
    auto glview = director->getOpenGLView();
    if(!glview) {
#if(CC_TARGET_PLATFORM == CC_PLATFORM_WP8) || (CC_TARGET_PLATFORM == CC_PLATFORM_WINRT)
        glview = GLViewImpl::create("hello_world");
#else
        glview = GLViewImpl::createWithRect("hello_world", Rect(0,0,900,640));
#endif
        director->setOpenGLView(glview);
    }
```

但是这个openGLView到底是如何渲染的呢，我们打开Xcode工程，打开ios文件夹下的AppController.mm.

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{

    // Override point for customization after application launch.

    // Add the view controller's view to the window and display.
    window = [[UIWindow alloc] initWithFrame: [[UIScreen mainScreen] bounds]];
    CCEAGLView *eaglView = [CCEAGLView viewWithFrame: [window bounds]
                                         pixelFormat: kEAGLColorFormatRGBA8
                                         depthFormat: GL_DEPTH24_STENCIL8_OES
                                  preserveBackbuffer: NO
                                          sharegroup: nil
                                       multiSampling: NO
                                     numberOfSamples: 0 ];

    [eaglView setMultipleTouchEnabled:YES];

    // Use RootViewController manage CCEAGLView
    viewController = [[RootViewController alloc] initWithNibName:nil bundle:nil];
    viewController.wantsFullScreenLayout = YES;
    viewController.view = eaglView;

    // Set RootViewController to window
    if ( [[UIDevice currentDevice].systemVersion floatValue] < 6.0)
    {
        // warning: addSubView doesn't work on iOS6
        [window addSubview: viewController.view];
    }
    else
    {
        // use this method on ios6
        [window setRootViewController:viewController];
    }

    [window makeKeyAndVisible];

    [[UIApplication sharedApplication] setStatusBarHidden: YES];

    // IMPORTANT: Setting the GLView should be done after creating the RootViewController
    cocos2d::GLView *glview = cocos2d::GLViewImpl::createWithEAGLView(eaglView);
    cocos2d::Director::getInstance()->setOpenGLView(glview);

    cocos2d::Application::getInstance()->run();
    return YES;
}
```

对于IOS原生应用稍微了解一点的同学都知道，IOS的展示是靠着一个个的ViewController来实现的，这里的view跟cocos的Scene非常似，比如说我们的QQ从登陆到进入好友列表，可能就是从登陆的view切换到好友的view一样，这段代码里我们可以看到，引擎创建了一CCEAGLVIew的对象，并且将这个对象放到了一个rootView上。然后将这个对象包裹成一个cocos2d:GLView ，然后再将它交给我们director，然后我们的cocos程序就启动起来啦。

所以在IOS上想要切换屏幕的横竖屏，其实我们要做的就是切换IOS原生的view的横竖屏。而且，正常的cocos应用在IOS里只有一个view，这个view就是用来承载渲染的opglView的view，我们游戏里的Scene不管怎么切换，也一直是在这一个view上渲染的。所以我们就找这个view下手就对了。

在IOS里，想要切换一个view的横竖屏应该是有接口的，毕竟这是一个很常见的需求，但是自从IOS8之后，苹果爸爸不再建议开发者这么干了，希望大家根据重力感应自由的让屏幕旋转来切换横竖屏的，当然在游戏大厅了。麻将就是竖屏，而打飞机一定就是横屏，这可不是靠着重力来的，所以，我们需要第二个解决方案，那就是创建第二个view，本来存在的第一个view是固定的横屏幕。而第二个view是固定的竖屏幕。我们需要什么方向的，就让那个方向的view显示，同时把CCEAGLVIew在2个view之间相互转移，谁显示就转移到谁的身上。

我们打开 RootViewController.mm观察这三个函数，
第一个允许屏幕的方向是 左横屏和右横屏，

```
// This method is deprecated on ios6
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    return UIInterfaceOrientationIsLandscape( interfaceOrientation );
}

// For ios6, use supportedInterfaceOrientations & shouldAutorotate instead
- (NSUInteger) supportedInterfaceOrientations{
#ifdef __IPHONE_6_0
    return UIInterfaceOrientationMaskLandscape;
#endif
}

- (BOOL) shouldAutorotate {
    return YES;
}
```

