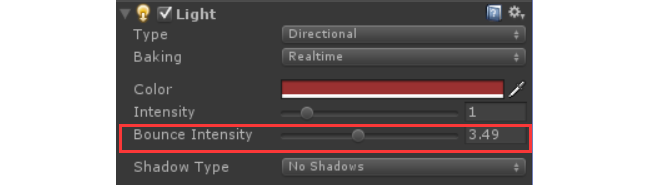
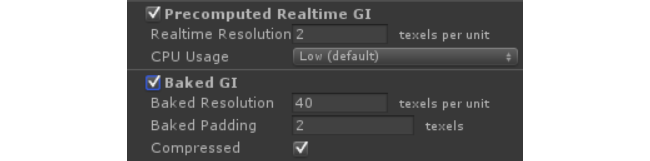
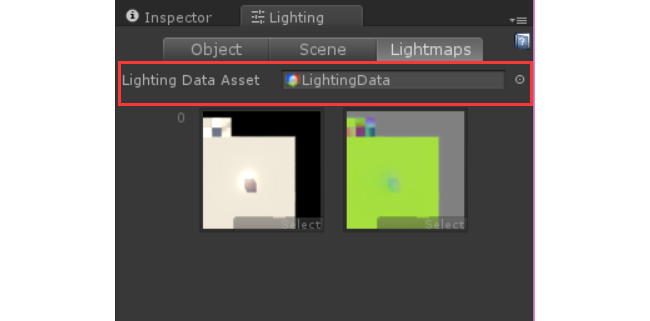
Unity 5正式版面世已经一段时间，许多开发者都已经在使用Unity 5进行开发。那么大家在使用Unity 5时都会碰到一些问题。今天我们Unity的官方技术工程师柳振东将在本文中针对Unity 5中全新的Enlighten全局光照系统的一些问题与大家进行一些分享。  
  
        为什么Unity 5的全局光照系统会让大家踩坑呢? 因为Unity 5取缔了Unity 4中使用的Beast全局光照系统，采用了全新的Enlighten全局光照系统。而这两套全局光照系统的算法并不一样，反映到使用上就是各种参数与操作细节的变化。  
  
        对于没有系统学习过计算机图形学的开发者，还是非常建议大家去了解一下全局光照明模型的，因为大家可能对局部光照明模型（就是我们平常所说的实时光照）比较熟悉，有时可能会把局部光照明模型的一些知识用在全局光照明系统上，白白花费时间做了一些无用功。  
  
        下面挑选一些常见问题与大家分享。  
          
**为什么场景烘焙出来的lightmap上有Realitime灯光的颜色？**  
  
        相信不少开发者都为这样的问题头疼过，因为灯光明明是Realtime的，怎么可能会被烘焙到lightmap里面去呢？但是这在Unity 5里并不是一个bug。  
  
        首先我们要知道Unity 5中新增一种全局光照明的使用方式：Precomputed Realtime GI（预计算实时全局光照明），简称：实时GI。而另一种Bake GI就是大家在Unity 4中一直用过来的ligtmap烘焙方式。  
  
        实时GI与Bake GI一样，也需要预先的Bake过程，但是与Bake GI不同，实时GI并不预计算场景中光线的反射信息，而是预计算场景中静态物体表面所有可能的反射光路，然后在游戏运行时结合灯光的位置、方向等信息实时计算出全局光照的结果。  
  
        这个计算机制使得预计算GI具备非常大的优势，在使用预计算GI的灯光，其位置、方向、强度、颜色等各种信息都可以在运行时实时变化。  
  
        举例：使用预计算GI我们就可以在场景中实现非常真实的日光变化效果（别忘了我们其实是在使用全局光照明模型啊，比实时光照效果强多了）。  
        但是，问题来了，预计算GI有一点非常容易让人混淆，那就是预计算GI需要灯光类型是Realtime。WTF？Realtime难道不是实时灯光吗？怎么变成全局光照明模型的灯光了？  
        Unity 5的灯光属性中增加了一项，叫做Bounce Intensity，如下图所示：



        Bounce Intensity是指全局光照中的间接光强度，要理解这点可以简单地把全局光照理解为直接光照与间接光照两部分。直接光照指直接从光源射到物体上的光，而间接光照指从其他物体表面反射而来的光。那么显而易见，直接光照的部分其实就是局部光照模型，也就是实时光照计算出来的结果。  
  
        回到正题，Bounce Intensity在Unity 5 中默认值是1，意思就是说一盏Realtime灯光默认是使用预计算GI的，并且间接光照的强度不做改变（大于1是强制增大间接光照，这在一些由室外光照射入洞穴这类的场合中可能会用到）。而lighting窗口中的预计算GI选项也是默认开启的，如下图所示：



