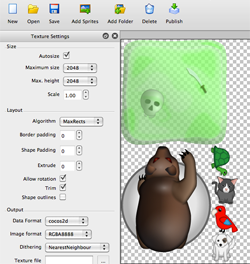
使用Texture Packer和像素格式来优spritesheet(cocos2d)

前言：这篇文章是我翻译的第一篇关于cocos2d的文章，我在翻译的时候尽量按原文意思来，但难免会加入自己一些理解进去。还有这篇文章所采用的xcode版本是3.2.5，用xcode4.0的朋友可能实现起来有点出入，我会在文章的最后给出解决办法。

　　文章原来出处：<http://www.raywenderlich.com/2361/how-to-create-and-optimize-sprite-sheets-in-cocos2d-with-texture-packer-and-pixel-formats>

　　在cocos2d里面，为了使你的游戏获得最佳性能，你需要把许多小的sprite图片组合到一张大图里面，这张大图就叫做sprite sheet。

　　如果你使用cocos2d已经有一段时间了的话，你可能已经使用过了一款叫做[Zwoptex](http://zwoptexapp.com/" \t "_blank)的工具来帮你生成sprite sheet。Zwoptex是一个非常棒的工具--我在我的很多程序里面使用它，而且确实帮我节省了很多时间。

　　然后，这里有一款新的工具，叫做Texture Packer.它类似于Zwoptex，也能创建sprite sheet，但是它还有一些非常方便的、很神奇的特性。

　　这篇文章将以一种教程的形式，讲述如何在cocos2d游戏开发使用Texture　Packer，同时，你还将学习到如何使用像素格式（pixel formats）、Texture Packer 如何智能地让你的游戏加载速度更快，运行更流畅，而且还能够在游戏界面看起来很不错的前提下使用尽可能少的内存。

　　澄清：我在发博客之前就从Texture Packer这个工具的作者手中拿到了license key。我当时并没有保证我会发一篇博文来回报他，但是，在我使用这个工具一段时间之后，它确实为我的应用程序减少了很多需要加载的内存，因此我爱上了这款工具。所以，我想让你们都了解它。

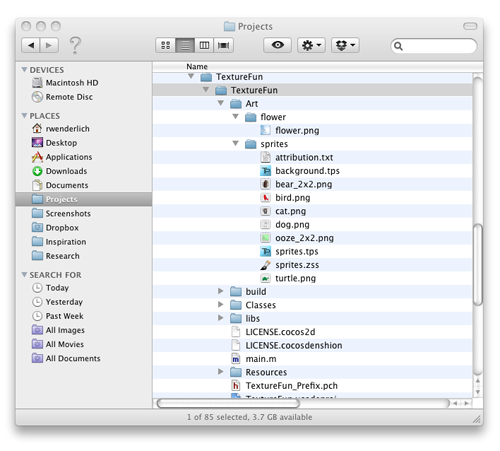
　　这篇教程是为那些熟悉cocos2d的人写的。如果你对cocos2d完全陌生的话，你应该从[“怎样做一个简单的iphone应用程序”](http://www.raywenderlich.com/1797/how-to-create-a-simple-iphone-app-tutorial-part-1" \t "_blank)系列和[其它一些cocos2d的教程](http://www.raywenderlich.com/tag/cocos2d" \t "_blank)开始。（目前这些链接依然是e文，但随着我翻译进度的前进，这些也会相应的更新。）

## 开始

　　首先，确保你安装了最新版本的cocos2d（在写作这篇文章的时候，版本号是 [Cocos2D v0.99.5-rc1](http://www.cocos2d-iphone.org/forum/topic/11150)，在翻译这篇文章的时候，已经是 [cocos2d-iphone-1.0.0-beta.tar.gz](http://cocos2d-iphone.googlecode.com/files/cocos2d-iphone-1.0.0-beta.tar.gz)了）。获得最新版本非常重要，因为新版本里面增加了对一些新的图片格式的支持，而这恰恰是这篇文章后面要用到的。

一旦你安装完之后，你就可以启动XCode，然后使用cocos2d应用程序模板来新建一个工程并把它命名为TextureFun。

　　下一步，你需要一些图片来制作sprite　sheets。你可以下载这些我收集来的[样例图片](http://www.cnblogs.com/andyque/admin/sample%20artwork" \t "_blank)，然后解压缩，并把整个解压缩后的目录拖到你的TextureFun工程的一个子文件夹下面，如下图所示：

