昵称： 马三小伙儿
园龄： 7年7个月
粉丝： 770
关注： 107
+加关注

< 2023年3月 >

日	一	二	三	四	五	六
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

我的标签

游戏开发(59)
Unity(36)
c/c++(17)
python(12)
C#(12)
Linux(11)
pygame(10)
Lua(10)
cocos2d-x(9)
Linux编程(7)
更多

积分与排名

积分 - 271515
排名 - 3361

阅读排行榜

1. 【python游戏编程之旅】第六篇---pygame中的Sprite（精灵）模块和加载动画(69762)
2. 【python游戏编程之旅】第九篇---嗷大喵快跑小游戏开发实例(69319)

【Unity游戏开发】SpriteAtlas与AssetBundle最佳食用方案

阅读目录

- 一、简介
- 二、图集的往事今生
- 三、实践工程
- 四、总结

回到顶部

一、简介

在Unity步入2019.4以后，新版的SpriteAtlas日趋完善，已经完全可以在商业项目中使用了。但是纵观网络平台上，许多关于SpriteAtlas的文章还停留在2018的初版时期，其中许多解释在现在看来都是过时的，许多国内知名论坛上还有各种错误的结论和教程。如果按照这种错误的教程来使用SpriteAtlas的话，一来有可能造成图集和资源的冗余，二来会导致享受不到新版图集带来的开发便利从而影响了效率。因此进行SpriteAtlas和AssetBundle的正确配合使用调研实在必行。

回到顶部

二、图集的往事今生

1.NGUI和TP时代

早在NGUI时代就已经有了图集的概念了，与UGUI先使用后制作图集的工作流程不同，NGUI是先制作图集再使用。

先制作图集再使用的时候，在反复迭代开发的过程中，图集打包容易引起冲突。同时，实现规划的图集随着需求的变更可能需要重新规划，重新规划以后，又要重新打图集，之前做好的界面又要重新修复引用，这中间的工作量，经历过的人都知道。

对于NGUI这种对迭代开发十分不友好的工作流，UGUI带来的改进可以说是广大开发者的福音。

在访问NGUI图集的图元时，我们需要先加载图集，然后再从这个图集中获取单个图元，用伪代码表示大概是这个流程：

```
var spriteAtlas = Resources.Load("spriteAtlasName");

var sprite = spriteAtlas:GetSprite("spriteName");
```

2.UGUI时代的旧SpritePacker和新SpriteAtlas

和NGUI不同，UGUI访问图集的图元不需要我们主动去加载图集，再从图集中获取图元。当我们加载图集的图元时，图集会被引擎自动加载，图集的释放也是自动完成的，不需要针对图集编写任何的业务逻辑代码，而有关图集的处理工作都是放在了资源后处理、资源分组和打包上。

简单来说，就是运行时写的那些业务逻辑不需要关心我这个图元属于哪一张图集，属于哪一个AssetBundle，直接以散图的形式去使用、去获取就可以了，比如代码可以写成下面这样：

```
var sprite = Resources.Load("Bag/spriteName");
```

UGUI图集的先使用后制作主要有两种作业模式，也就是旧的SpritePacker和新的SpriteAtlas模式：

1.旧的SpritePacker模式，给Sprite设置PackingTag，Tag相同的会被分配到同一个图集中，如下图所示：

3. 【Unity3d游戏开发】浅谈UGUI中的Canvas以及三种画布渲染模式(61079)

4. 【Unity游戏开发】SDK接入与集成——小白入门篇(49315)

5. 【python游戏编程之旅】第一篇---初识pygame(43066)

评论排行榜

1. 【年终总结】马三北漂记之8102年总结(66)

2. 【Unity游戏开发】SpriteAtlas与AssetBundle最佳食用方案(48)

3. 【python游戏编程之旅】第九篇---嗷大喵快跑小游戏开发实例(41)

4. 【年终总结】马三京沪漂流记之2019年总结(40)

5. 【Unity游戏开发】SDK接入与集成——小白入门篇(35)

推荐排行榜

1. 【python游戏编程之旅】第九篇---嗷大喵快跑小游戏开发实例(37)

