项目总结之截屏细节考虑

DionysosLai([906391500@qq.com](mailto:906391500@qq.com)) 2014/12/22

2014项目总算告一段落，这个产品顺顺利利从开始到最后的上市，圆满成功。借着项目总结，回顾自己做的几个游戏，将一些细节问题归纳。第一篇，就以目前做的的新游戏《圣诞节》开篇，讲讲游戏截屏问题。

对于截屏，本身技术，并不是很复杂，一般有两种方法，一种是使用Opengl像素取点方式；另一种是使用RenderTexture纹理方法，详细内容，可以参考，之前写的一篇文章：<http://blog.csdn.net/dionysos_lai/article/details/23467209>。

**两个技术方案考究：**

在我游戏中，我使用的是RenderTexture方法，之所以我要使用的是RenderTexture方法，而不是Opengl中获取像素点glReadPixels的方法，是基于以下几点考虑：

1. 二者在效率方面，基本没有差别，可以不考虑；（实际上RenderTexture方法，效率应该更低点）；
2. 由于在截屏时，我们不可能是将整个当前屏幕所有的元素全部截取下来，必然存在一些元素并不是我们需要的，还有一些元素需要我们临时贴上去，比方说logo之类的。这样的话，使用Opengl方法，必然在截屏之前，必须先去掉一些元素、添加上一些元素，截屏结束，也必须将元素反过来处理一遍。这之间，就存在着一些时间差，做的不好，就是出现闪屏效果。而使用Rendertexture方法，就可以在begin和end之外对元素进行处理，不会有闪屏效果。
3. 如果游戏中，我们使用了蒙版技术或者贴上一些半透明图片，难么很抱歉，如果不对图片进行特殊出处理，游戏中图片和截屏出来的图片是不一样的。而使用Rendertexture方法，我们可以很容易先对图片进行混合处理，使截屏图片和游戏画面一致。

**截屏细节把握：**

正是基于以上三点，我采用的RenderTexture方法。下面，详细介绍截屏细节。

1. **文件读写问题：**

文件读写问题，包括图片保存和获取，由于图片读取，基本上不用考虑很多，因此重点是文件写问题。

对于文件写问题，首先考虑的第一个问题就是系统容量检测问题，这点十分必要，如果系统容量不够，而程序强制性写入，导致的第一个问题必然是程序卡机，甚至是系统挂掉（我们平板是第一次试水，在防护方面做得比较差，曾经有音效加载问题，导致系统奔溃）。这可是一个不折不扣的A类bug啊，如果没做这不处理，那么就等着测试妹子找你麻烦吧。

**检测系统**代码如下：

.h

///@brief 检测sd卡是否可用容量足够

///@param[in] size---检测容量 注意：这里是已MB来就算

///@return 0---sd卡不可用 1---sd卡容量不够 2---sd卡可用，容量足够

int checkAvailableSDSize(const unsigned int& size);

.cpp

int HomeScene::checkAvailableSDSize( const unsigned int& size )

{

int availableOk = 0;

/// 目前，只做android平台检测，其他平台一律默认通过

#if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_ANDROID)

JniMethodInfo methodInfo;

jint ret = 0;

if (JniHelper::getStaticMethodInfo(methodInfo,

"com.mesrjni.MesrJni",

"checkAvailableSDSize",

"(I)I"))

{

ret = methodInfo.env->CallStaticIntMethod(methodInfo.classID, methodInfo.methodID,

size);

if (0 == ret)

{

CCLOG("sd's state is abnormal!");

availableOk = 0;

}

else if (1 == ret )

{

CCLOG("sd's availabel size is not enough!");

availableOk = 1;

}

else if (2 == ret )

{

CCLOG("sd's availabel size is enough!");

availableOk = 2;

}

else

{

CCAssert(false, "There is some wrong");

}

}

#else

availableOk = 2;

#endif

return availableOk;

}

.java

/\*

\* 检测sd卡是否可用容量足够

\*/

public static int checkAvailableSDSize(int size){

/// 先检测SD卡是否可用

String state = Environment.getExternalStorageState();

if(!Environment.MEDIA\_MOUNTED.equals(state)){

/// 对sd卡上的存储可以进行读/写操作

Log.d("DEBUG", "SD's state is abnormal!");