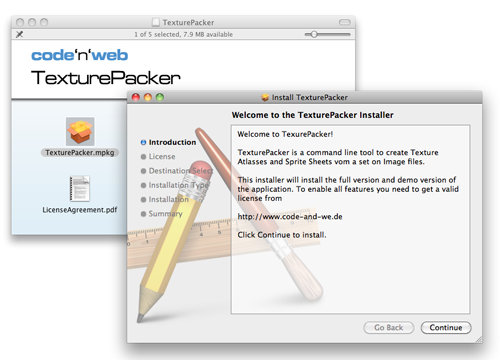


　　好，现在你拥有一个工程模板和一些样例图片以，是时候使用TexturePacker来制作spritesheet了！

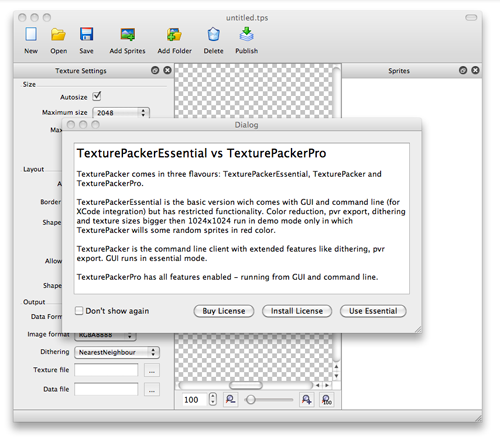
## 使用TexturePacker来创建spritesheet

　　你需要做的第一件事情就是下载 [Texture Packer](http://texturepacker.com/download/)的免费版本。请注意，你并不需要购买任何东西，对于这篇教程来说你只需要免费版本就足够了。

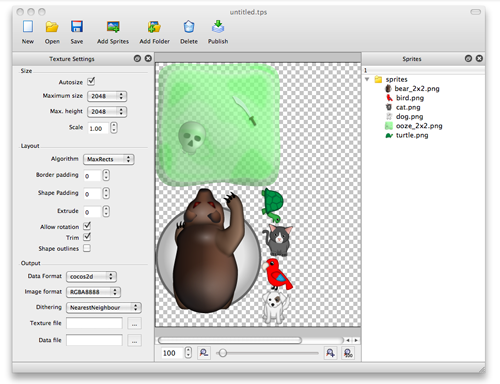
当你下载完之后双击，然后点击“TexturePacker.mpkg”，接下来就会弹出一个窗口，然后按照提示一步步安装在你的mac上。



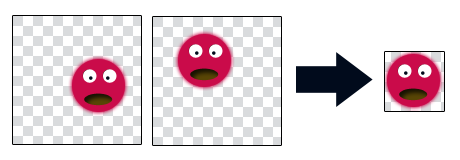
　　在你完成安装之后，在你的Application文件夹下面找到它并运行。当你看到第一个提示窗口出现的时候，选择“ Use Essentia”（免费版本）继续。



　　现在，点击工具栏上的“Add　Folder”按钮，然后选择TextureFun\Art\sprites文件夹。Texture　Packer将会加载图片并且智能地把这些图片布局在spritesheet中，如下图所示：



　　另一方面，你能够看到所有导入到纹理集（Texture　Atlas）中的图片，当你选中某一个的时候会看到有一个边框--另一个非常方便的功能！你也能够把鼠标停留在精灵（sprite）上面，看它是否创建走样（alias）（走样（alias）指那些被裁剪（cropping）之后的图像看起来实际上是一样的）。如下图所示：



　　顺便说一下通过点击“Add　Folder”按钮来添加图片的一些注意事项。首先，当你像这样通过增加文件夹的方式来增加精灵后，Texture　Packer并不是对每一个精灵增加一个索引（reference），而是对整个文件夹增加了一个索引。这意味着，当你以后向这个文件夹中增加精灵之后，下一次你运行Texture　Packer的时候，它会重新根据文件夹下面的所有的精灵来创建spritesheet--多么方便啊！

　　同时，你也不需要非得把所有的精灵都放在同一个根文件夹下面，你可以按照自己的方式把这些精灵放在不同的子目录下（比如sprites\animals，sprites\monsters），之后当你从cocos2d中引用它们的时候只需指定相对路径即可。

　　最后，需要注意的是，你可以包含多个精灵文件夹--这也是一个非常方便的功能，尤其是在你的游戏非常大的时候，你可以为每一个关卡制作一个spritesheet。

　　好了，现在让我们来看一下软件左边的一些选项。通过这些选项，你能够配置spritesheet的大小、布局和输出格式。首先，让我们来快速浏览一遍控制大小和布局的选项：