2. 【年终总结】马三北漂记之8102年总结(30)

3. 【年终总结】马三京沪漂流记之2019年总结(22)

4. 【Unity3d游戏开发】浅谈UGUI中的Canvas以及三种画布渲染模式(13)

5. 【小白学C#】浅谈.NET中的IL代码(12)

最新评论

1. Re:【Unity3d游戏开发】游戏中的贝塞尔曲线以及其在Unity中的实现

画图工具是啥~求告知

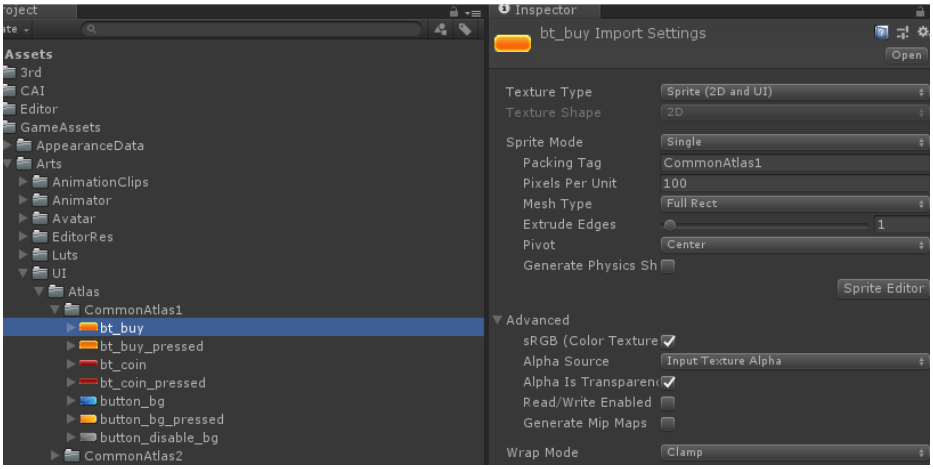
--章校长

2. Re:【Unity游戏开发】SpriteAtlas与AssetBundle最佳食用方案

大佬，为什么我将散图打成一个AB,使用AssetsStudio查看会有若干Sprite,一个Texture2D,另外还有一个SpriteAtlas和一个AssetBundle？？

