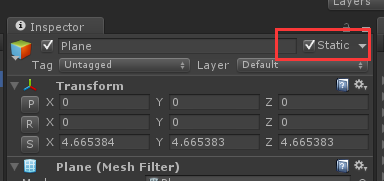
**Unity5场景烘焙方法**

1.场景中的物体 都要勾选 Static



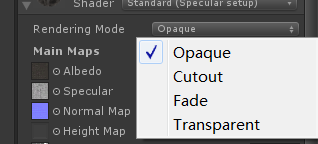
2.场景物体 Shader的使用



Standard以及Standard(Specular setup)两者的区别是, Standard使用了金属模拟的外观, 而Standard(Specular Setup)使用了镜面模拟的外观.

使用中, 对于任何不是金属的材质, 都应该选择Standard(Specular Setup)..

**我们几乎没有金属材质 所以使用Standard(Specular Setup).**



选择Shader的渲染形式, 四种可选:

`Opaque---不透明

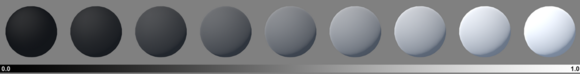
`Cutout---透明但没有半透明

`Fade---整体透明

`Transparent---只有像素透明(意思就是相比起Fade来说, 材质上的高光等信息不会变得透明化).

**对于房子（使用Opaque） 对于植物 （使用Cutout）**

Albedo---就是Diffuse漫射的意思, 但是这里的Alpha通道还会控制透明程度.

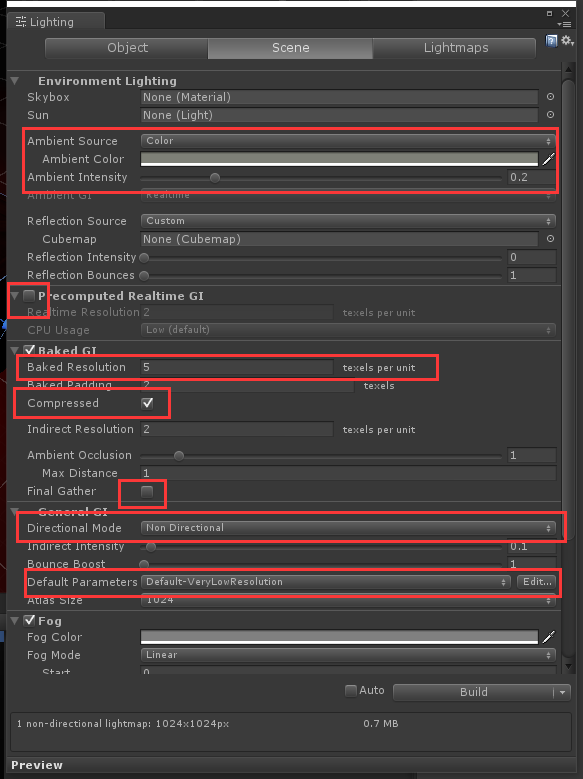


Secondary Maps---这里可以赋予细节贴图.

UV Set---设置UV通道.

技巧: 你赋予上去的贴图, 按住ctrl点击缩略图可以打开贴图预览.

值得一提的是, 两个Standar材质是经过高度优化的, 你没有赋予贴图的部分在工程打包的时候是不会造成任何性能消耗的, 比如你没有为"Heightmap"赋予任何贴图, 那么你就无需担心这个Heightmap在Shader中有什么冗余



Baked GI就是烘焙光影贴图的内容了. 可以和Precomputed Realtime GI同时存在.

Baked Resolution---Bounce效果的分辨率, 由于是Baked的, 所以不用像Realtime那样设置得很小.

Baked Padding---防止光照贴图的像素在物体之间溢出. 是物体之间而不是UV簇之间, Unity没有智能到可以自动帮我将一个物体的UV拆开来然后再合并

Compressed---是否压缩光照贴图, 提高性能, 缩小容量, 但画质会降低.