* Autosize （默认） – 这个选项会为你的spritesheet挑选最小的２的指数倍的大小。这是一个非常方便的特性，因为它可以省去你自己去计算spritesheet大小的时间。
* Min/max size 让你为你的spritesheet指定一个最大值。如果你想设置特定大小的spritesheet的时候，这也是一个非常方便的特性。（因为对于特定的设备来说，你可不想超过设备能够支持的最大限制，比如2代touch最大支持texture大小为1024\*1024）
* Scale 让你可以保存一个比原始图片尺寸要大一点、或者小一点的spritesheet。比如，如果你想在spritesheet中加载“2x"的图片（也即为Retina-display设备或者ipad创建的）。但是你同时也想为不支持高清显示的iphone和touch制作spritesheet，这时候只需要设置scale为０.５就可以了。也就是说，只需要美工提供高清显示的图片，用这个软件可以自己为你生成高清和普清的图片。
* Algorithm TexturePacker里面目前唯一支持的算法就是MaxRects，即按精灵尺寸大小排列，但是这个算法效果非常好，因此你不用管它。
* Border/shape padding 即在spritesheet里面，设置精灵与精灵之间的间隔。如果你在你的游戏当中看到精灵的旁边有一些“杂图”的时候，你就可以增加这个精灵之间的间隔。
* Extrude 精灵边界的重复像素个数. 这个与间隔是相对应的--如果你在你的精灵的边边上看到一些透明的小点点，你就可以通过把这个值设设置大一点。
* Trim 通过移除精灵四周的透明区域使之更好地放在spritesheet中去。不要担心，这些透明的区域仅仅是为了使spritesheet里面的精灵紧凑一点。--当你从cocos2d里面去读取这些精灵的时候，这些透明区域仍然在寻里。（因为，有些情况下，你可能需要这些信息来确定精灵的位置）
* Shape outlines 把这个选项打开，那么就能看到精灵的边边。这在调试的时候非常有用。

　　对于spritesheet来说，上面提到的各个选项的默认值，你一个也不需要改变--因为它们本来就已经很好了。然后，在输出部分，你需要改变一些设置。但是在讲到那个之前，让我们先谈一谈cocos2d中的像素格式。

## cocos2d和像素格式

　　在cocos2d里面，理解像素格式非常重要。因为，像素格式会影响在你的游戏中加载一张图片到底需要多少内存。因为游戏通常要加载大量的图片资源，所以你要尽可能充分利用移动设备上面非常少的可用物理内存。

　　默认情况下面，当你在cocos2d里面加载一张图片的时候，对于每一个像素点使用４个byte来表示--１个byte（８位）代表red，另外３个byte分别代表green、blue和alpha透明通道。这个就简称RGBA8888。

　　因此，如果你使用默认的像素格式来加载图片的话，你可以通过下面的公式来计算出将要消耗多少内存来加载：

　　图像宽度（width）×图像高度（height）×每一个像素的位数（bytes　per　pixel）　=　内存大小

　　此时，如果你有一张５１２×５１２的图片，那么当你使用默认的像素格式去加载它的话，那么将耗费

　　５１２×５１２×４=１MB（好多啊！）

　　这里，我们以Iphone３G为例。它总共只有１２８兆内存，但是系统就要占掉一大半，还有其它一些程序也要使用一些内存，实际可用的内存更少。对于单独一张spritesheet来说那确实足够了。可是想像一下你有许许多多的spritesheet，而且游戏里面经常需要大量的spritesheet！

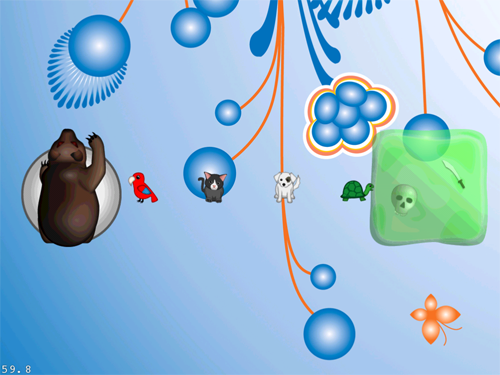
　　这里就需要让像素格式来帮忙了。你可以为图片的每个像素点指定更小的字节来保存图片。（比如每个像素点２个字节，即每个像素点１６位），这种方式就能够在图片质量和内存消耗之间取得一个很好的平衡点。

　　通常，你是在你的游戏看起来还ok的提前下，尽可能少地使用内存。背景图片就非常适用用８位或者１６位来存储，而精灵则一般要用１６位或者３２位。对于更多可选的像素格式和适用的场合，你可以参考Riq（cocos2d的作者）的一篇文章： [understanding pixel format guide](http://www.cocos2d-iphone.org/archives/61).（理解像素格式向导）

　　顺便说一下，如果你注意看窗口的右下角，你会看到Texture　Packer会基于你当前选择的像素格式计算出这张spritesheet所消耗的内存大小，因此你不必手动计算了。：）

## 像素格式和抖动

　　很多时候，当你使用较小的像素格式来加载图片的时候，你会发现图片的质量也在相应的降低。这时你会看到图像存在许多颜色的梯度变化。这里有一个例子，展示了当你使用像素格式RBGA4444去显示一张图片的时候会是什么样子：