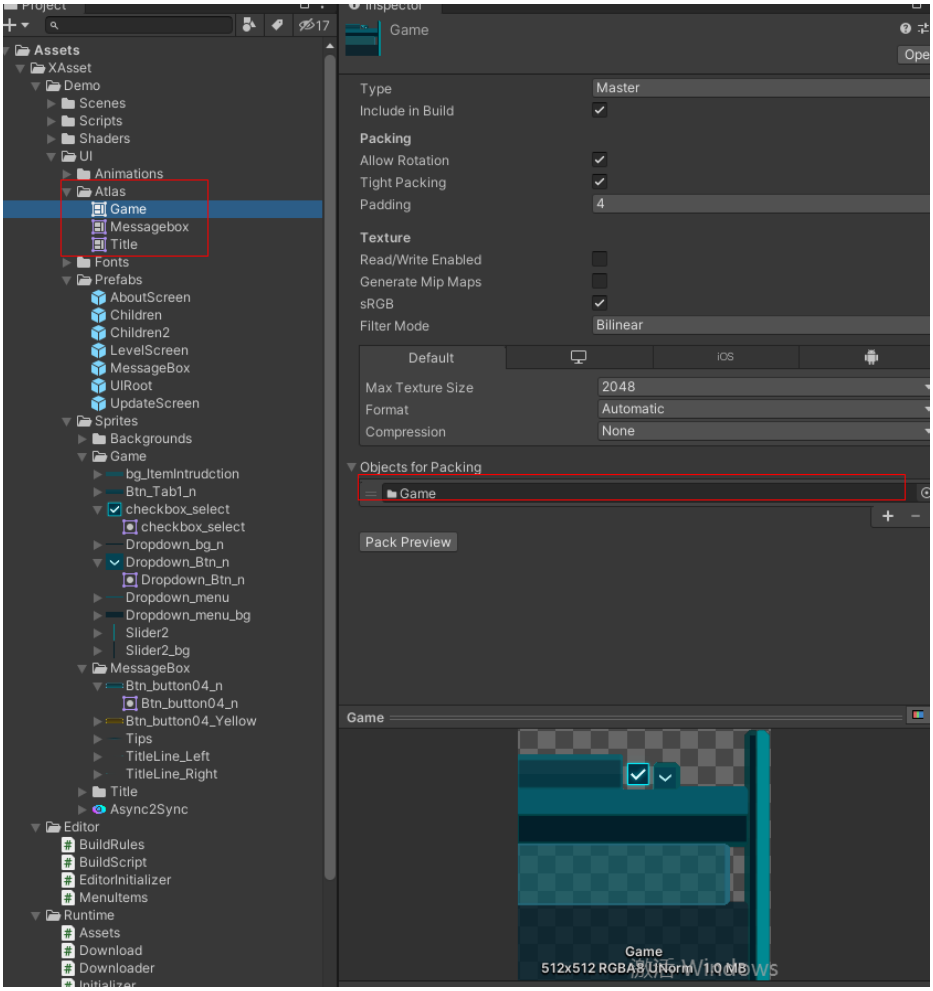
--一只向攻城狮进化的程序猿

3. Re:【Unity游戏开发】SpriteAtlas与AssetBundle最佳食用方案



2.新版的SpriteAtlas相比之前的SpritePacker做了更多的优化，比如可以实时查看图集的大小，图集里面元素的排列布局，并且增加了LateBinding等特性

我们可以通过Asset/Create/Sprite Atlas去创建一个图集，图集创建以后可以通过拖拽的方式去选择要打包的对象，如下图所示：



Objects for Pakcing指定了图集中需要打包的图元，可以指定单独的文件，也可以指定整个文件夹。

在新的作业模式中，每一个图集都是用一个单独的SpriteAtlas管理起来，图集的格式也被定义在这个资源中，预览单张图集不用像旧版的那样需要把所有的图集都集中在一个SpritePacker的编辑器窗口中预览，而是可以实时在Inspector窗口可视化预览。旧版的图集管理方式在图集数量多的时候，查找不方便还非常卡，新版的作业方式是一种分而治之的理念，更为方便和快捷。

需要注意的，UGUI的图集，无论新旧，在构建AssetBundle的时候，同一个图集内的所有图元都要放在同一个AssetBundle中，否则，如果同一个图集的图元被分散到多个AssetBundle中，那么每一个AssetBundle都会包含一份这个图集的Copy，最终的结果就是包体冗余、内存膨胀和加载耗时等问题。

如果是看的CSDN等网络上的教程的话，多半会让你不要勾选SpriteAtlas的Include In Build选项，说是会造成图集的双份冗余。但是这种说法实际上早就过时了，这个Bug早已经在

你好，请问有测试过sbp（addressable）打包嘛，我测试下来2020、2021的unity，sbp1.19.6版本，打包是冗余的，散图、图集全打进包了。而legacy打包管线，打出来的是正确的...

--Sarofc

4. Re:【Unity游戏开发】SpriteAtlas与AssetBundle最佳食用方案

你好，请问你是怎么卸载的，我卸载完bundle,看profile里面的Texture2D还在

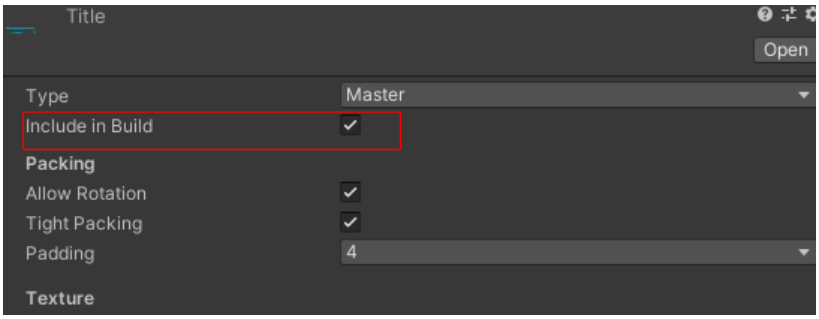
--Lp千阳

5. Re:【Unity3d游戏开发】游戏中的贝塞尔曲线以及其在Unity中的实现

22年才看到这篇博客,写得很好,很有收获,感谢博主!新账号还不能点赞!