        所以如果你并不更改这些设置，即使看起来你好像只使用了Realtime的灯光，但是真正运行的时候却其实是在使用全局光照系统！  
  
        接下来要说明的一点是，只要一盏灯的Bounce Intensity大于0，Unity就会认为你需要使用全局光照系统。那么即使你在lighting窗口中取消勾选Precomputed Realtime GI，Unity依然会尝试使用全局光照。  
  
        既然已经取消了预计算GI的选项，Unity会去使用另外一种全局光照的方式，Bake GI（只要你还勾选着Bake GI）。而结果就是，烘焙出来的lightmap里也有那盏Bounce Intensity大于0的Realtime灯光的信息。  
  
        总结：Unity 5中灯光有个新属性Bounce Intensity，这个值只要大于0系统就会认为你需要使用预计算GI计算这个灯光，而如果此时Precomputed Realtime GI没有被勾选，而Bake GI勾选了那么Unity会把这个灯光也烘焙到lightmap中去。  
  
**为什么在Unity 5中动态更换lightmap没有作用？**  
  
        在场景中动态更换lightmap是挺常用的需求。例如：同一个场景需要白天与黑夜两个时间的效果，那么我们就会烘焙两张不同的lightmap，然后用脚本在运行时切换。在Unity 4中可以通过把lightmap资源load到Texture2D中，然后赋值给一个LightmapData结构，最后赋给LightmapSettings.lightmaps来达到更换当前使用lightmap的目的。  
  
        但是在Unity 5中会发现使用这个方法并不凑效。是因为lightmap没有正确更换吗？不是的，其实lightmap已经更换了，问题是在于此时使用lightmap的物体并不知道自己应该使用哪张lightmap，也不知道要从lightmap的哪个地方开始采样，而这其实就是每个Renderer上的两个参数，lightmapIndex与lightmapScaleOffset。  
  
        现在大家肯定会问为什么在Unity 4中没有问题呢？那是因为这些信息在Unity 4中已经被序列化进场景文件中，当场景加载进来的时候这两个值就被赋回到每个Renderer中。而在Unity 5中，lightmapIndex与lightmapScaleOffset因为多场景编辑的逻辑需要，不再被序列化到场景文件中去了，而是存在于一个伴随lightmap烘焙产生的一个新文件，Lighting Data Asset中去了（这个文件在初期的5.x中叫做lightmapSnapShot）。这个文件与lightmap在同层目录中，并且可以在Editor中随时更换当前使用的LightingData文件，如下图所示：

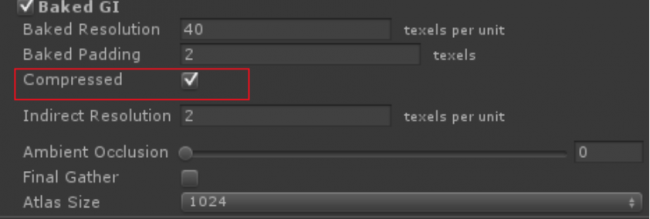


        那么回归正题，既然LightingData文件存储着lightmapIndex与lightmapScaleOffset，那么只要保证LightingData文件能被最终的可执行文件使用不就没有问题了吗？其实这样是没有问题的，如果我们只需要更换灯光条件不同情况下的lightmap，而场景本身并没有更改的话，那么多套lightmap对于场景中的静态物体Renderer而言也只需要同一套lightmapIndex与lightmapScaleOffset信息而已。所以在这种情况下要达到运行时更换lightmap的效果，只要保证lighting窗口的Ligtmaps页面中Light Data Asset里选中了正确的LightingData文件即可（因为很多开发者习惯性删除掉这个文件，因此也就丢失了lightmapIndex与lightmapScaleOffset信息）。  
  
        简单的情况我们简单地使用LightingData文件即可。但是还有一种情况，如果我们需要场景本身也有所改变要怎么办呢？  
  
        举例：我可能需要原始的场景lightmap与一个被炸弹破坏过后的场景lightmap，由于场景中的静态物体有所改变，这个时候两次烘焙会产生两个数据不同LightingData文件，无法简单使用其中一个。然而比较可惜的是现在Unity并不能在运行时切换LightingData文件，其设计之初就仅仅作为Editor使用的资源。  
        那么对于这种情况，我们需要自己记录下每个Renderer上的lightmapIndex与lightmapScaleOffset信息，然后在更换lightmap的时候把这个信息还原回去。  
  
        具体做法有很多种，基本原理都是把lightmapIndex与lightmapScaleOffset序列化起来。比如说我们可以给每个静态物体挂一个脚本，在里面定义一个包含Render,lightmapIndex与lightmapScaleOffset的结构，然后一个Save函数用来在每次烘焙lightmap之后把lightmapIndex与lightmapScaleOffset序列化进结构中，一个Load函数用来在切换lightmap时还原每个Renderer的lightmapIndex与lightmapScaleOffset信息。现在这个问题就解决了。