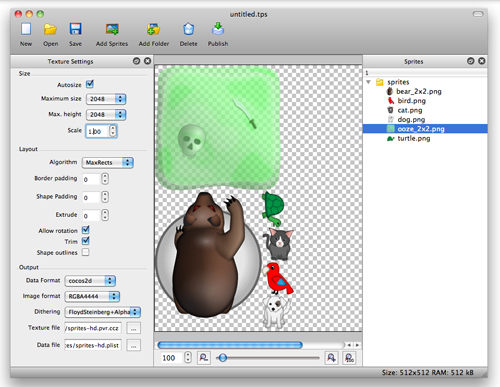


　　看到没有，图像上面有许许多多的“条条”和颜色梯度变化，特别是熊和绿色的框框那里。

　　这时，你可能想重新设计你的图片来确保使用更少的梯度，或者使用大一点的像素格式。但是，在这里，TexturePacker实现了另外一个杀手锏功能--图像抖动。

　　当你使用TexturePacker来保存spritesheet的时候，你可以指定目标像素格式为RGBA4444，然后选择“dithering　method”。这个默认选项会修改一些你的图像的颜色，但是当有梯度变化或者其它一些会带来问题的颜色以后，图像看起来就会非常糟糕。

　　继续，我们为spritesheet选择RBGA4444格式，然后改变抖动选项为“FloydSteinberg＋Alpha”。Texture　Packer将会在动态修改你的图片，而且马上显示出效果来。和上面的图片相比，是不是好看多了？



　　现在让我们保存这个spritesheet。点击Texture　file旁边的“...”，在弹出的对话框中选择TextureFun\Resouces目录，然后命名为“sprites-hd.pvr.ccz”。然后，TexturePack会自动为我们在Data　file那里生成相应的plist文件路径。并且会命名为“sprites-hd.plist”，这个名字是根据前面你提要的名字来命名的。

　　“但是，等一下！”，你可能会说，“为什么是pvr.ccz？！”。好吧，我很高兴你会这样问。。。

## PVRs和压缩

　　PVR图像是专门为ios设备上面的PowerVR图形芯片指定的图像容器。它们在ios设备上非常好用，因为可以直接加载到显卡上面，而不需要经过中间的转化。

　　PVR图像也可以包含许多种不同像素格式的图像数据。之前，cocos2d仅仅支持一些用  sdk指定的 [texturetool app](http://developer.apple.com/library/ios/" \l "documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/TextureTool/TextureTool.html)来创建的[图片格式](http://www.cocos2d-iphone.org/forum/topic/5984" \t "_blank)，不过后来cocos2d已经[扩展了许多格式](http://www.cocos2d-iphone.org/forum/topic/8073)了。

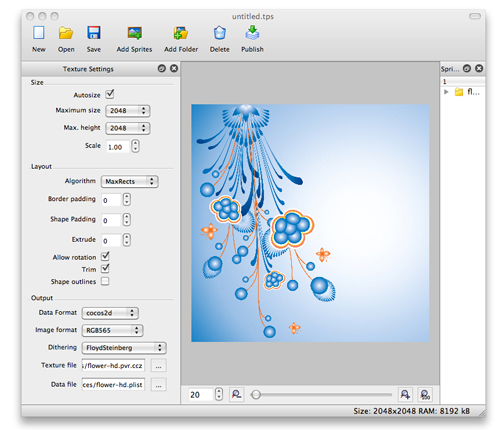
　　而且，最近cocos2d更新到了可以支持压缩了的pvr图像格式pvr.ccz。使用这种图片格式的好处有两点：一、可以使你的应用程序更小，因为图片是压缩过了的。二、你的游戏能够启动地更快。

　　总而言之，对于spritesheet来说，你可能通过指定１６位的像素格式来减少内存消耗，同时保存为pvr.ccz格式来使程序加载速度更快。最后，点击“Publish”按钮，你的spritesheet和属性列表文件就生成好了。Teture　Packer会提示你，一些精灵将会创建成红色（因为你使用的是免费版本）。

## 优化背景图片

　　现在，让我们也来加载并优化一下我们的背景图片。点击new创建一个新的Texture　Packer窗口，然后点击“Add　Folder”，并且选择“TextureFun\Art\flower”文件夹。

　　把图片格式改成RBG565（对于大的图片来说，你可能需要更好的质量），然后改变抖动方法为“FloydSteinberg”（为什么不是FloydSteinberg＋Alpha呢？因为像素格式是RBG565，没有了Alpha通道）。然后指定保存texture　file的路径为“TextureFun\Resouces\flower-hd.pvr.ccz”。最后，点击“Publish”，关闭警告信息，这时你的屏幕看起来会是下面这样：



## 在cocos2d里面使用spritesheet

　　现在回到我们的项目，右键点击Resources，然后选择“Add\Existing　Files...”。选择flower-hd.plist， lower-hd.pvr.ccz, sprites-hd.plist, and sprites-hd.pvr.ccz。同时，确保没有选中“ Copy items into destination group’s folder (if needed)”,然后单击完成。

　　下一步，打开HelloWorldScene.m，并且用下面的代码替换掉你的init方法里的内容：

-(id) init  
{  
if( (self=[super init] )) {  
CGSize winSize = [CCDirector sharedDirector].winSize;  
  
[CCTexture2D setDefaultAlphaPixelFormat:kCCTexture2DPixelFormat\_RGB565];   
CCSprite \* background = [CCSprite spriteWithFile:@"flower-hd.pvr.ccz"];  
background.anchorPoint = ccp(0,0);  
[self addChild:background];  
  