允许屏幕旋转的方向

屏幕的方向

是否允许屏幕旋转

接下来我们创建第二个View， 在这个view里我们允许view显示的方向为竖直方向

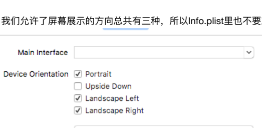
RootViewControllerV.h
RootViewControllerV.m

```
#import <UIKit/UIKit.h>
#import "RootViewController.h"

@interface RootViewControllerV : RootViewController {

}
- (NSUInteger) supportedInterfaceOrientations;
- (BOOL) shouldAutorotate;
@end
```

```
****************************************************************************/

#import "RootViewControllerV.h"

@implementation RootViewControllerV

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    return UIInterfaceOrientationIsPortrait( interfaceOrientation );
}

// For ios6, use supportedInterfaceOrientations & shouldAutorotate instead
- (NSUInteger) supportedInterfaceOrientations{
#ifdef __IPHONE_6_0
    return UIInterfaceOrientationMaskPortrait;
#endif
}

- (BOOL) shouldAutorotate {
    return YES;
}


@end
```

只允许屏幕旋转到竖直方向

在这里。我们允许了屏幕展示的方向总共有三种，所以Info.plist里也不要忘记构选上相应的方向哦



接下来就是实现我们的函数啦。

AppController.h

```
@class RootViewController;
@class RootViewControllerV;

@interface AppController : NSObject <UIAccelerometerDelegate, UIAlertViewDelegate, UITextFieldDelegate,UIApplicationDelegate>
{
    UIWindow *window;
    RootViewController    *viewController;
    RootViewControllerV * viewControllerV;
}

@end
```

hello_world 〉 ios 〉 AppController.mm 〉 -application:didFinishLaunchingWithOptions:

```
#import "RootViewController.h"
#import "RootViewControllerV.h"
#import "platform/ios/CCEAGLView-ios.h"

@implementation AppController

#pragma mark -
#pragma mark Application lifecycle

// cocos2d application instance
static AppDelegate s_sharedApplication;
static AppController * appController;
```

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{

    // Override point for customization after application launch.

    // Add the view controller's view to the window and display.
    window = [[UIWindow alloc] initWithFrame: [[UIScreen mainScreen] bounds]];
    CCEAGLView *eaglView = [CCEAGLView viewWithFrame: [window bounds]
                                         pixelFormat: kEAGLColorFormatRGBA8
                                         depthFormat: GL_DEPTH24_STENCIL8_OES
                                  preserveBackbuffer: NO
                                          sharegroup: nil
                                       multiSampling: NO
                                     numberOfSamples: 0 ];

    [eaglView setMultipleTouchEnabled:YES];

    // Use RootViewController manage CCEAGLView
    viewController = [[RootViewController alloc] initWithNibName:nil bundle:nil];
    viewController.wantsFullScreenLayout = YES;
    viewController.view = eaglView;

    appController = self;
    viewControllerV = [[RootViewControllerV alloc] initWithNibName:nil bundle:nil];
    viewControllerV.wantsFullScreenLayout = YES;
```

在AppController.mm里加入如上的代码，

然后我们实现 +(void)changeOrientationH:(BOOL)val 函数

```
//改变屏幕旋转
+(void)changeOrientationH:(BOOL)val{
    CCEAGLView * glview = nil;
    RootViewController * nextViewController = nil;
    if(val == NO){
        //切换到竖屏
        glview = (CCEAGLView *)appController->viewController.view;
        appController->viewController.view = nil;
        appController->viewControllerV.view = glview;
        nextViewController = appController->viewControllerV;
    }
    else{
        //切换到横屏
        glview = (CCEAGLView *)appController->viewControllerV.view;
        appController->viewControllerV.view = nil;
        appController->viewController.view = glview;
        nextViewController = appController->viewController;
    }

    // Set RootViewController to window
    if ( [[UIDevice currentDevice].systemVersion floatValue] < 6.0)
    {
        // warning: addSubView doesn't work on iOS6
        [appController->window addSubview: nextViewController.view];
    }
    else
    {
        // use this method on ios6
        [appController->window setRootViewController:nextViewController];
    }
//    CATransform3D transform;
//    if(val == NO)
//        transform = CATransform3DMakeRotation(M_PI / 2, 0, 0, 1.0);
//    else
//        transform = CATransform3DMakeRotation(0, 0, 0, 1.0);
//    appController->viewController.view.layer.transform = transform;
    NSLog(@"%@", [NSString stringWithFormat:@"width %d height %d", [glview getWidth],[glview getHeight]]);
    cocos2d::GLView * cocosView = cocos2d::Director::getInstance()->getOpenGLView();
    cocosView->setFrameSize([glview getWidth],[glview getHeight]);

