**一、综述**

**二、研究成果**

**（一）成果总览**

1.Unity可视化美术资源性能分析工作流；

2.地下城堡3基于2D地图可视化编辑器(未完待续)；

3.地下城堡3基于Spine角色可视化编辑器(未完待续)；

4.局部Bloom的实现方式；

5. Unity空间坐标中的Spine与屏幕坐标下的UI交互的实现;

6. Unity中同时使用Sprite与Spine产生的批次过大问题解决方案;

7. Unity使用Spine结合TexturePacker的Z轴排序错位问题解决方案;

**（二）成果细项**

**1.Unity可视化美术资源性能分析工作流**

1. 研发时限

2019年06月~2019年06月

1. 问题

【名词解释】：OverDraw：在屏幕内同一坐标上，像素与像素的叠加层数；RunTime：代码在游戏运行时候的状态；AlphaBlend ：透明测试；Avg FPS：游戏运行时的平均帧数；moba：移动平台多人对战竞技类型游戏；Shader：Unity内置可编程着色器；

为了让移动端运行时，具有检测美术资源OverDraw和帧数的功能。则必须制作一个RunTime模式下，可以渲染OverDraw和FPS的可视化窗口。而Unity提供一个可以在Scene窗口下查看OverDraw和FPS显示的功能，使用者只能在编辑器内查看。这种方式有如下几个缺点：

1. 在编辑器显示OverDraw如果非常多，其实对电脑下性能是毫不在乎的，反而手机上只要几层全屏OverDraw就受不了了这正是手机GPU的硬伤。那么这里显示的帧数就不具有代表性，而编辑器内想通过OverDraw确定性能和帧数确实不实用的。
2. 不能根据不同机型和不同平台进行测试。
3. Unity提供的帧数显示非常不精准，有时候编辑器里面帧数一会几百帧，一会就只有几十帧，跨度大不准确。
4. 解决方案

利用Unity强大的Shader、相机全局渲染、GUI开发功能，我们通过以下3点解决上述问题。

1.在图像渲染的时候其实不管是否透明的物体，进行叠加都会存在OverDraw。那么使用一种半透颜色，去做AlphaBlend检测判断先后顺序，然后进行不断叠加就会越来越趋近于白色，这种原理来表现OverDraw层数的原理就非常相近了。编写一个Shader来显示当叠加色彩越多的时候就越趋近白色，那么他的OverDraw就越高。移动端上面全屏幕无叠加层数是性能最理想状态，颜色上呈暗橙色；全屏幕叠加层数2层也是普遍移动端机型可承受范围内，颜色呈微亮橙色如此趋近白色；如果出现叠加层数3层，移动端机型就会出现明显降低帧数的情况，需要减少叠加区域的范围，也就是说游戏内不可以出现相机内大范围的3层叠加情况；如果出现4层以上叠加，这是严重的性能问题，就需要美术制作人员重新修改这种资源。

2. 通过Unity的相机全局渲染，把上面完成的Shader渲染到当前游戏的主相机中。

3.通过Unity提供GUI开发，将计算得到间隔时间内Avg FPS可视化显示。除了通过OverDraw模式下颜色方式来判断美术资源性能以外，还可以通过Avg FPS来判断来判断性能，并且通过Avg FPS来判断性能是最准确的方式之一。一般手游画面帧数维持在30帧无波动的状态是最佳的，也是普通手机能承受的状态；性能较好的手机可以开启60帧低延迟模式，一般这用于moba或多人射击类型游戏，但帧数高了意味着对手机硬件消耗大，要考虑到耗电和发热为题，需要更高要求的优化美术资源。