AO环境光

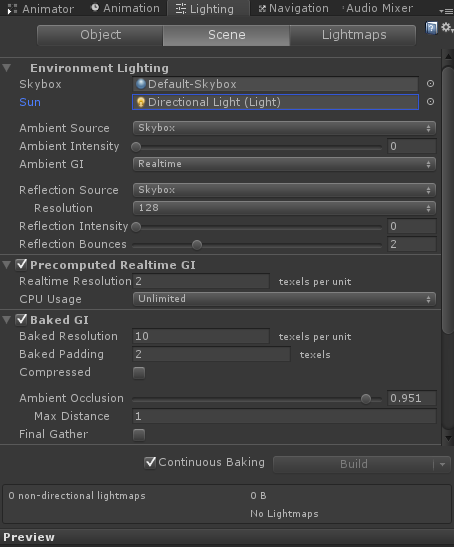
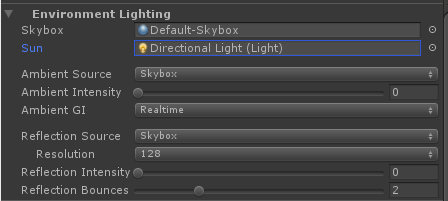
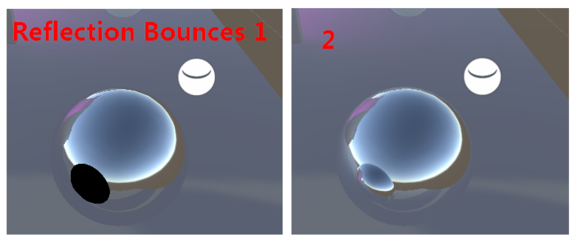
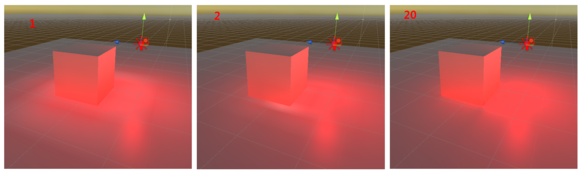
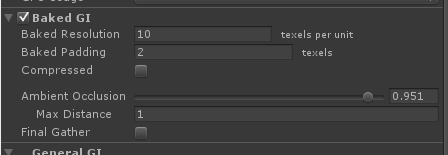
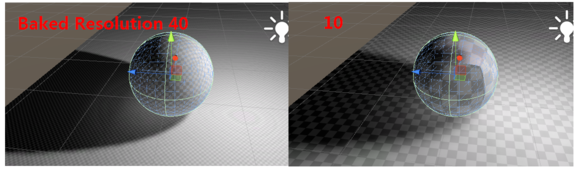
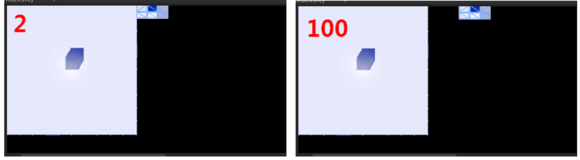
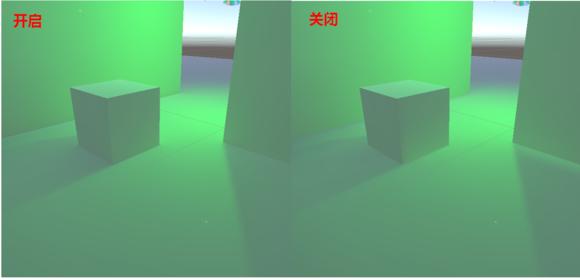
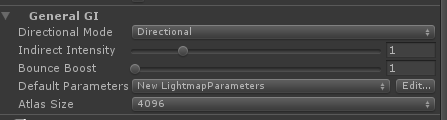
Final Gather（最终聚焦）---开启后效果的确有很大的提升, 但是会让烘焙变慢, 成品必开.