/\* new AlertDialog.Builder(activity)

.setTitle("圣诞节" )

.setMessage("小朋友，目前不能保存相册，快叫爸爸妈妈来解决吧！" )

.setPositiveButton("确定" , null )

.show(); \*/

return 0;

}

/// 获取sd卡用容量

File path = Environment.getExternalStorageDirectory(); //取得sdcard文件路径

StatFs stat = new StatFs(path.getPath());

long blockSize = stat.getBlockSize();

long availableBlocks = stat.getAvailableBlocks();

long availableSize = (availableBlocks \* blockSize)/1024/1024;

Log.d("DEBUG", "可用空间:" + availableSize + "Mb");

/// 判断容量是否足够

if(size > availableSize){

Log.d("DEBUG", "The sd's availavle size is not enough!");

/\* new AlertDialog.Builder(activity)

.setTitle("圣诞节" )

.setMessage("小朋友，SD卡空间不足了哦，不能保存相册。快叫爸爸妈妈清理一下吧！" )

.setPositiveButton("确定" , null )

.show();\*/

return 1;

}

/// 容量足够

Log.d("DEBUG", "The sd's availavle size is enough!");

/\* new AlertDialog.Builder(activity)

.setTitle("圣诞节" )

.setMessage("小朋友，SD卡空间不足了哦，不能保存相册。快叫爸爸妈妈清理一下吧！" )

.setPositiveButton("确定" , null )

.show();\*/

return 2;

}

这里调用JNI方法，因此这段代码"com.mesrjni.MesrJni"，要根据自己游戏进行适配；同时这里注意我注释的一段话，我会调用一个AlertDialog类，来提醒玩家出现问题。这里我将代码注释掉了，因为会出现一些很奇怪问题，其实与我下面JNI调用Toast类一样，后面讲。

保存图片，一般是保存在自己的一个文件夹中，因此我们要首先判断我们是否创建了文件夹，没有，就直接创建一个新的文件夹。这里代码简单，ps，在在创建文件之间，要确保SD可用，这个工作也是必须要做的，可参考上面代码：

**创建文件夹代码：**

File destDir = new File("/sdcard/Christmas/");

if (!destDir.exists()) {

destDir.mkdirs();

}

这里我创建了一个Christmas文件夹。

由于保存图片工作，在Android端比较慢，大约会有2s左右时间，这个时间还得看这个平台的硬件水平，在我们公司的平板上，一般是2s左右。因此，点击保存图片时，会出现游戏卡顿问题，那么玩家还以为是游戏没反应，会一直点击保存图片图标（这个现象，基本是玩过这个游戏的玩家，都会如此操作）。因此，很有必要添加一个提示保存图片中的提示标签。

**保存提示：**

保存提示功能，这里我直接使用了Android自带的Toast类，同时使用Toast类，只能是使用JNI非静态类调用，因此要先同时JNI静态类调用，或者App的object对象。

获取App的object对象：

.cpp

#if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_ANDROID)

JniMethodInfo methodInfo;

jobject jobj;

bool isHave = JniHelper::getStaticMethodInfo(methodInfo,

"com.mesrjni.MesrJni", "getRunActivity","()Ljava/lang/Object;");

if (isHave)

{

jobj = methodInfo.env->CallStaticObjectMethod(methodInfo.classID, methodInfo.methodID);

}

#else

.java

/\*

\* 获取this

\*/

public static Object getRunActivity() {

System.out.println("----------GetRunActivity");

return activity;

}

下面就是调用调用Toast类方法类，给出两端java不同代码，仔细分析问题出现的原因：

.cpp

#if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_ANDROID)

JniMethodInfo methodInfo2;

/// 中文转码

std::string strInfo("图片已保存在系统相册中。");

XtcUtils::GBKToUTF8(strInfo);

const char\* info = strInfo.c\_str();

if (JniHelper::getMethodInfo(methodInfo2,

"com.mesrjni.MesrJni",

"showToast",

"(Ljava/lang/String;)V"))