通过下图1和图2设计方案进行帧数计算，并且根据帧数数值渲染不同颜色，那么当帧数显示为绿色资源为佳，显示为黄色次之，显示为红色则为严重消耗性能的资源。

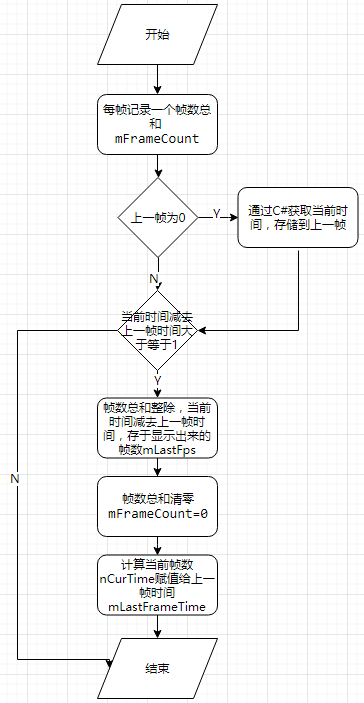


图1

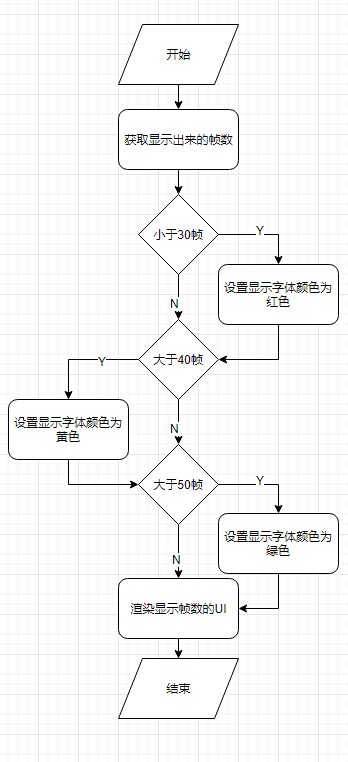


图2

1. 效果展示

以下两张图片分别展现同一相机画面里的美术资源，在关闭性能分析工具，以及开启性能分析工具的画面；如图3，在没有性能分析工具的情况下只能通过运行是否流畅判断当前相机的美术资源OverDraw性能部分的消耗，但是这种方式不能严谨的表现OverDraw性能消耗部分；如图4，开启性能分析工具后，可以通过叠加颜色的亮度判断当前相机的美术资源是否可用，能相对严谨的表现OverDraw性能部分的消耗。



图3 图4

1. 成果

实现了高效的工作流，可以在IOS、Android、WebGL平台内使用真机调试，并根据OverDraw显示的颜色和帧数去判断相机区域内美术资源性能消耗程度，从而减少美术资源因性能问题的返工。

**2.地下城堡3基于2D地图可视化编辑器**

1. 研发时限

2019年05月~2019年11月

1. 问题

【名词解释】： SpineGUI方案：Spine提供显示在Unity GUI上的一种方式，这种方式可以实现Spine与UI交互，但缺点是被转换成GUI的Spine不能使用多材质，只能使用单一材质；Spine：二维动画制作软件； BoneFollower：Spine提供可以跟踪到骨骼节点空间坐标的一种方式；

为了方便地下城堡3的美术人员和策划人员进行2D地图的设计，需要专门为地下城堡3定制一款地图编辑器，除了进行地图设计以外还需要有进行静态阻挡编辑功能、能够在一定范围内随机出现怪物的功能、能够通过事件进行阻挡状态改变的动态阻挡功能。