**为什么烘焙的point灯光会有一圈圈的色阶变化？**  
  
   大家在烘焙lightmap的时候可能会发现类似下图的奇怪效果。



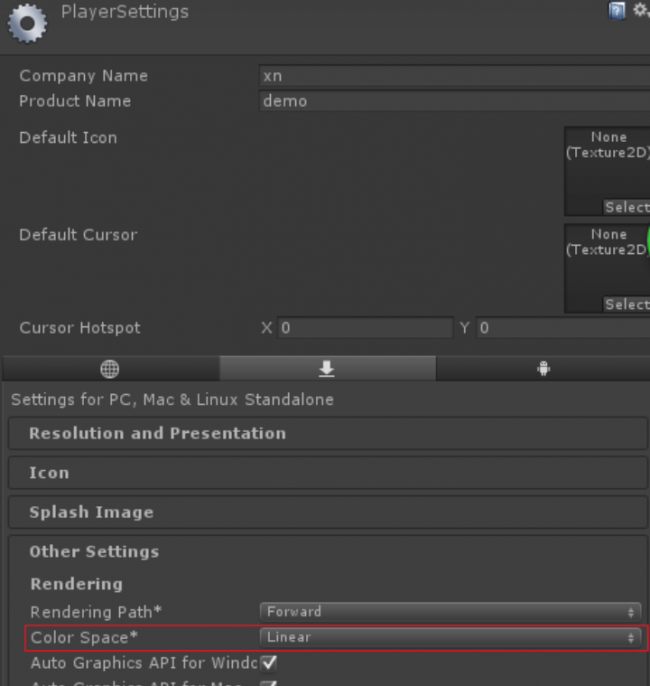
   图中我们使用了一盏Baked的point light，烘焙之后发现光照效果会出现一圈圈的色阶突变。  
  
   原因是我们使用的是压缩版本的lightmap。如下图Lighting窗口中选项所示：



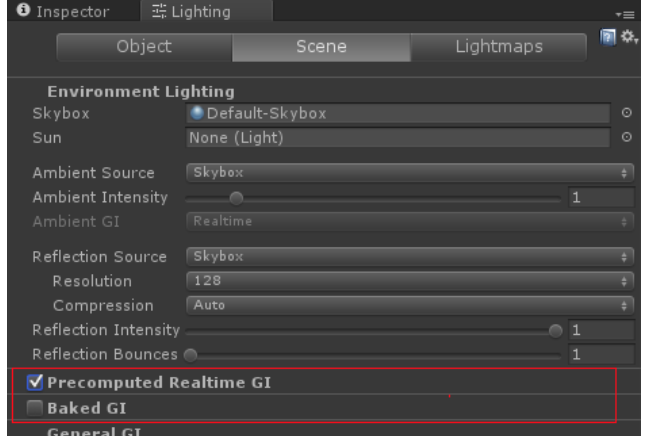
   这个压缩虽然可以帮助我们减少游戏最后的打包体积，但是对场景的光照效果也会产生影响，所以对于要追求最好的光照效果的场景建议不要勾选此选项。  
  
   最后我们来看看去掉此选项后的效果，现在，色阶变化不存在了。



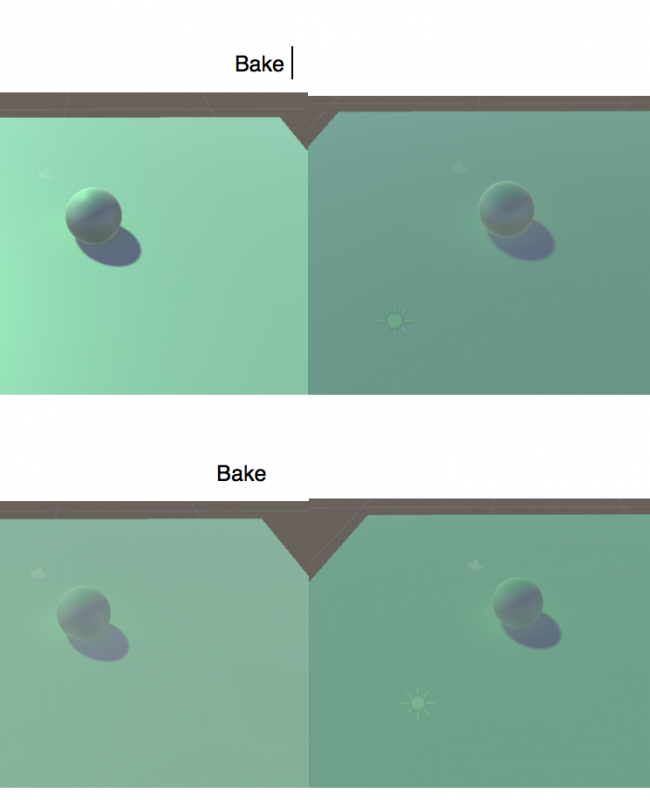
**为什么在Realtime灯光下的效果和改为Baked烘焙的效果差别很大？**  
  