    if(val)
        NSLog(@"切换屏幕方向到横屏");
    else
        NSLog(@"切换屏幕方向到竖屏");
}
```

这就是是整个函数的实现， 在这里的话，一开始，我们发现屏幕确实从横屏旋转成了竖屏幕，但是竖直屏幕的Scene的frameSize仍旧是横屏的 1136 x 640。原来我们切换了方向之后，CCEGLView的frameSize确实发生了变化，但是director的opglView的frameSize还是老的，没有变化，所以，红框里的代码很重要。

其实opglView的frameSize就是在

```
cocos2d::GLView *glview = cocos2d::GLViewImpl::createWithEAGLView(eaglView);
```

这句代码里根据eaglView的 分辨率来设置的，感谢兴趣的同学可以点进去看看。这里不再多费笔墨了。

填完了IOS的坑，我们再来趟android的水吧，

通过刚才对IOS底层的探索，我们对付起来android会更加的得心应手，其实android也有个叫activity的东西，cocos也是把opgelVIew放在一个固定的activey上渲染，所以我们只需要1.旋转activey 。2.改变direct的opglView的frameSize。

```
public class AppActivity extends Cocos2dxActivity {

    public  static AppActivity instance = null;

    @Override
    public boolean navigateUpTo(Intent upIntent) { return super.navigateUpTo(upIntent); }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SDKWrapper.getInstance().init(this);
        instance = this;
    }
```

```
    public static void changeOrientationH(boolean val){
        if(val == true){
            instance.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE);
        }
        else{
            instance.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        }

        if(val)
            Log.d("切换屏幕","横屏幕");
        else
            Log.d("切换屏幕","竖屏幕");
    }
```

相对IOS来说， android的代码要少很多,但是说好的 把opgelVIew放到activity上呢？？ 在 onCreate函数里根本啥子都没有啊。不要急，我们点看AppActivity的父类 Cocos2dxActivity进去看一看。

Cocos2dxActivity onCreate()

```
    // ===========================================================

    @Override
    protected void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        this.hideVirtualButton();

        onLoadNativeLibraries();

        sContext = this;
        this.mHandler = new Cocos2dxHandler(this);

        Cocos2dxHelper.init(this);

        this.mGLContextAttrs = getGLContextAttrs();
        this.init();

        if (mVideoHelper == null) {
            mVideoHelper = new Cocos2dxVideoHelper(this, mFrameLayout);
        }

        if(mWebViewHelper == null){
            mWebViewHelper = new Cocos2dxWebViewHelper(mFrameLayout);
        }

        if(mEditBoxHelper == null){
            mEditBoxHelper = new Cocos2dxEditBoxHelper(mFrameLayout);
        }

        Window window = this.getWindow();
        window.setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_ADJUST_PAN);
        this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
    }
```

这个init()看着似乎很可疑，我们再点进去看看。

Cocos2dxActivity init()

```
    // Methods
    // ===========================================================
    public void init() {

        // FrameLayout
        ViewGroup.LayoutParams framelayout_params =
            new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                                       ViewGroup.LayoutParams.MATCH_PARENT);

        mFrameLayout = new ResizeLayout(this);

        mFrameLayout.setLayoutParams(framelayout_params);

        // Cocos2dxEditText layout
        ViewGroup.LayoutParams edittext_layout_params =
            new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                                       ViewGroup.LayoutParams.WRAP_CONTENT);
        Cocos2dxEditBox edittext = new Cocos2dxEditBox(this);
        edittext.setLayoutParams(edittext_layout_params);


        mFrameLayout.addView(edittext);

        // Cocos2dxGLSurfaceView
        this.mGLSurfaceView = this.onCreateView();

        // ...add to FrameLayout
        mFrameLayout.addView(this.mGLSurfaceView);

        // Switch to supported OpenGL (ARGB888) mode on emulator
        if (isAndroidEmulator())
           this.mGLSurfaceView.setEGLConfigChooser(8, 8, 8, 8, 16, 0);

        this.mGLSurfaceView.setCocos2dxRenderer(new Cocos2dxRenderer());
        this.mGLSurfaceView.setCocos2dxEditText(edittext);