// More coming here soon...  
}  
return self;  
}

　　你需要做的第一件事情就是加载背景图片。首先，你告诉cocos2d使用RBG565的像素格式（你正在为你的背景图片的每个像素使用８位），然后调用spriteWithFile从磁盘上加载pvr.ccz格式的图片。注意，这里你并不需要把它当作一个spritesheet（比如，加载plist文件），因为这里“spritesheet”就只有一张图片。

　　注意，其实你在加载pvr.ccz格式的文件的时候并不需要指定像素格式，因为这个文件格式本身就包含了这样一些信息。但是，我们还是显示地在这里指　定了像素格式，因为如果你加载png格式的图片的话，（png格式图片总是保存为每个像素３２位，尽管你可能会使用不同的像素格式把它加载到内存里）。

　　下面，让我们在“more　coming　here　soon”注释的地方添加下面的代码：

[CCTexture2D setDefaultAlphaPixelFormat:kCCTexture2DPixelFormat\_RGBA4444];  
CCSpriteBatchNode \*spritesBgNode;  
spritesBgNode = [CCSpriteBatchNode batchNodeWithFile:@"sprites-hd.pvr.ccz"];  
[self addChild:spritesBgNode];   
[[CCSpriteFrameCache sharedSpriteFrameCache] addSpriteFramesWithFile:@"sprites-hd.plist"];

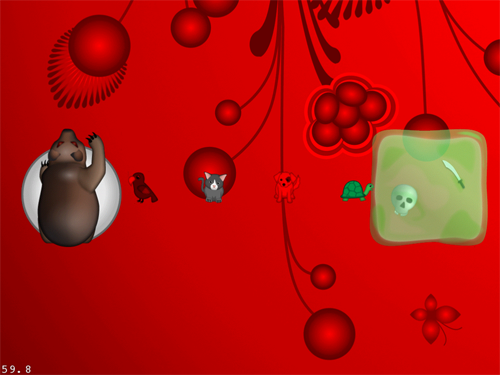
　　这里把像素格式设置为RBGA4444（你为主精灵所使用的每个像素点１６位的像素格式），然后为这个spritesheet创建一个batch　node。你也需要加载plist文件，把每一个精灵对应的帧（frame）加载到精灵帧缓冲区（sprite　frame　cache）中。

　　最后，紧接着上面加入下面的代码：

NSArray \*images = [NSArray arrayWithObjects:@"bear\_2x2.png", @"bird.png", @"cat.png", @"dog.png", @"turtle.png", @"ooze\_2x2.png", nil];   
for(int i = 0; i < images.count; ++i) {  
NSString \*image = [images objectAtIndex:i];  
float offsetFraction = ((float)(i+1))/(images.count+1);   
CGPoint spriteOffset = ccp(winSize.width\*offsetFraction, winSize.height/2);  
CCSprite \*sprite = [CCSprite spriteWithSpriteFrameName:image];  
sprite.position = spriteOffset;  
[spritesBgNode addChild:sprite];  
}  
  
[CCTexture2D setDefaultAlphaPixelFormat:kCCTexture2DPixelFormat\_Default];

　　这个循环遍历spritesheet中的所有的图片，并把他们合理地旋转在屏幕上面。

　　如果你使用iPad来编译并运行你的代码的话，你会得到下面的运行结果：



　　这个不就是你想实现的效果吗--记住，这里的红色仅仅因为你使用的是免费版本。（ps：正式版本也不贵，１００多块人民币可以搞定）

　　但是真正让人着迷的是那些你看不到的东西。

　　在背后，你的应用程序加载速度会明显比以前快很多。而且使用更少的内存，更让人心动的是，它看起来还是那么棒！而这些功能使用Texture　Packer都可以很容易的完成。

## 不相信我？

　　当我写这篇文章的时候，我做了一系列的简单的测试，从最好的情形到最坏的情形，来测试到底我的程序是如何运转的。下面是我得出的一些结论：

* 做最原始的事情。使用默认像素格式单个单个精灵地加载，不使用任何spritesheet。大约花费了０.７３秒钟加载，消耗大约２６兆内存。而且当你添加更多的精灵进去的时候，游戏就开始卡了。
* 使用默认的像素格式，并且使用spritesheet：前进了一大步。这样会使游戏性能更好，同时也会减少内存消耗（因为你会把所有的精灵加载到一张大小的spritesheet的，而opengl使用纹理的大小都是２的指数幂，如果你一张精灵的大小是３２０×２００的话，那么opengl会创建５１２×５１２的纹理来加载精灵，这样就有很多空白的地方。实际上就是浪费了内存。）
* 使用不含抖动的Zwoptex并保存为png格式，同时减少像素格式：这样可以大幅地减少内存消耗，大约只需要１５兆左右）。但是却增加了程序的启动时间，上升到大约１秒钟。我认为可能是由于不得不改变颜色缓冲的缘故吧。另外，图像显示的效果并没有在“像素格式和抖动”一节中的截屏效果那么好。
* 使用通过Texture　Packer创建的抖动过的spritesheet并且保存为pvr.ccz格式：这在启动时间和显示效果上都前进了一大步！（启动时间大约只有０.３１秒左右，内存也只需要大约１７兆左右，我认为这可能是由于*[内存泄漏](http://www.cocos2d-iphone.org/forum/topic/11214" \l "post-63675)*，这个问题在现在的版本中已经解决了。