--LinVicissitude

Unity2018.4.6中修复了，所以我们在使用中放心大胆地勾选Include In Build就好了，这样也可以避免使用LateBinding。



Fixes

- Android: Fixed Application.Quit not correctly quitting the application process, previously it would only destroy Unity runtime, keeping activity alive, that lead to incorrect application resume. (1171368, 1172044)
- Apple TV: Fixed GetKeyDown and GetKeyUp not working correctly with Siri Remote buttons. Due to platform limitation, GetKeyDown, GetKeyUp will work with delay when receiving events from keyboard, see documentation for more info. (1143342, 1160636)
- Apple TV: Fixed pressing menu button on Siri remote not correctly exiting to home screen, if tvOS.Remote.allowExitToHome is set to true. (1134856, 1160637)
- Apple TV: Fixed regression, where clicking B on gamepad would show "JoystickButton0" as pressed. (1151006, 1160635)
- Asset Pipeline: Fixed sprite atlas sprites being included in asset bundles multiple times. (1121868, 1170282)
- Editor: Fixed an issue where tests with the inconclusive result would stop the test run from continue. (1169256)

同样，如果是看了网上的教程的话，也会发现有一些在使用SpriteAtlas时遇到了白图或者不显示的情况，这种情况实际上是对UGUI新图集的工作流不熟悉导致的。经过测试，只要打包的时候勾选图集的Include In Build，然后，不需要主动对SpriteAtlas资产文件进行打包，也不需要写额外的代码，就可以正常使用了，只需要对文件下下的散图进行ab打包即可。实例代码如下：

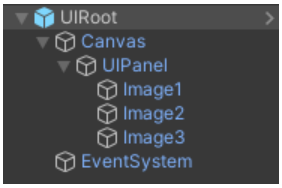
```
string path = "UI/Atlas/MySprite.png"

var sprite = Assets.LoadAsset(path,typeof(Sprite)); //封装好的加载接口

MyImage.image = sprite;
```