    这个的问题美术同学会经常遇见。美术同学一般在烘焙场景之前会先把灯光都设为Realtime，来实时查看调整灯光参数后的效果。调整到较满意的效果之后，就把灯光类型改为Baked然后进行烘焙。  
     
   在Unity 4中这样做一般可以达到烘焙之后的效果与实时灯光的效果比较接近，但是在Unity 5中可能会发现差别很大。究其原因主要有二点。  
  
   1、在上面我们已经提到，新的灯光参数Bounce Intensity，这个参数的调整会影响最终烘焙效果中间接光分量的比例。  
   2、Unity 5中对Gamma和Linear颜色空间的实现做了调整以更符合真实情况，但这也导致实时灯光效果与烘焙效果在两种颜色空间中存在差异。  
  
   所以，建议美工同学在Linear颜色空间中进行Realtime灯光的调整。因为在Linear颜色空间下，实时灯光与烘焙效果最为接近。不过由于目前移动平台并不支持颜色空间切换，我们可以让项目在PC平台下切换为Linear颜色空间。该选项在Player Settings中的Other Settings里，如下图所示：



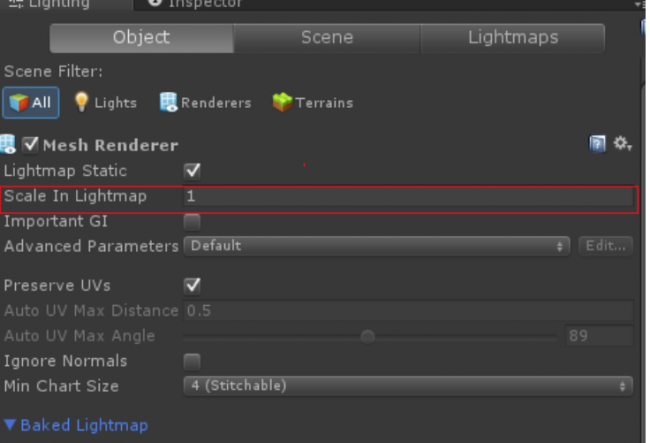
   另外，为了让Bounce Intensity参数的调整也能反映在Realtime灯光中，建议美术同学在实时调整灯光参数的时候在Lighting窗口中勾选Precomputed Realtime GI，如下图所示：



当然Precomputed Realtime GI也是需要在场景静态物体的位置固定之后烘焙一次的，所以如果怕会忘记烘焙的话也可以勾选Build旁边的Auto选框。  
  
    最后，在调整好灯光效果之后我们把灯光类型改为Baked（或Mixed），取消勾选Precomputed Realtime GI，并勾选Baked GI，然后进行烘焙。  
下面展示了两种颜色空间下Realtime灯光与烘焙效果的对比：



    我们可以看到Linear颜色空间下两种光照方式的效果是最为接近的。  
  
    另外还有朋友曾经咨询：项目的场景在手机上的烘焙效果与Editor中看到的实时灯光的差异非常的大，已经远远比上面对比图中的差别要大了。  
  
     如果大家也有这样的问题，那么请注意：不同的屏幕对于同一种颜色的表现都会有或多或少的差别，要对比手机与电脑上看到的颜色差别不能直接进行对比，而应该让两个效果在同一个屏幕中对比。一个简单的办法是在手机中截屏然后传到电脑中进行对比。感兴趣的同学可以尝试，保证你一定会为手机与电脑屏幕的颜色解析差异感到惊讶。  
  
**怎么让不同物体占用不同数量的lightmap像素？**  
  
   这个问题经常会有开发者遇见。主要目的是为了最大限度缩减lightmap的大小，通过让场景中次要物体的lightmap效果粗糙一点来节省游戏的空间。  
  
   其实做法非常简单，我们可以通过lighting窗口中Object页面里的Scale In Lightmap来设定某个选定的Object在lightmap中占用像素的多少，默认是1对于重要的物体通过提高这个数值来增加此物体使用的lightmap像素，反之我们可以把这个值设为0到1之间来减少此物体所使用的lightmap像素。选项如下图所示：



为大家解答的Unity 5 全局光照系统Enlighten问题就到这里了，如果大家还有其他问题希望和我们的技术工程师直接进行交流，欢迎大家访问中文官方论坛进行提问： [http://forum.china.unity3d.com](http://forum.china.unity3d.com/)