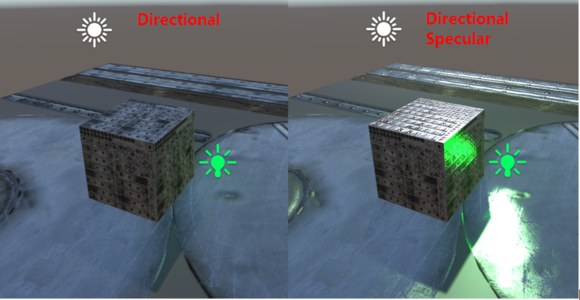
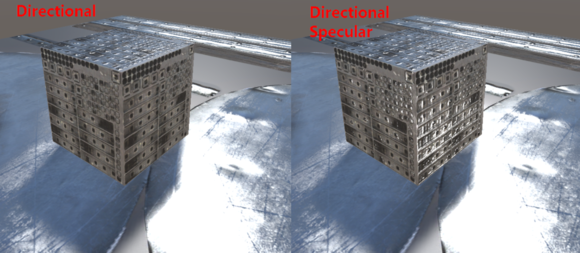
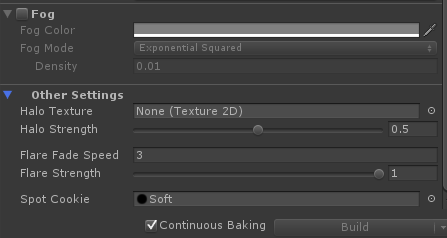
Ray Count---与Final Gather有关. 该值越大效果越好. 烘焙也越慢. 经过仔细对比, 默认的1024和4096几乎没有区别, 但4096烘焙得出奇地慢. 事实上1024和128也几乎没有区别.

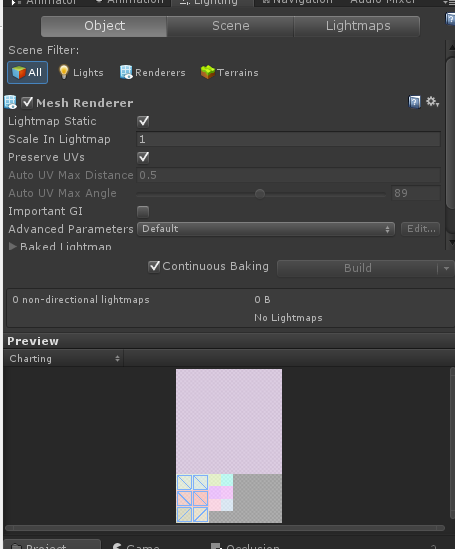
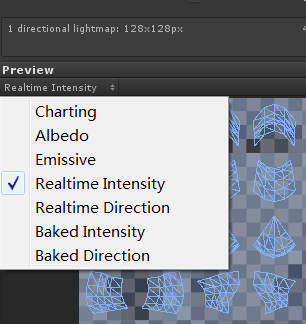
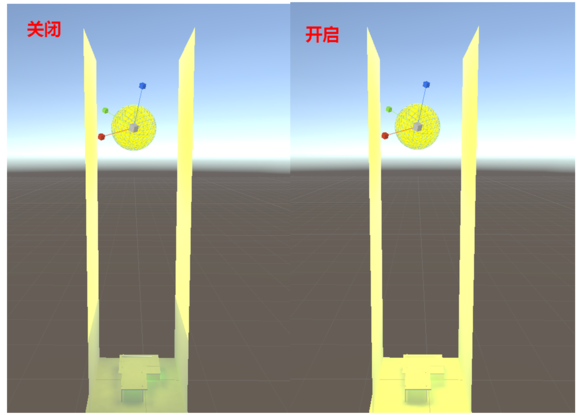
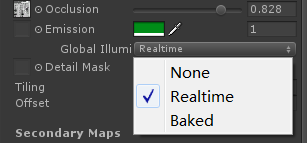
**注意事项：烘焙之前 勾选 Auto 使用预览查看效果**！