1. 解决方案

利用Unity强大的编辑器二次开发功能以及编辑器可视化绘制二次开发功能功能，我们结合图1，并通过以下2点解决上述问题。

1.通过Spine的骨骼坐标跟踪获得空间坐标。通过BoneFollower指定到需要追踪的骨骼节点，获取当前节点的空间坐标。

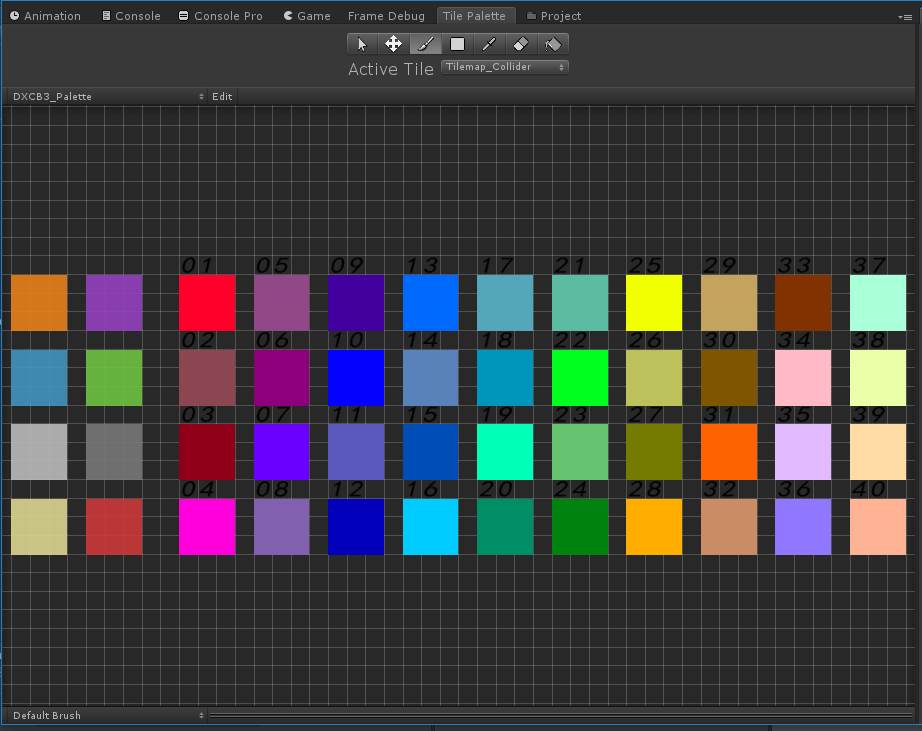
2.通过Unity的空间坐标转换方式生成需要的屏幕坐标。在拿到Spine的骨骼节点空间坐标后，将其通过空间坐标转换方式转成屏幕坐标，就可以将需要跟随骨骼运动的UI对象设置为转换后的屏幕坐标，用这种方法解决了交互的问题。



图1







1. 成果

实现了既可以在Unity中使用Spine的多材质功能，并且能让Spine与UI进行交互，如图2和图3所示，展现了UI元素Button跟随Spine骨骼坐标变换的过程。

**3.地下城堡3基于Spine角色可视化编辑器**

1. 研发时限

2019年12月~2019年12月

地下城堡3基于Spine角色可视化编辑器(未完待续)；

1. 问题

【名词解释】： SpineGUI方案：Spine提供显示在Unity GUI上的一种方式，这种方式可以实现Spine与UI交互，但缺点是被转换成GUI的Spine不能使用多材质，只能使用单一材质；Spine：二维动画制作软件； BoneFollower：Spine提供可以跟踪到骨骼节点空间坐标的一种方式；

为了方便地下城堡3的美术人员和策划人员进行2D地图的设计，需要专门为地下城堡3定制一款地图编辑器，除了进行地图设计以外还需要有进行静态阻挡编辑功能、能够在一定范围内随机出现怪物的功能、能够通过事件进行阻挡状态改变的动态阻挡功能。

1. 解决方案

利用Unity强大的编辑器二次开发功能以及编辑器可视化绘制二次开发功能功能，我们结合图1，并通过以下2点解决上述问题。

1.通过Spine的骨骼坐标跟踪获得空间坐标。通过BoneFollower指定到需要追踪的骨骼节点，获取当前节点的空间坐标。

2.通过Unity的空间坐标转换方式生成需要的屏幕坐标。在拿到Spine的骨骼节点空间坐标后，将其通过空间坐标转换方式转成屏幕坐标，就可以将需要跟随骨骼运动的UI对象设置为转换后的屏幕坐标，用这种方法解决了交互的问题。

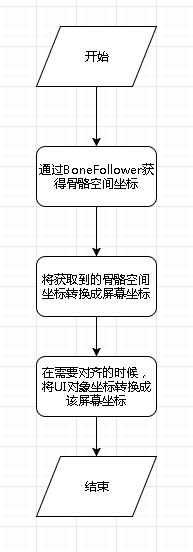


图1

1. 成果

实现了既可以在Unity中使用Spine的多材质功能，并且能让Spine与UI进行交互，如图2和图3所示，展现了UI元素Button跟随Spine骨骼坐标变换的过程。

1. **局部Bloom的实现方式**
2. 研发时限

2019年04月~2019年05月

1. 问题

【名词解释】： Bloom：泛光效果；Avg FPS：游戏运行时的平均帧数； Shader：Unity内置可编程着色器；

为了让美术制作人员可以只对某些物体制作Bloom效果，并且不影响场景内的其他物体。则必须制作一种可以标记某些物体，并采集这些物体的图像进行泛光处理。而传统的全屏幕泛光效果，使用者直接对相机上所有物体泛光。这种方式有如下几个缺点：

1. 全屏泛光消耗较大，一般的移动端机型上展现会出现掉帧卡顿。
2. 有些物体不想被泛光影响，只能通过调暗色调之后调整阈值，导致这些物体暗淡无光难以调到满意效果。
3. 解决方案