{

methodInfo2.env->CallVoidMethod(jobj, methodInfo2.methodID,

methodInfo2.env->NewStringUTF(info));

}

#else

#endif

.java

代码1：

public void showToast(final String str) {

Handler handler = new Handler();

handler.post(new Runnable() {

@Override

public void run() {

Toast.makeText(getApplicationContext(), str.toString(),

Toast.LENGTH\_SHORT).show();

}

});

}

代码2：

public void showToast(final String str) {

Handler handler = new Handler(getApplicationContext().getMainLooper());

handler.post(new Runnable() {

@Override

public void run() {

Toast.makeText(getApplicationContext(), str.toString(),

Toast.LENGTH\_SHORT).show();

}

});

}

前一段代码，会导致一个[Can't create handler inside thread that has not called Looper.prepare()](http://www.cnblogs.com/sonicit/archive/2013/01/13/2858475.html)，至于具体，可参考文章：<http://www.cnblogs.com/sonicit/archive/2013/01/13/2858475.html>，这里详细讲解了原因。

上一段代码，调用了函数GBKToUTF8，这个函数是用来c++与java转码问题，我在文章：<http://blog.csdn.net/dionysos_lai/article/details/38389765> 和 <http://blog.csdn.net/dionysos_lai/article/details/40350313> 中详细方法的原理与解决方法。

到此文件读写操作细节，基本阐述完毕。

**截屏细节：**

截屏的效果，大家要参考的话，可以玩一下游戏Toca Boca 游戏《FairyTales》中截屏效果，ps：这个游戏的装扮做的真好。

对于截屏具体技术，上面已经分析过了。但是，由于这次游戏出现一个新问题，就是使用了CCLayerColor类，在游戏关灯时，会罩一个黑色但有透明的图层，表示晚上的感觉。下面两张图片，分别表示白天和夜晚。



白天 夜晚

由于夜晚使用的是的CCLayerColor，本身带有透明度，因此在截屏时，获取的图片跟看起来不大一样。如果没有做混合处理，实际截取出来的图片，类似如下所示；



未做混合处理，截屏效果

如何处理呢？就必须使用混出处理。什么是混出呢？这里可以参考以前写的一篇文章：

<http://blog.csdn.net/dionysos_lai/article/details/39030081> 这篇文章开头详细介绍了混合原理，以及如何做擦除效果，ps：类似刮彩票效果，也可以使用这个原理来做。

对于混合处理，比较关键是设置混合参数。这里提供一个网站：<http://www.andersriggelsen.dk/glblendfunc.php>， 这个网站可以非常方面的帮助我们调试参数。

根据调试结果，混合参数为：源--GL\_ONE，目标--GL\_ONE\_MINUS\_SRC\_ALPHA。

下面给出详细代码：

void HomeScene::ShowPicAlbum()

{

SimpleAudioEngine::sharedEngine()->playEffect(CHR\_MF\_CAMERA\_EXPOSURE);

//根据要截取屏幕大小，定义一个渲染纹理

m\_renderTextureSplot = CCRenderTexture::create(1280, 800);

CCScene\* pCurScene = CCDirector::sharedDirector()->getRunningScene();

CCPoint ancPos = pCurScene->getAnchorPoint();

visibleNode();

//渲染纹理开始捕捉

int outOpa = m\_colorlayerTreeOut->getOpacity();

int inOpa = m\_colorlayerTreeIn->getOpacity();

ccBlendFunc blendFunc = { GL\_ONE, GL\_ONE\_MINUS\_SRC\_ALPHA}; ///< 设置混合模式, 这里不设置混合的话，关灯是，遮罩会有一种透明关系。

ccBlendFunc blendFuncB = m\_colorlayerTreeIn->getBlendFunc();

m\_colorlayerTreeOut->setBlendFunc(blendFunc);

m\_colorlayerTreeIn->setBlendFunc(blendFunc);

m\_renderTextureSplot->begin();

//绘制当前场景

pCurScene->visit();

//结束

……(下面是一些动作代码)

}