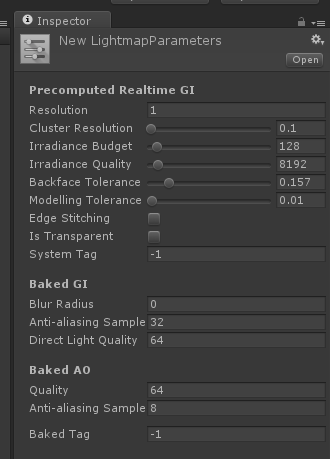
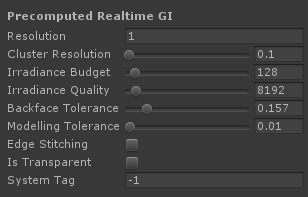
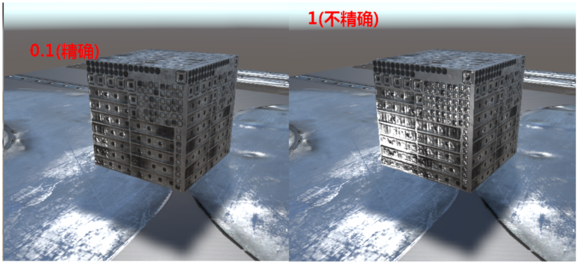
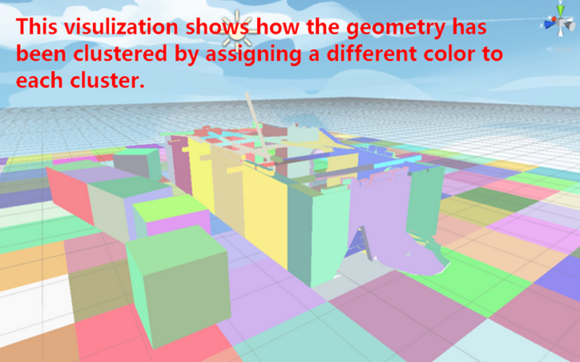
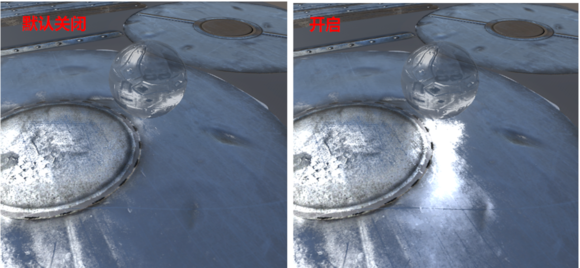
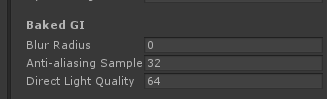
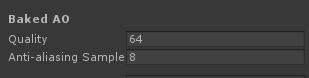
**开始烘焙的时候 把压缩和最终聚焦勾选**

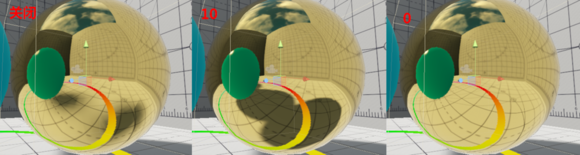
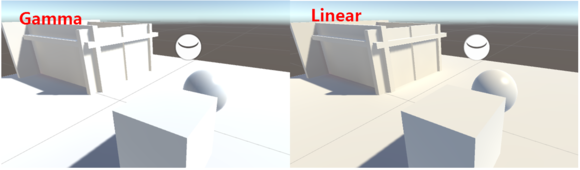
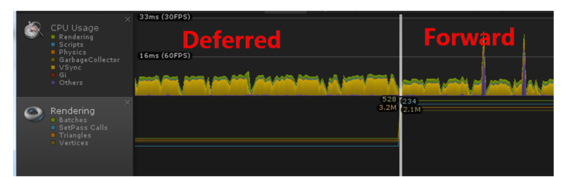
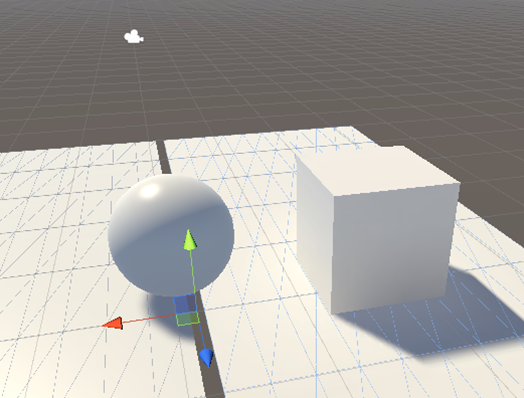
==================以下内容来自互联网 大家有兴趣就自己看看====================

GI是什么? GI =直接光照+间接光照+环境光+反射光.  
直接光照先不用说了,  
间接光照是光线在物体上反射所带来的光照. 核心参数: 每个光源上的Bounce Intensity.  
环境光可以直接理解为你天空盒的颜色, 蓝天白云, 那么环境光就是蓝色白色. 核心参数: Lighting视图中的Ambient Intensity.  
反射光可以理解为镜面反射一样的存在, 核心参数: Lighting视图中的Reflection Intensity  
请各位逐一尝试这四个参数并且尝试能将场景调成纯黑然后再调回来, 就能理解其中的概念了.  
  