利用Unity强大的Rendertexture渲染、stencil功能，我们可以结合图1，并通过以下6点解决上述问题。

1. 创建一个临时的Rendertexture RT1，尺寸为屏幕尺寸，设置为主相机的target。用于保存包含stencil值的场景图。创建一次，且当分辨率更新时重新创建。
2. 创建一个临时的Rendertexture RT2,尺寸为屏幕尺寸，用于获取stencil过滤后的图像。创建一次，且当分辨率更新时重新创建。
3. 一个临时Rendertexture RT3,尺寸初始为屏幕尺寸，用于迭代实现Bloom效果，可调手动调节尺寸。并将RT2的的信息通过Graphics.Blit传递到RT3中，如果RT3尺寸比RT2小，也就起到一个降采样的效果。
4. 一个临时的Rendertexture RT4,和RT3一样的尺寸。用于迭代实现Bloom效果。
5. 迭代完后，将RT3通过Graphics.Blit传递到RT2,在OnRenderImage使用RT2进行最后的混合。
6. 修改目标对象的Shader文件，在其中加入Stencil命令进行标记。

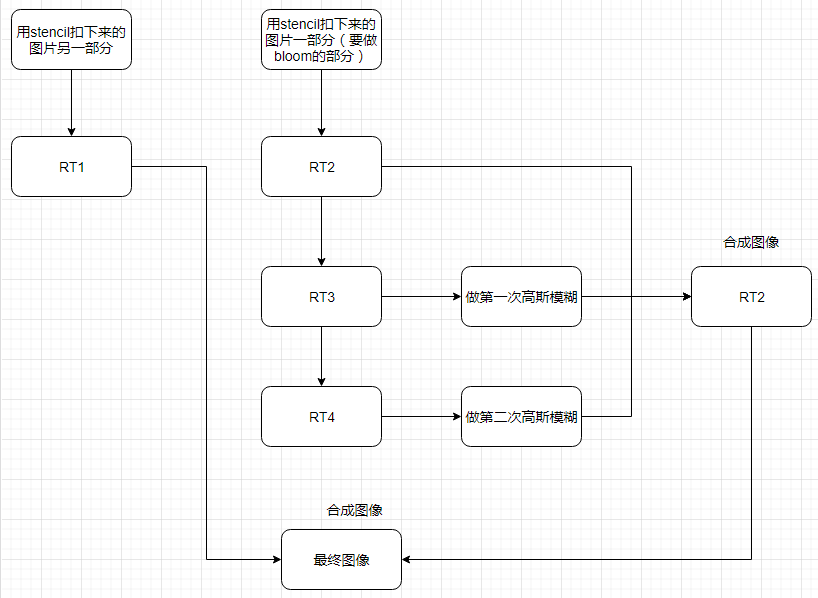


图1

1. 效果展示

以下两张图片分别展现同一相机画面里的美术资源，使用全屏Bloom，以及使用局部Bloom的画面；如图2，使用全屏Bloom时，不但使得美术制作人员难以控制Bloom物体与场景所有物体泛光度的协调，并且性能消耗较大；如图3，使用局部Bloom后，精准控制物体是否泛光，而且不会影响到场景内的其他物体，并且性能消耗较小；为了方便对比，图4展现了没有Bloom效果的对比图。

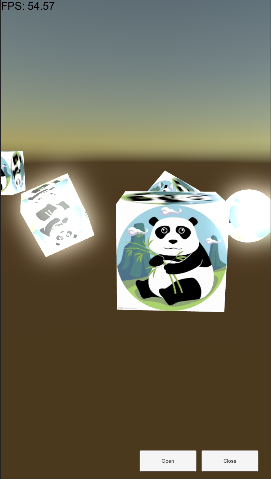
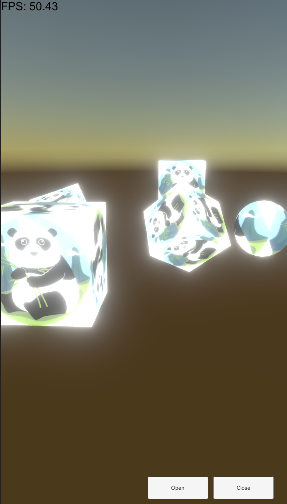