　　好了，如果你按照上面所讲的最佳实践来做的话，我想你在大部分情况下都会做得非常好。：）如果你想看看我写的测试程序，也想拿来跑一跑的话，[可以点此下载](http://www.raywenderlich.com/downloads/MemoryTest2.zip)。

## Texture Packer 和XCode集成

　　当使用Texture　Packer的时候，你可以像这里介绍的一样使用GUI工具，但是你还可以把它集成到Xcode构建过程中去。这样你每一次编译的时候，它都会自动地（如果没有更改，就不会更新）为了更新spritesheet。

　　如果你过去用cocos2d写过游戏的话，你肯定明白一遍又一遍地重新生成你的spritesheet是那么的烦人！虽然每次可能都只需要几秒钟的时间，但是老是这样重复地做这样的事，确实很烦。

　　因此，让我们更方便地构建我们的工程吧--这里只需要花几秒钟时间，但是却可以为你以后节省大量的时间。右键点击“Resources”，选择“Add\New　File...“，然后选择　Mac　OS　X\Other\Shell Script，然后选择下一点。并命名为PackTextures.sh，单击完成。

　　然后使用下面的代码替换掉PackTextures.sh里面的内容：

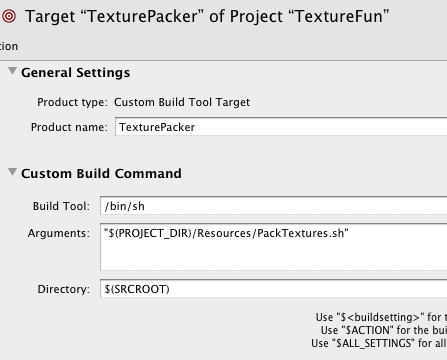
#!/bin/sh  
  
TP="/usr/local/bin/TexturePacker"  
  
if [ "${ACTION}" = "clean" ]  
then  
echo "cleaning..."  
  
rm resources/sprites-hd.pvr.ccz  
rm resources/sprites-hd.plist  
  
rm resources/flower-hd.pvr.ccz  
rm resources/flower-hd.plist  
  
# ....  
# add all files to be removed in clean phase  
# ....  
else  
echo "building..."  
  
# create hd assets  
${TP} --smart-update \  
--format cocos2d \  
--data resources/sprites-hd.plist \  
--sheet resources/sprites-hd.pvr.ccz \  
--dither-fs-alpha \  
--opt RGBA4444 \  
Art/sprites/\*.png  
  
${TP} --smart-update \  
--format cocos2d \  
--data resources/flower-hd.plist \  
--sheet resources/flower-hd.pvr.ccz \  
--dither-fs-alpha \  
--opt RGB565 \  
Art/flower/\*.png  
  
# ....  
# add other sheets to create here  
# ....  
fi  
exit 0

　　所有Texture　Packer　GUI界面能够做的事情，命令行工具也能做。如果你在命令行里面输入“TexturePacker”，你将会看到一系列它能够接收的参数说明。

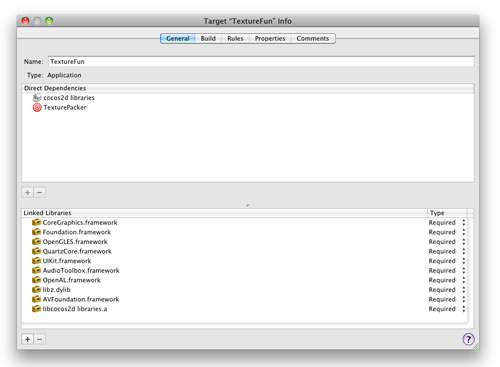
　　这个脚本仅仅通过运行TexturePacker来从你的Art目录下读取精灵文件并创建spritesheet--就像你之前用GUI工具所做的一样。你可以通过查看TexturePacker命令行工具帮助来获得更多有关每个参数具体的用法。

　　接下来，你需要让你的工程在编译的时间能够运行这个脚本。右键点击Targets，选择“Add\New　Target...”，然后选择“External　Target”（不是Shell　Script　Target），然后点击下一步，重命名这个Target为TexturePacker，最后点击Finish。

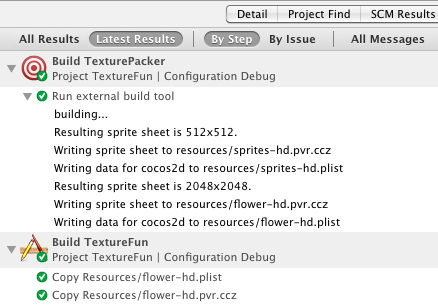
　　然后在你新建的target上面双击，然后把里面的内容设置成下图所示：



　　最后一步就是把这个target设置成你的程序的一个依赖。在TextureFun　target上面双击，然后选择General　tab，再在Direct　Dependencies窗口下面选择＋按钮。然后从列表中选择TexturePacker，最后单击Add　Target。如图下所示：