  
GI分为两种, 一种是Precomputed Reatime GI, 这种GI需要预先计算, 计算场景中所有的Static物体的信息, 并且允许在运行时任意修改光源的Bounce Intensity或者移动光源的位置. 所有的变化都是实时的.  
第二种是Baked GI, 这种GI不会预先计算但会进行预先烘焙, 无法像Precomputed Realtime GI那样在运行时更改光源.  
  
  
GI的重点在于Bounce , 可以将Static物体上的光反弹到其他Static物体上, 无法将非Static物体上的光反弹到Static物体上. 也无法将Static物体上的光反弹到非Static物体上, 但是Unity5引入了新的概念, 那就是Realtime的LightProbe, Unity4中的LightProbe只能静态烘焙, 而Unity5的LightProbe会实时捕捉包括Precomputed Realtime GI的Bounce光在内的任何光线信息并且将其赋予非Static物体上.  
  
  
要想完全理解GI, 首先要好好说一下Lighting面板.  
  
  
先说重点的Scene页面.  
  
  
Environment Lighting这些内容控制的是场景总体的光线信息, 包括天空盒的设置, 可以赋予一个天空盒材质上去, 而天空盒材质在Unity5里面也有更新, 很有意思, 大家可以试试, 新建一个材质然后选择Skybox/Procedural就行了.  
Sun---如果你的天空盒材质是Skybox/Procedural那么这里允许指定一个Directional Light, 你试着旋转一下这个Directional Light就知道他是干嘛的了.  
Ambient Source---环境光来源, 一般都是天空盒. 如果你的天空盒会在游戏运行时实时变化, 一定要在Ambient GI这里选择Realtime, 否则可以选择Baked.  
Reflection Source---镜面反光的来源, 说到镜面放光, 需要说到Reflection Probe, 可以理解为Reflection Probe会规定一个区域, 任何进入这个区域的物体的镜面反光的信息都会被覆盖, 而没有进入任何Reflection Probe的物体的镜面放光的信息就是这个Reflection Source所指定的内容.  
Reflection Bounces---当你使用Reflection Probe的时候, 最多允许镜面来回反弹几次, 举个例子, 两面镜子对着放, 会有无穷的发射次数, 而这显然是有性能消耗的, 这里指定这个来回反弹的最大的次数.  
  
  
  
  
http://imgsrc.baidu.com/forum/w%3D580/sign=d7cefc04f303918fd7d13dc2613f264b/eaa532fa828ba61e1c3a63cb4534970a324e598a.jpg  
  
Precomputed Realtime GI的概念是, 预先计算场景中的所有静态物体的信息, 具体计算出来了哪些信息咱们开发者不用操心, 这些计算出来的信息会用于实时GI.  
Realtime Resolution控制的是实时GI的分辨率, 建议不要调太高.  
  
  
CPU Usage这个越大, 在调节Realtime光源的时候, GI效果就会响应得越快.   
  
  
  
  
Baked GI就是烘焙光影贴图的内容了. 可以和Precomputed Realtime GI同时存在.  
Baked Resolution---Bounce效果的分辨率, 由于是Baked的, 所以不用像Realtime那样设置得很小.  
  
  
Baked Padding---防止光照贴图的像素在物体之间溢出. 是物体之间而不是UV簇之间, Unity没有智能到可以自动帮我将一个物体的UV拆开来然后再合并.   
  
  
Compressed---是否压缩光照贴图, 提高性能, 缩小容量, 但画质会降低.  
AO不用说了吧  
Final Gather---开启后效果的确有很大的提升, 但是会让烘焙变慢, 成品必开.  
  