图2 图3

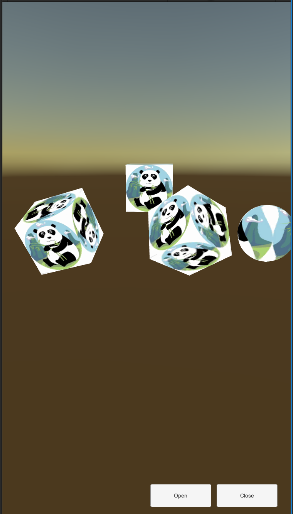


图4

1. 成果

实现了高性能的泛光效果，并且可以支持现有大部分平台包括IOS、Android、WebGL、PC。使美术制作人员可以精准的控制场景下哪些物体需要做泛光效果，在满足美术资源美观的同时，又解决了性能问题。

**5.Unity空间坐标中的Spine与屏幕坐标下的UI交互的实现**

1. 研发时限

2019年06月~2019年10月

1. 问题

【名词解释】： SpineGUI方案：Spine提供显示在Unity GUI上的一种方式，这种方式可以实现Spine与UI交互，但缺点是被转换成GUI的Spine不能使用多材质，只能使用单一材质；Spine：二维动画制作软件； BoneFollower：Spine提供可以跟踪到骨骼节点空间坐标的一种方式；

使用Spine制作游戏时，通常就会涉及到需要Spine与UI进行交互的功能。Spine官方提供了一种基于Unity GUI生成Spine的方案，这种方案虽然满足Spine与UI交互功能，但是该Spine只能使用单一材质，如果使用了多材质会导致Spine错乱问题。为了使用Spine的多材质功能，就需要研究一种即可以让Spine正常使用多材质，又能够满足Spine与UI进行交互的方案。

1. 解决方案

利用Unity强大的引擎开发功能以及Spine的一些特性，我们结合图1，并通过以下2点解决上述问题。

1.通过Spine的骨骼坐标跟踪获得空间坐标。通过BoneFollower指定到需要追踪的骨骼节点，获取当前节点的空间坐标。

2.通过Unity的空间坐标转换方式生成需要的屏幕坐标。在拿到Spine的骨骼节点空间坐标后，将其通过空间坐标转换方式转成屏幕坐标，就可以将需要跟随骨骼运动的UI对象设置为转换后的屏幕坐标，用这种方法解决了交互的问题。

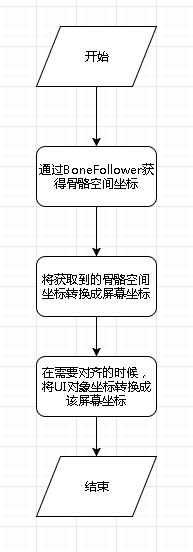


图1

1. 成果

实现了既可以在Unity中使用Spine的多材质功能，并且能让Spine与UI进行交互，如图2和图3所示，展现了UI元素Button跟随Spine骨骼坐标变换的过程。



图2



图3

**6.Unity中同时使用Sprite与Spine产生的批次过大问题解决方案**

1. 研发时限

2019年08月~2019年08月

1. 问题

【名词解释】： Spine：二维动画制作软件； Sprite：Unity提供的一种精灵，用于制作2D游戏；Sorting Layer:Unity通过层级对2D对象在世界坐标下的排序;Order in Layer:Sorting Layer层级内的子层级；

在制作2D游戏的时候Unity官方推荐使用自研的Sprite工具，它能大幅度提高游戏性能。但Sprite在动画制作没有Spine制作出的产品美观，因此制作的时候需要同时使用到Sprite和Spine工具，而在使用Sprite和Spine的情况下会产生批次过大就是要解决的核心问题。

1. 解决方案

要解决该问题不单需要工具支持，更侧重于在使用合批工具的规范流程，我们通过以下两方面解决上述问题。

1.首先上面介绍到Sprite在制作2D游戏的时候能提高大量游戏性能，实则是因为它使用了Unity的动态合批功能，需要使用该功能是通过调用了Sorting Layer功能。为了更方便在Sprite中使用Sorting Layer，Unity将Sorting Layer植入了Sprite组件当中，如图1所示。而Spine官方也将Sorting Layer组件植入自身组件方便调用动态合批功能，如图2所示。这里Sorting Layer表示创建出来的一个层级，Order in Layer相当于子层级。