        // Set framelayout as the content view
        setContentView(mFrameLayout);
    }
```

哈 surfaceView是什么鬼？？ 你是opglView的表哥吗？？ 等等，这个cocos2dxRenderer()不就是渲染的意思吗。666，我们进去这2个Java文件里看看有什么可以的地方吗?

Cocos2dxRenderer onSurfaceChanged()

```
    public void onSurfaceChanged(final GL10 GL10, final int width, final int height) {
        Cocos2dxRenderer.nativeOnSurfaceChanged(width, height);
    }
```

```
private static native void nativeOnSurfaceChanged(final int width, final int height);
```

功夫不负有心人，我们终于找到了这个奇怪的函数，貌似是当屏幕变化的时候会触发。经过打断点验证，每次，我点击屏幕选择的时候，都会触发这个函数，燃而且传入的width和 height都是正确的屏幕旋转了之后的值。我们来看看这个nativeOnSurfaceChangedC++层是怎么实现它的。

在cocos2d-x/cocos/platform/android/javaactivity-android.cpp里有这个函数的实现

```
JNIEXPORT void Java_org_cocos2dx_lib_Cocos2dxRenderer_nativeOnSurfaceChanged(JNIEnv*  env, jobject thiz, jint w, jint h)
{
    cocos2d::Application::getInstance()->applicationScreenSizeChanged(w, h);
}

}
```

一时语塞，我们再在同级的目录下找到了 CCApplication-android.cpp里找到了 applicationScreenSizeChanged这个函数，里边居然是空的，厉害了，我的哥，

正好在C++层，所以我们来写点代码吧

```
void Application::applicationScreenSizeChanged(int newWidth, int newHeight) {
    CCLOG('android Application::applicationScreenSizeChanged %d,%d',newWidth,newHeight);
    cocos2d::GLView * cocosView = cocos2d::Director::getInstance()->getOpenGLView();
    cocosView->setFrameSize(newWidth,newHeight);
}
```

这下子，妥妥没问题了，跑起来，跑起来，屏幕是旋转了，但是还是frameSize还没有变化。此时的我机智的想到了多线程的问题，毕竟安卓的多线程比较容易坑到人，所以要留意下，

```
window.JsNativeBridge = {


    // true 是切换到横屏， false 是切换到竖屏

    changeOrientationH:function (val,cb) {
        if(this._isOrientationH() == val){
            cc.log('当前屏幕已经是指定的方向了，无需做更改');
            return;
        }

        if(cc.sys.isBrowser){
            var frameSize = cc.view.getFrameSize();
            cc.view.setFrameSize(frameSize.height,frameSize.width);
        }
        else{
            if(cc.sys.os == cc.sys.OS_IOS){
                jsb.reflection.callStaticMethod(
                    "AppController",
                    "changeOrientationH:",
                    val
                );
            }
            else if(cc.sys.os == cc.sys.OS_ANDROID){
                jsb.reflection.callStaticMethod(
                    "org/cocos2dx/javascript/AppActivity",
                    "changeOrientationH",
                    "(Z)V",
                    val
                );
            }
        }

        cb && setTimeout(function () {
            cb();
        },500);

    },

    //当前屏幕是不是横屏
    _isOrientationH:function(){
        var frameSize = cc.view.getFrameSize();
        return frameSize.width > frameSize.height;
    },

};
```

重新改早一下函数，加入旋转之后的回调，旋转之后延迟 0.5秒再进行回调，在回调里转跳Scene，再次试验，搞定了。

最后一个问题，**applicationScreenSizeChanged**在安卓里会触发，在IOS里会触发吗，我们找到了cocos2d-x/cocos/platform/ios/CCApplication-ios.mm进去看看，发现这也是个空函数，拿他什么时候会触发呢？？ 我们打开RootViewController.mm里看到这个函数

```
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation {
    [super didRotateFromInterfaceOrientation:fromInterfaceOrientation];

    cocos2d::GLView *glview = cocos2d::Director::getInstance()->getOpenGLView();
    if (glview)
    {
        CCEAGLView *eaglview = (CCEAGLView*)glview->getEAGLView();

        if (eaglview)
        {
            CGSize s = CGSizeMake([eaglview getWidth], [eaglview getHeight]);
            cocos2d::Application::getInstance()->applicationScreenSizeChanged((int) s.width, (int) s.height);
        }
    }
}
```

这个**didRotateFromInterfaceOrientation**触发的条件的貌似是 同一个view在旋转之后触发，我们因为是2个view来回切换，所有并不会触发这个函数，自然也没法在这个空函数里改变frameSize啦