  
Ray Count---与Final Gather有关. 该值越大效果越好. 烘焙也越慢. 经过仔细对比, 默认的1024和4096几乎没有区别, 但4096烘焙得出奇地慢. 事实上1024和128也几乎没有区别. 反正在我的小测试场景中是几乎没有任何区别.  
  
  
  
  
General GI组的这些参数同时适用于实时GI以及烘焙GI.  
Directional Mode这个真是太有意思了!   
靠, 度娘一楼只能插十张图, 楼下继续.

Directional Mode这个真是太有意思了!   
  
  
图中是静态物体+Bake光源. 可以看到如果选择了Directional, 会像Unity4一样对静态物体处理得很不好, 但如果选择了Directional Specular, 画质马上就提升了一个等级.   
不仅对于静态物体+Baked光源有提升, 对于静态物体+Realtime光源也有提升. 但是对于动态物体是没有提升的.  
  
  
这两幅图是静态物体+动态光源. 可以说Directional Specular直接提升了静态物体对于直接光和间接光的效果.  
但是这里有一个不得不重视的警告, 说Directional Specular对于SM2.0以及GLES2.0无法支持. 如果选择了Directional Specular, 就说明我不得不在我的PC游戏的最低配置上提升需求(当然了要是在八年前是要这么说的)! 这里说一下GLES(OpenGL ES)和SM(Shader Model). OpenGL ES针对手机平台, SM针对主机和PC平台. 从iPhone5s开始支持了OpenGL ES 3.0版本. 而对于SM版本来说, 目前还停留在SM2.0的显卡基本已经不存在了.   
  
  
Indirect Intensity---同时调节Bounce, 静态物体的自发光对其他物体的影响, 以及ambient lighting的强度.  
  
  
Bounce Boost顾名思义  
  
  
Default Parameters---设置一个LightmapParameters资源上去. 这么做是为了复用性.  
Atlas Size---Atlas尺寸.  
  
  
Fog 组合Other Settings组的内容都是Unity4就有了的. 新功能教学中不赘述

  
  
下面来看一下Lighting窗口的Object页面  
Preserve UVs以及Auto UV Max Distance以及Auto UV Max Angle--- 说到这三个UV选项, 就不得不说一个新的知识点.  
每个物体在Unity中最多会使用3个UV通道, UV1是贴图通道, UV2是光照贴图的通道, 而第三个UV是Unity自动生成的要用于Precomputed Realtime GI的UV通道.   
在这里会看到区别, 其中最后两个, 使用了光照贴图的UV通道, 而前面五个使用了第三个UV,  
而这三个参数正是用于控制如何生成这第三套UV的. 这三个选项自己测试一下就有结论了, 不再赘述. 说到这里, 如果你坚持看到这里还没有打开Unity手动进行测试并且记笔记的话, 我推荐你不要这样. 我不是专门写教程的我无法写得很好如果你不能一边测试一边看教程你说不定什么都学不到.  
  
  
Important GI---在比较大的场景里面, Emission物体够大够亮. 但有些物体没有却没有被他照射到, 那么可以为他开启这个Important GI.  
  
  
好了, 这里要说一下Emmision了, 在第一章里, 我们说到了两种自带PBS的Shader都有Emmision可以用, , 重点: 要想Emmision影响其他物体, 那么Emmision物体必须是Static的. Realtime和Baked的区别在于, Realtime允许在游戏运行时更改Emmision颜色以及强度来影响整体GI, 当然了更改了Emmision颜色之后需要调用一个API, 这个API就是DynamicGI.UpdateMaterials, Unity官方文档说, 之所以没有做成自动刷新显示是因为这方面的刷新比较消耗性能, 而Unity正在想办法将这个操作的性能消耗减少并且将这个做成实时刷新的.  
  
  
Advanced Parameters---允许单独的物体的LightParameter的设置覆盖Default Parameters. 关于LightParameter一会再说哈.

下面说LightmapParameters,为了复用性, 将LightmapParameters做成了资源.  
  
  
LightmapParameters资源可以赋予全局, 或者赋予单个物体来覆盖全局中的设置.  
  