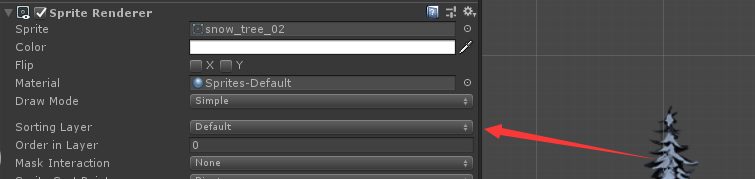


图1

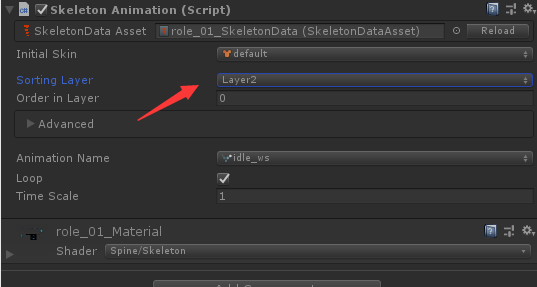


图2

2.既然有了工具可以调用动态合批功能，现在就需要知道如何使用这个工具达到动态合批效果了。我们通过一系列实验发现我们需要遵守 以下规则才能真正调用到动态合批功能：

* 同一个Sorting Layer上，用同一图集的Spine或Sprite只产生一个批次，能调用到动态合批。
* 使用不同图集的Spine或Sprite，会产生多个批次，不能调用动态合批。
* 三个Sprite或Spine对象中，其中两个对象在同一图集，另一个对象用自己一个图集，他们在同一个Sorting Layer，第三个对象的Order是穿插在两面两者中间的。虽然前面两个对象在同一个Sorting Layer中，但是中间穿插了不同图集的对象，因此合批被打破，会产生3个批次。
* 总结：需要合批的对象，必要使用同一图集，并且同一个Sorting Layer，并且当前层级下不能有其他图集的对象穿插在另外两个中间，它会打破当前层合批，同一图集允许使用不同Order产生前后遮挡关系。

1. 成果

按照上述规则进行制作后，可以实现在Unity同时使用Sprite和Spine的时候使用Unity的动态合批功能，可以将性能大幅度的提升。

**7.Unity使用Spine结合TexturePacker的Z轴排序错位问题解决方案**

1. 研发时限

2019年08月~2019年08月

1. 问题

【名词解释】： Spine：二维动画制作软件； TexturePacker：用于制作图集的软件；SpritePacker：Unity官方推荐的一种在编辑器内部进行快速图集制作的工具;Sorting Group：Unity提供给空间坐标下2D物体之间的一种排序工具；Sprite Render：Unity提供Sprite的渲染组件；SpiteSortPoint：Srite Render组件内轴心；Pivot：轴心；

在制作2D游戏的时候我们需要调用到Unity的动态合批功能，根据Unity官方要求，需要把零散的图片转换成一张图集才能调用动态合批进行优化。这个时候我们有两个选择，使用Unity自带的SpritePacker工具，内嵌的工具使用是很方便的，但由于Spine的制作需要在外部就制作图集并导入Spine中进行编辑，Unity的SpritePacker就满足不了这个导出编辑的需求了，我们需要用到TexturePacker的图集制作导出功能。同时我们就有了一个需要解决的核心问题，就是Spine在导入Unity后怎么进行正确的Z轴排序。

1. 解决方案

既然是排序，则涉及到多个物体。首先是Spine对象与Spine对象之间的排序问题，其次是Spine对象与Unity其他对象之间的排序，我们选择一个比较常用的Spine与Sprite之间的排序问题。要解决该问题，我们通过以下两方面解决上述问题。

1. 首先是Spine对象之间的排序，因为Spine使用TexturePacker是没有办法设定轴心的，如图1所示。如果不能在TexturePacker设置的话，那导入到Unity后就会以图集的中心点为轴心，这个就是导致Spine排序错误的关键。因此我们这里使用了Sorting Group组件，加在两个设置了相同层级的空对象上，再将Spine对象分别放到这两个空对象当中并调节了位置，如图2所示。结果产生了正确的排序效果，如图3所示。之所以会有这种效果，是因为实则其去排序的就是两个父对象之间，2个Spine继承了父对象的轴心位置，加上自身轴心上移才产生了这样的效果。