编译并运行你的程序，你将会从你的构建结果信息中看到Texture　Packer的一些输出信息，由此来判断是否一切运转良好。



　　如果你看到这些输出信息的话，那么意味着如果你想要增加一些新的文件到spritesheet中，你只需要把这些新的文件拖到指定的文件下面（这里是Art目录），然后重新编译一下，那么就会自动生成新的spritesheet。相反，如果你要移除一些精灵图片，再重新编译一下，也ok。是不是非常方便？

## Texture Packer vs. Zwoptex

　　首先，让我们再说一遍，我是Zwoptex的超级粉丝。我认为Robert已经做了一件非常好的事情来把这些东西整合到一起，而且我老实说，如果没有这个工具，cocos2d不会走到今天这一步！

　　然后，当我们来比较Texture　Packer和Zwoptex的特性的时候，Texture　Packer似乎包含了Zwoptex９０％的功能。但是，Texture　Packer有三个杀手锏级别的功能是Zwoptex所没有的。

* 抖动，抖动，抖动. 噢，我是多么喜欢抖动啊！在过去，有时我想使用比较低的像素格式，但是我不能，因为它看起来效果很不好。但是Texture　Packer内置的抖动功能使得图片看起来还是那么棒，尽管此时的图片质量很低。
* pvr.ccz 支持. 我喜欢这个特性. 它确实能够使游戏启动得更快, 而且会使你的可执行程序更小. 现在我不用花很长时间来上传和下载我以前写的应用程序了，我可以很快的下载并更新。
* 命令行工具支持. 一旦你花一点时间把Texture　Packer集成到你的Xcode中去，你将会热爱生活。它是如此地方便，特别是在整个开发过程中，美工对图片改来改去的时候。

　　尽管Zwoptex非常棒, 但是它目前为此还是没有提供我这里列举的这些功能特性。

　　Texture Packer 有一点点贵 ($17.95 vs. Zwoptex’s $14.95), 但是，我认为多花这点钱值得。而且就像  [Steffen Itterheim](http://www.learn-cocos2d.com/tag/gui/)所说, ２个工具都有所长，都做了一件了不起的工作。

## 接下来该怎么做呢？

[这里](http://www.raywenderlich.com/downloads/TextureFun.zip)有上面的教程中所使用的示例代码。

　　你是一个Texture　Packer粉丝　还是一个Zwoptex粉丝呢？不妨在下面一起聊聊吧！或者你有其它一些很好的策略来高效地在cocos2d里面加载纹理的话，请让我知道。：）

ps：使用xcode4的朋友，主要需要注意的就是新增加的Target的参数的一些设置。因为xcode４生成的项目文件夹下并不是直接包含了所有的项目文件，而是一个工程文件和另外一个和项目名字一模一样的文件夹，然后所有的项目有关的源文件和资源文件都放在那个目录下，所以，我们增加的Target里的设置信息应该改成：

"$(PROJECT\_DIR)/TextureFun/Resources/PackTexture.sh"

$(SRCROOT)/TextureFun

其实可能需要注意的就是自动生成普清spritesheet，选中AutoSD选项就行了。

但是PackTexture.sh脚本里要相应增加

 ${TP} --smart-update \  
--format cocos2d \  
--data resources/sprites-hd.plist \  
--sheet resources/sprites-hd.pvr.ccz \  
--dither-fs-alpha \  
--opt RGBA4444 \

