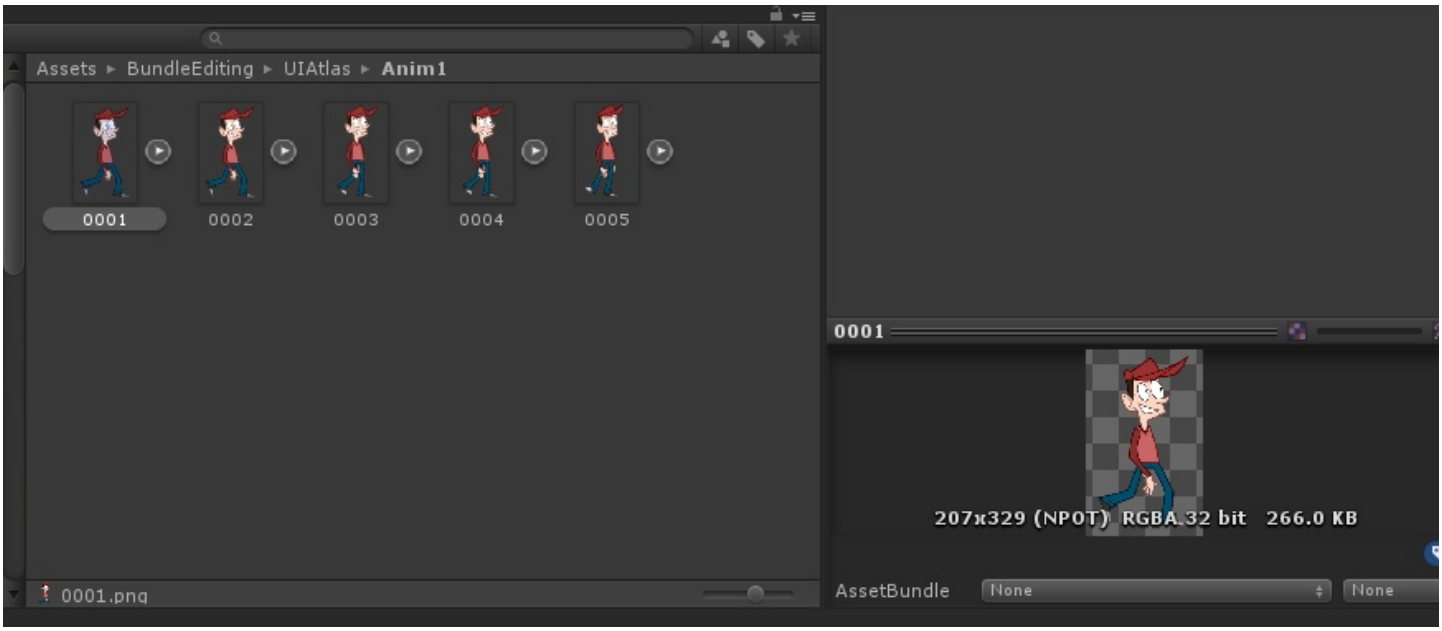


干货：Unity游戏开发图片纹理压缩方案

陈小霖 K...
公众号「小霖的认知旅行」，不懂技术的运营不是好的产品经理
关注他

Unity3D引擎对纹理的处理是智能的：不论你放入的是PNG，PSD还是TGA，它们都会被自动转换成Unity自己的Texture2D格式。

在Texture2D的设置选项中，你可以针对不同的平台，设置不同的压缩格式，如IOS设置成PVRTC4，Android平台设置成RGBA16等。

嗯，非常的智能。

但是，在一些进阶的使用中，一些情况是难以满足的。

比如，我们NGUI的图集纹理，在Android平台，使用ETC1纹理+Alpha通道图的方式；iOS平台，使用PVRTC4的纹理。

个别图片纹理，要求清晰度较高的，使用RGBA16，但是使用RGBA16的渐变显示图片却惨不忍睹；

一些要求高保真的，则需要直接使用最高质量的RGBA32格式。

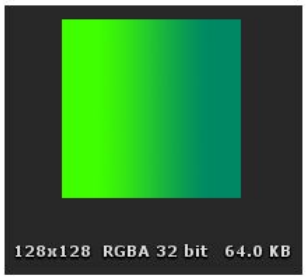
很多时候，随着项目的复杂需求发展，单纯的Unity纹理管理已经无法满足我们的需求了。这时候，往往需要我们做一些额外工作。

总结一下我自己的纹理压缩方案：

纹理压缩的策略

手游开发（Android/iOS）中，我会使用3个级别的压缩程度：高清晰无压缩、中清晰中压缩、低清晰高压缩；4种压缩方法：RGBA32， RGBA16+Dithering，ETC1+Alpha，PVRTC4。一般足够应付大部分的需求了。