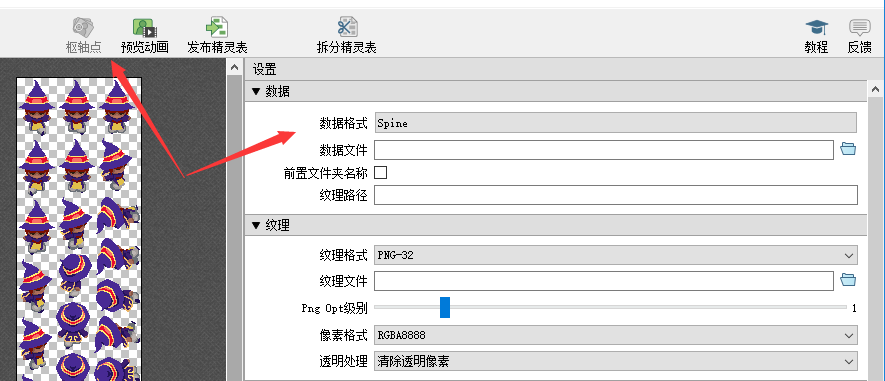


图1

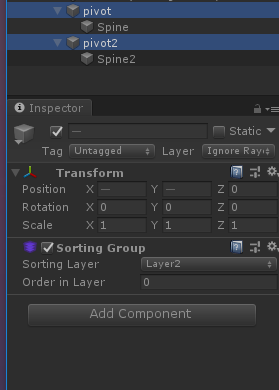


图2 图3

1. 随后则是Spine与Sprite之间的排序如何实现，首先我们把图片导入TexturePacker进行轴心调节，由于选择导出模式是Unity的Sprite是可以使用轴心功能的，如图4所示。随后我们将轴心点调至脚下，如图5所示。然后导入图集到Unity，制作成Spite后Sprite Render组件设置上面的SpiteSortPoint轴心点为Pivot，如图6所示。分别进行先后测试结果正确，如图7和图8所示。

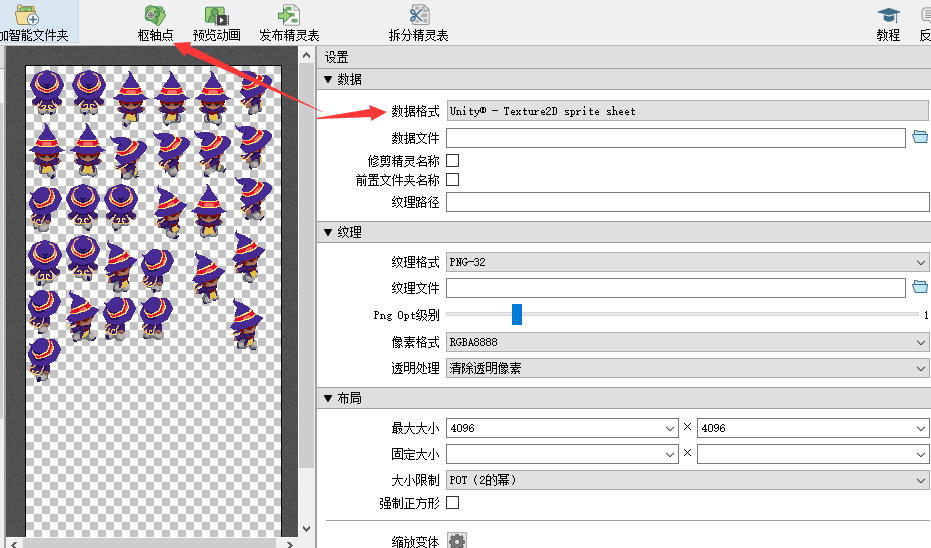


图4

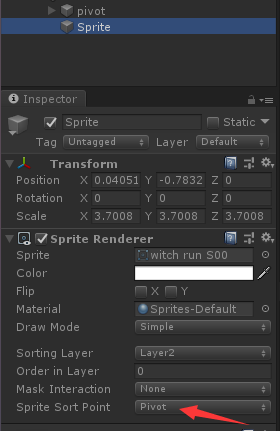


图5 图6



图7 图8

1. 成果

按照上述方法进行制作后，可以实现在Unity使用Sprite对象和Spine对象，以及使用Spine对象和Spine对象的时候能正确调用Unity的动态合批功能，可以将性能大幅度的提升，并达到我们想要的正确排序效果。