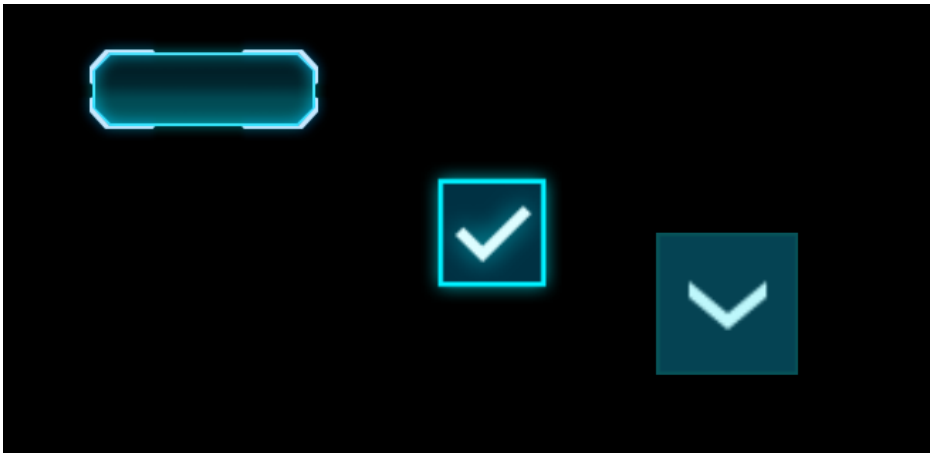
回到顶部

三、实践工程

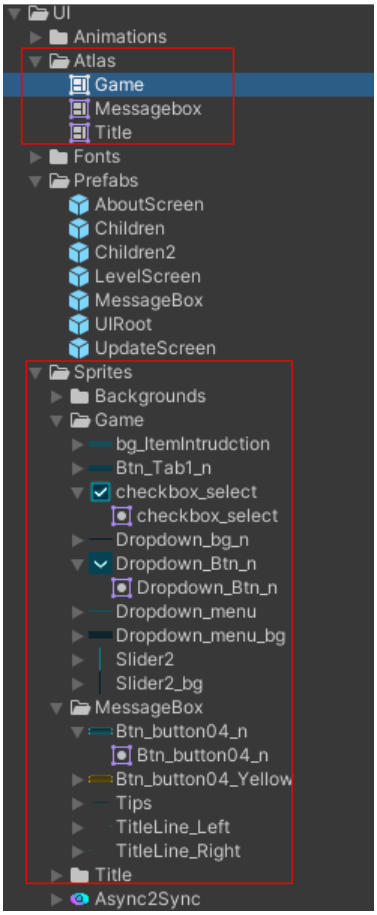
为了佐证上面的一些结论，这里特意配置了一份教练工程。

首先有一个引用了图集中图元的UIPrefab，如下图所示，它上面有三个Image，右边的两张Image分别引用了同一个图集中的图元，我们用它来验证图元是否合批，左边的Image是引用了另外一个图集中的图元，我们用它来对比验证DrawCall：

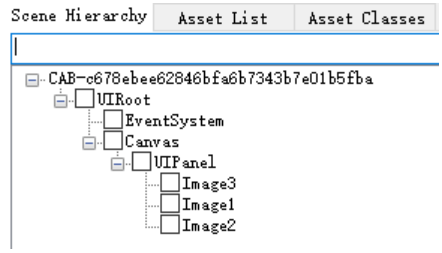


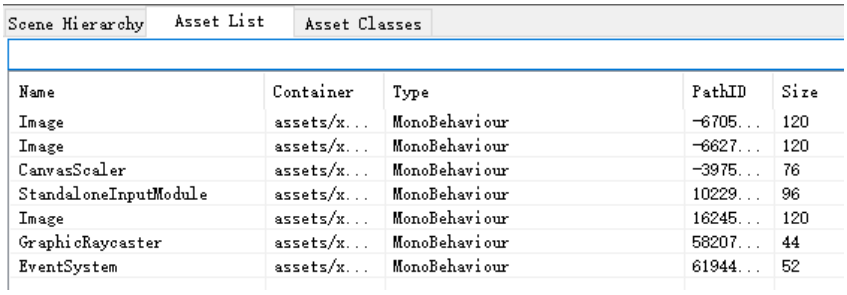


然后这里面有三分SpriteAtlas文件，它们都勾选了Include In Build 但是不参与打包。这三个图集分别管理这Sprites目录下的每个子目录中的散图文件，这些散图文件时需要参与打包的(AssetBundle)。



然后我们进行打ab操作，打出ab以后我们用AssetStudio去验证ab包的内容，看看它有没有冗余情况出现。首先看一下打出来的UIPrefab的ab文件，prefabs_uiroot.bundle，可以看到尽管UIPrefab引用了图集里面的图元，并且图集勾选了Include In Build，但是并没有UIPrefab的ab里面并没有冗余，Unity的确是修复好了这个bug：



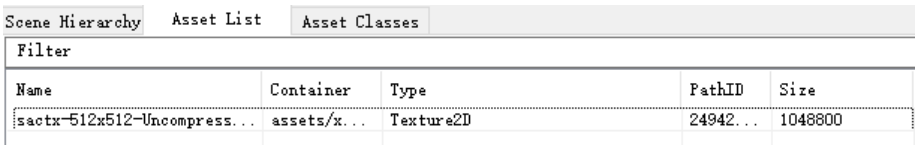


Name	Container	Type	PathID	Size
Image	assets/x...	MonoBehaviour	-6705...	120
Image	assets/x...	MonoBehaviour	-6627...	120
CanvasScaler	assets/x...	MonoBehaviour	-3975...	76
StandaloneInputModule	assets/x...	MonoBehaviour	10229...	96
Image	assets/x...	MonoBehaviour	16245...	120
GraphicRaycaster	assets/x...	MonoBehaviour	58207...	44
EventSystem	assets/x...	MonoBehaviour	61944...	52