高清晰无压缩 - RGBA32



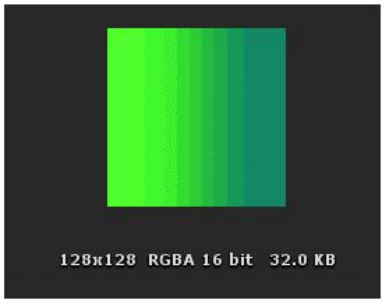
Unity RGBA32 - 高清晰无压缩.png

RGBA32等同于原图了，优点是清晰、与原图一致，缺点是内存占用十分大；对于一些美术要求最好清晰度的图片，是首选。

要注意一些png图片，在硬盘中占用几KB，怎么在Unity中显示却变大？因为Unity显示的是Texture大小，是实际运行时占用内存的大小，而png却是一种压缩显示格式；可以这样理解，png类似于zip格式，是一个压缩文件，只不过在运行时会自动解压解析罢了。

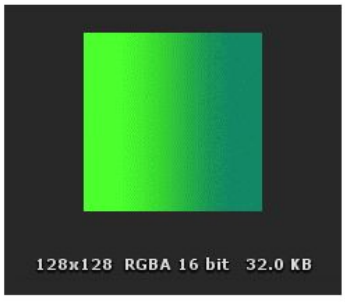
中清晰中压缩 - RGBA16 + Dithering

RGBA16 + Dithering



Unity RGBA16，不抖动处理的渐变图片惨不忍睹

既然叫RGBA16，自然就是RGBA32的阉割版。
对于一些采用渐变的图片，从RGBA32转换成RGBA16，能明显的看出颜色的层叠变化，如上图。

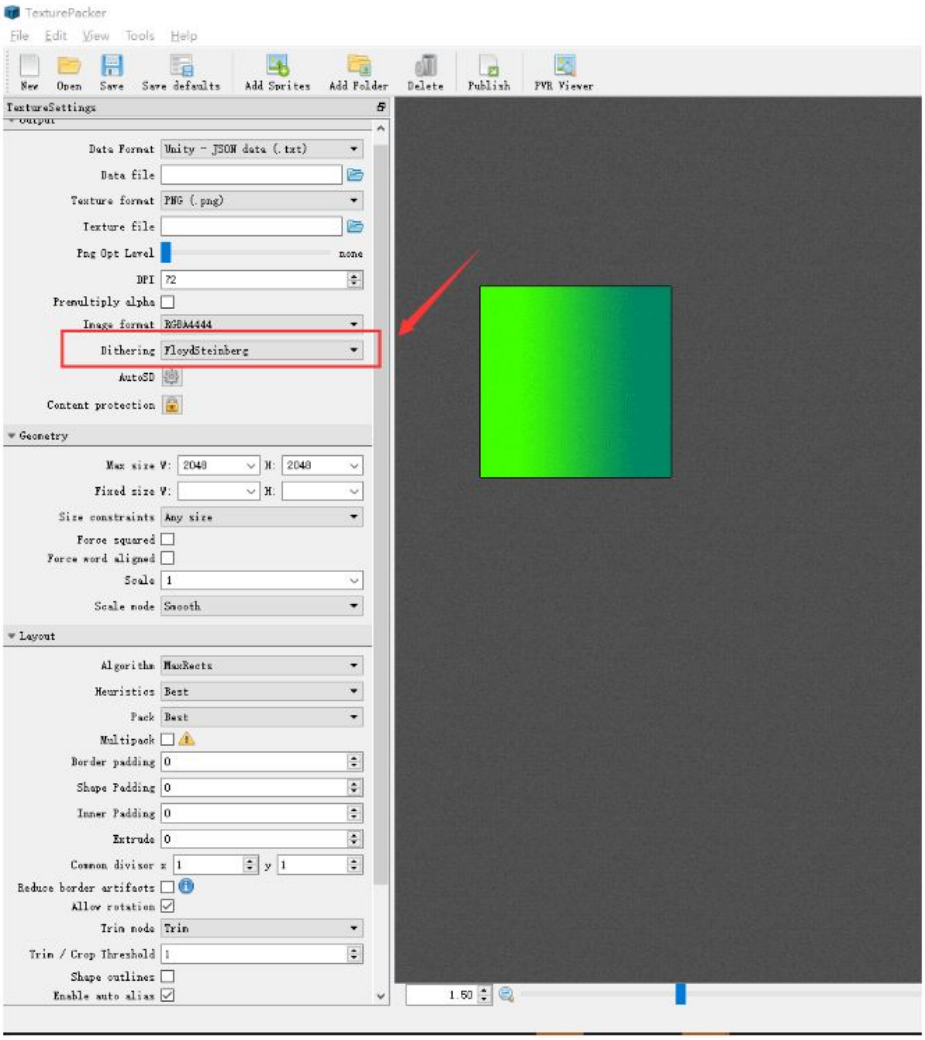


采用Floyd Steinberg抖动处理后，除非放大，否则肉眼基本看不出区别

RGBA16的优点，内存占用是RGBA32的1/2；搭配上Dithering抖动，在原尺寸下看清晰度一模一样；

缺点，Unity原生不支持Dithering抖动，需要自己做工具对图片做处理；对于需要放大、拉伸的图片，Dithering抖动的支持不好，会有非常明显的颗粒感。

如何进行Dithering抖动？



Texture Packer工具中Image Format选择RGBA4444，Dithering选择FloydSteinberg

在我的项目中，TexturePacker具有非常重要的作用，像UI的图集生成，预先生成好正方形的IOS PVRTC4图集和非正方形的Android ETC1图集、 缩放原图50%等工作都由TexturePacker完成。