--auto-sd \ #只要增加这一行就可以自动生成普清的图片了  
Art/sprites/\*.png

还有一些软件的设置选项，大家可以参照软件自带的帮助文档。

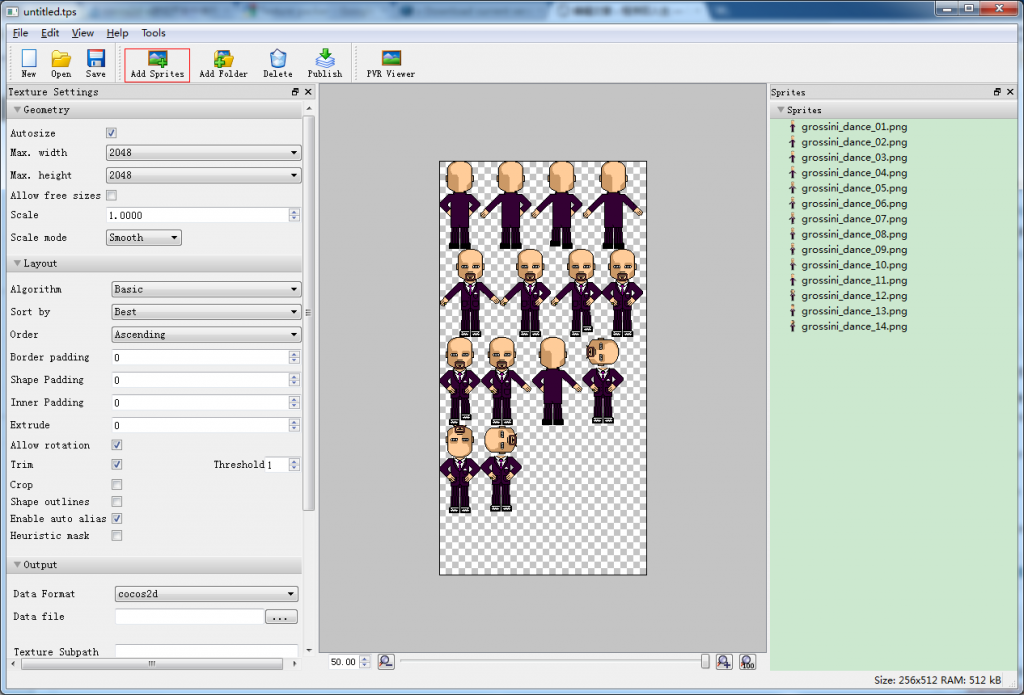
如果大家在实践的过程中，遇到一些问题，可以留言。

第二篇

[cocos2d-x开发游戏时，使用Texture packer来合并图片](http://blog.csdn.net/vagrxie/article/details/6715416)

[Texture packer](http://www.texturepacker.com/)的确比Zwoptex更加强大，功能更多，跨平台，在使用cocos2d-x在win32下开发的时候，能够在win32下使用的优点就更加明显了。何况Zwoptex以前是完全免费的，在用户多了以后坑爹般的开始收费了，连个可用的免费版本都没有（只有自己保留的老版本可用），而Texture packer一直有可用的免费版本，虽然Pro版本贵的离谱（对于中国人来说,100多块钱的工具太贵了，但是对于美国人来说又很便宜）。另外，假如你是博客作者或者是框架开发者，可以向作者申请免费的license,我的确收到了。有意思的是，作者后来以做SEO为由，要求我添加以[Sprite Sheet Maker](http://texturepacker.com/sprite-sheet" \t "_blank)为文字的链接，链接到他们网站，真让人感叹，天下没有免费的午餐啊~~~~

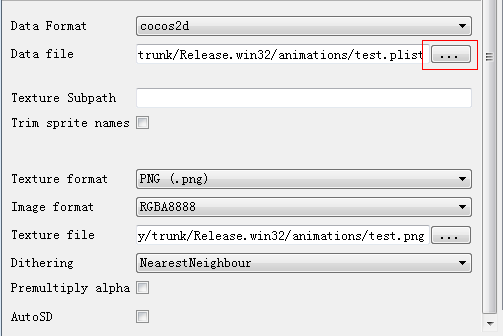
Texture packer的主界面如下：

[](http://www.jtianling.com/wp-content/uploads/2011/08/texturepacker1.png)

**1.Add Sprites**

上图中的样子就是用Add sprite功能，添加了cocos2d的示例图片后的样子。texture packer能够自动的用较为优化的算法来排列图片，使得占用空间尽量的少，作为初级版本，没有一些包括裁剪在内的高级功能（在强行使用高级功能后，会有提示，也可以导出图片，但是会随机在某些图片上印上一些英文文字）

**2.设置导出文件的位置**

[](http://184.82.230.158/wp-content/uploads/2011/08/texturepacker2.png)

这个位置是一个绝对位置，此时可以保存一份texture packer的工程。这个工程会记录这个导出的位置。有一点用户体验非常好的地方是设置了plist的位置后，texture的为止texture packer会自动生成一个。

**3.导出**

点击publish按钮即可。将来在原始图片更改的时候，直接载入这个工程，然后也只需要再次点击public按钮即可，会同时生成plist和texture文件。相当易用。

Enable auto alias非常有用，可以将相同的图片自动合并，对于三维导出的序列帧和flash自动导出的序列帧动画，常常可以节省很多空间。

到目前为止，这已经很强大了。但是Texture packer的强大不仅如此，手工编辑再方便，也不如自动化来的快~~~~

Texture packer在安装后，在安装目录下（windows)会有一个叫texture packer的可运行文件，直接将刚才生成的tps文件作为参数传进去，就能自动的生成~~~再配合VS或者XCode的工程配置，完全可以做到生成程序的时候对texture的全自动化处理。（不过这个需要pro的license）

**4.程序中的使用**

先通过以下接口预加载

void CCSpriteFrameCache::addSpriteFramesWithFile(const char \*pszPlist)

然后通过以下接口使用：

CCSpriteFrame\* CCSpriteFrameCache::spriteFrameByName(const char \*pszName)

CCSprite\* CCSpriteFrameCache::createSpriteWithFrameName(const char \*pszName)

CCSpriteFrameCache本身是个单件。