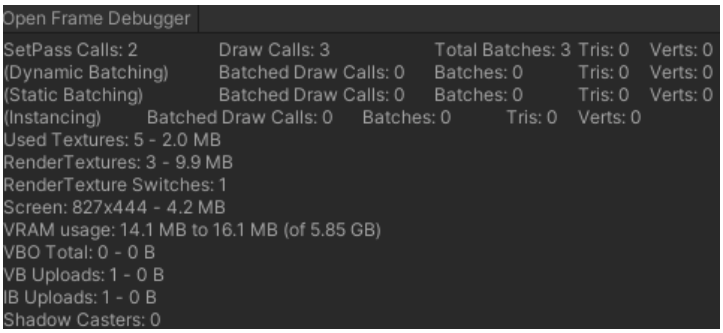
然后再来看一下图集的ab包，因为没有对.spriteatlas文件特意的包，所以打包的实际上按目录划分的散图文件，shared_ui_sprites_game.bundle，可以看到里面只有一张合好的Texture和若干Sprite，也没有多余的冗余文件出来：

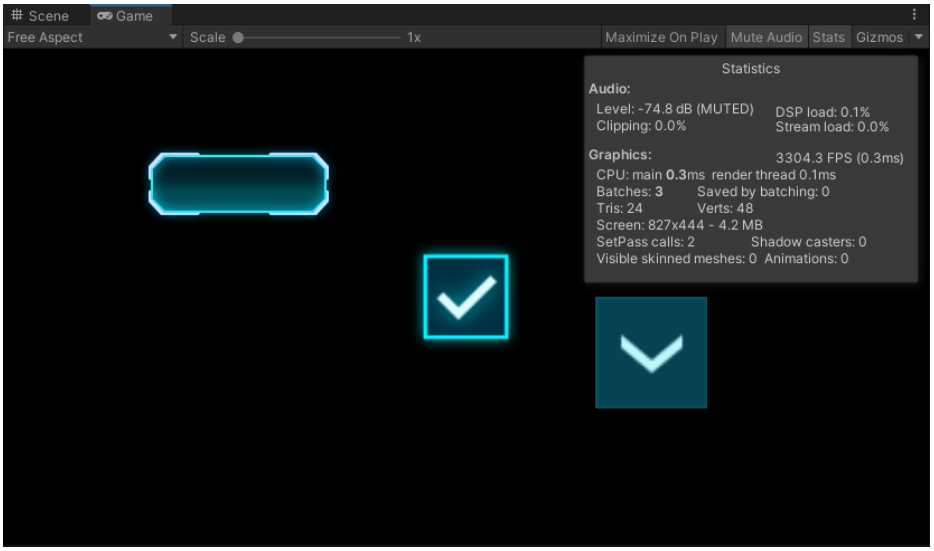
Name	Container	Type	PathID	Size
Slider2_bg	assets/x...	Sprite	-7658...	472
Dropdown_menu_bg	assets/x...	Sprite	-3124...	476
bg_ItemIntrudction	assets/x...	Sprite	-2410...	516
sactx-512x512-Uncompress...	assets/x...	Texture2D	-1049...	104...
Dropdown_menu	assets/x...	Sprite	15217...	476
checkbox_select	assets/x...	Sprite	15240...	540
Dropdown_Btn_n	assets/x...	Sprite	15265...	476
Btn_Tab1_n	assets/x...	Sprite	21544...	472
Slider2	assets/x...	Sprite	51468...	468
Dropdown_bg_n	assets/x...	Sprite	69752...	476

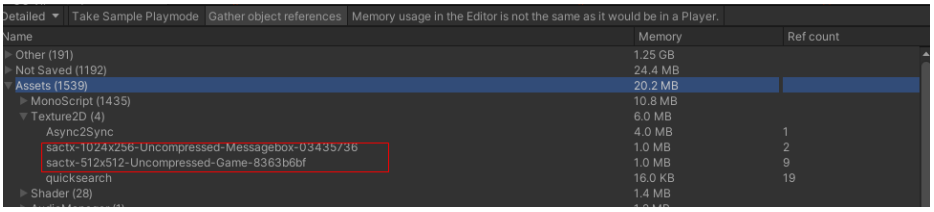
但是，如果我们故意指定对.spritealtas文件也打包，特意指定一下，然后我们再看spriteatlas资源文件打出来的ab，atlas_game.bundle，可以发现里面有个合并好的512x512的图集，这个就造成了的冗余，因为散图目录下已经有一份资源文件了：