同样，对图像进行抖动处理，也是预先在TexturePacker使用FloydSteinberg算法进行图像抖动，再在Unity中导入使用。

TexturePacker提供命令行工具，可以做成自动化的工具。具体方法这里不详述。

RGB16

Unity RGB16

而RGB16，是主要针对一些，不带透明通道，同时长宽又不是2的次方的图片；对于这些图片，使用RGB16可以降低一半的内存，但是效果会略逊于RG

当然了，RGB16其实也是可以搭配抖动，也能提升显示效果；但同样的Dithering抖动对拉伸放大是不友好的。

低清晰高压缩 - ETC1+Alpha/PVRTC4

很多初学者都会疑惑，为什么游戏开发中经常看到一些图片，需要设置成2的次方？因为像ETC1、PVRTC4等这类在内存中无需解压、直接被GPU支持的格式，占用内存极低，而且性能效率也是最好的。

但是，相对RGBA32，还是能肉眼看出质量有所下降的。

ETC1

ETC1+Alpha一般应用在Android版的UI图集中，ETC1不带透明通道，所以需要外挂一张同样是ETC1格式的Alpha通道图。方法是，在原RGBA32的原图中，提取RGB生成第一张ETC1，再提取A通道，填充另一张ETC1的R通道；游戏运行时，Shader将两张ETC1图片进行混合。

> 生成Alpha通道图的方法可参考：
> 【改进版】Unity工程里图片的RGB和Alpha通道的分离 - 一只飞鸟的自白 - 博客频道 - CSDN.NET
>
> 后来，由于不想基于Unity API生成透明图，我生成Alpha通道图的方法。我使用Python的一个png.py库，用Python脚本来处理：

```python
def png_to_alpha(filename, output_file):
    """
    把PNG的Alpha单独填满一张Png的r(不带a)
    """
    print 'Processing alpha from ... %s' % filename

    pic = png.Reader(filename)
    new_colors = [] # get all alpha color

    data = pic.asRGBA()
    for rowPixels in data[2]:
        new_row = []
        # print rowPixels
        for i in range(1, len(rowPixels)+1):
            if i % 4 == 0: # get alpha
                new_row.append(rowPixels[i-1])
                new_row.append(0)
                new_row.append(0)
                # new_row.append(255)

        new_colors.append(new_row)

    png.from_array(new_colors, 'RGB').save(output_file)

    print 'Success from png `%s`, to alpha pic `%s`' % (filename, output_file)
```

png.py生成alpha图

要配合ETC1+Alpha，还需要Shader支持，这里参考直接修改NGUI的Unlit/Transparent With Colored的Shader。

```
fixed4 frag (v2f i) : COLOR
{
    fixed4 col = tex2D(_MainTex, i.texcoord);
    col.a = tex2D(_AlphaTex, i.texcoord).r;
```

PVRTC4

PVRTC4在Unity中是直接支持的，不过要注意的细节是，它必须是二次方正方形，也就是说，长宽在二次方的同时，还必须要相等。

几种纹理格式的对比

格式	内存占用	质量	透明	二次方大小	建议使用场合
RGBA32	1	★★★★★	有	无需	清晰度要求极高
RGBA16+Dithering	1/2	★★★★	有	无需	UI、头像、卡牌、不会进行拉伸放大
RGBA16	1/2	★★★	有	无需	UI、头像、卡牌，不带渐变，颜色不丰富，需要拉伸放大
RGB16+Dithering	1/2	★★★★	无	无需	UI、头像、卡牌、不透明、不会进行拉伸放大
RGB16	1/2	★★★	无	无需	UI、头像、卡牌、不透明、不渐变，不会进行拉伸放大
RGB(ETC1) + Alpha(ETC1)	1/4	★★★	有	需要二次方，长宽可不一样	尽可能默认使用，在质量不满足时再考虑使用上边的格式
RGB(ETC1)	1/8	★★★	无	需要二次方，长宽可不一样	尽可能默认使用，在质量不满足时再考虑使用上边的格式
PVRTC4	1/8	★★	无	需要二次方正方形，长宽一样	尽可能默认使用，在质量不满足时再考虑使用上边的格式

> - 内存占用，相对于RGBA32做比较
> - 质量星级，更多是本人感受，仅供参考

一个商业项目，混搭多种纹理格式是在所难免的事情。把项目纹理划分成高、中、低三种质量需求，是这个方案的落脚点。

在项目中，尽可能是使用ETC1和PVRTV4等GPU直接支持的图片格式，不仅内存占用低、性能也更好；当出现质量不及格时，再逐步的提升压缩格式，来满足需要。

编辑于 2017-02-13

Unity（游戏引擎） 游戏开发 游戏编程