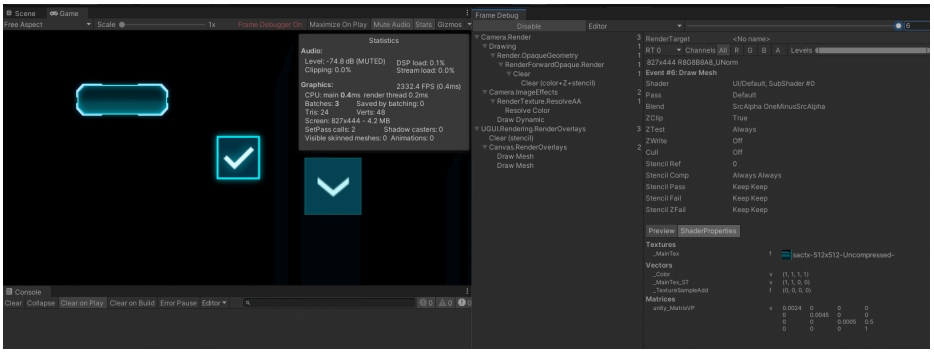


然后我们再看下，只设置了图集，但是不特意针对图集进行导出打包的情况下，UI使用图集内的图元，是否还可以正常的合批，DrawCall是否正常。









通过上面的一些信息，我们可以看到Unity进行了合批操作，DrawCall为3也是正常的。在FrameDebugger里面也可以看到引用了同一图集内的两张图元的Image，也是在一个Batch里面去绘制的。

回到顶部

四、总结

实际上，通过上面的一系列测试，我们可以得出以下结论，新版的SpriteAtlas可以看做是对旧版的SpritePacker的升级，我们在使用的时候仍然是不需要关注图集这个东西的，这里的SpriteAtlas可以看做仅仅是用来作为一种对散图的归纳与整理。当我们加载图集的图元时，图集会被引擎自动加载，图集的释放也是自动完成的，不需要针对图集编写任何的业务逻辑代码，而有关图集的处理工作都是放在了资源后处理、资源分组和打包上。简单来说，就是运行时写的那些业务逻辑不需要关心我这个图元属于哪一张图集，属于哪一个AssetBundle，直接以散图的形式去使用、去获取就可以了。

简单来说遵循以下几点就不会有错了：

- 工作过程中（拼接UI等）放心大胆地以散图的方式去引用UI/Atlas/XXX下的各种图片即可
- SpriteAltas文件需要勾选Include In Build，但是不要特意打包，会造成冗余和包体膨胀
- 代码中动态加载Sprite的地方，直接使用散图的资源路径去加载就可以了，比如：var sprite = Assets.LoadAsset<Sprite>(path);
- 平时工作的机器上SpritePacker的Mode可以设置为Disable，这样可以提高效率（开启的话每次Run之前，图集会打包，资源多了以后会卡）。但是在打包机上一定要把Mode设置为Enable for build或者Always enable，这样图元才能被正确地合批
- 同一个图集内的所有图元打包时都要放在同一个AssetBundle中
- Editor下加载ab包中图集，需要将SpritePackerMode切换至AlwaysEnabled状态，否则加载出来的就是白图。（此条感谢海崖提供）

如果觉得本篇博客对您有帮助，可以扫码小小地鼓励下马三，马三会写出更多的好文章，支持微信和支付宝哟！

作者：马三小伙儿
出处：https://www.cnblogs.com/msxh/p/14194756.html
请尊重别人的劳动成果，让分享成为一种美德，欢迎转载。另外，文章在表述和代码方面如有不妥之处，欢迎批评指正。留下你的脚印，欢迎评论！

分类: unity3d, 游戏开发, AssetBundle

马三小伙儿
粉丝 - 770 关注 - 107

8

+加关注

« 上一篇： 【Unity游戏开发】升级Unity2019后，资源管线后处理采坑记录
» 下一篇： 【Unity游戏开发】初探Unity动画优化