  
  
  
Precomputed Realtime GI这组的都是与Precomputed Realtime GI相关的.  
Resolution---Realtime Resolution会乘以这个值来得出最终要使用于实时GI的Resolution.  
Cluster Resolution---Too large clusters will result in light/shadow bleeding and too small clusters slow down the precompute. 数值越小效果越精确, 但是烘焙起来也越慢.  
  
  
什么是Cluster? 看图  
  
  
  
Irradiance Budget---数值越小效果越平滑. 但是更加消耗内存和少量的CPU.  
  
  
  
  
Irradiance Quality---数值越大效果越好, 但是预计算越慢. 对游戏运行时的性能消耗无影响.  
  
  
  
  
Backface Tolerance---数值越小, GI就越容易无视物体的背面. 同时, 两个完全贴合的物体的贴合部分也越容易被GI无视.  
  
  
  
  
Modelling Tolerance---数值越小效果越好但是不能是0.  
  
  
  
  
Edge Stitching---不能说完全消除但是可以缓解UV接缝的视觉纰漏.  
  
  
Is Transparent---还记得刚才说过什么吗? LightmapParameters资源可以赋予全局, 或者赋予单个物体来覆盖全局中的设置. 没错, 为透明的物体赋予可以覆盖全局设置的LightParameters, 并且将这个LightParameters中的这个Is Transparent开启是更好的选择.  
  
  
System Tag---和Is Transparent一样, 使用单独物体的LightParameter来确定这个物体被考虑进哪个System. 说到System, 可以理解为Precomputed Realtime GI的Atlas.   
  
  
  
  
Baked GI这里的选项都是针对Baked GI的.  
  
  
Blur Radius---光照贴图的模糊处理. 可用于将烘焙出来的阴影进行柔化处理.  
Anti-aliasing Samples---数值越高, 烘焙出来的阴影的锯齿越少.  
Direct Light Quality---数值越高烘焙效果越精确. 烘焙时间也会更长.  
  
  
  
  
Baked AO这组是与AO有关的参数.  
Quality---数值越高AO的质量越高.  
Anti-aliasing Samples---AO抗锯齿.  
  
  
Baked Tag---这个参数虽然和Baked AO组离的很近, 但是注意观察其实是有一定Space的, 因此他不属于AO组. 控制光照贴图的Atlas编号

  
  
下面说一下Reflection Probe, 这东西很扯, 工作原理在这里摆着呢, 显然Reflection无论怎么调节Bound的或者采样点的位置, 都无法适应一个”房间”或者一个”地板”, 他只适用于一个玻璃球, 一个杯子, 或者是其他的单个的小型物体.  
  
  
当一个物体横跨了多个Reflection Probe的时候, 他的Mesh Renderer中会自动加入所有他触及到的Reflection Probe. Weight是通过计算每个他触及的Reflection Probe和这个物体之间距离来得出的. 另外还可以单独调节Reflection Probe的”Importance”这个参数来手动干预这个计算结果.  
  
  
Mesh Renderer的Reflection Probes选项能够选择四种使用ReflectionProbe的方法,   
Off---代表不使用Reflection Probe.  
Simple---代表只使用列表中Weight值最大的那个ReflectionProbe.  
Blend Probes---should be used for interior objects, that way you’ll ensure that object won’t accidentally use reflections from skybox (a.k.a default reflection).  
Blend Probes And Skybox---should be used for exterior objects, that way when object leaves the bounding box of reflection probe, it will gradually switch reflections from reflection probe to skybox.  
  
  
这里只讲无法第一时间测试出意义的参数. 其他参数都比较简单.  
Type选择类型, 三种可选.  
Dynamic Objects---在Custom模式中是否将非静态物体也给Bake出来.   
  
  
Time Slicing---官方文档上说这个选项仅仅针对于当我选择了Every frame作为刷新模式的情况, 实则不然, 因为不管是On Awake还是Every frame还是Via scripting, 这个Time Slicing都是有效果的, 要了解这个选项. 首先要了解一下Reflection Probe的内部生成机制, 一个Realtime的Reflection的生成有三个步骤, 第一个步骤要计算一个Cubemap的六个面, 第二步要考虑这个Cubemap的mipmap(文档上说为了支持Glossy Reflections, mipmap是必要的, 而Glossy Reflections对于调节不同程度的Smoothness时是有一定效果上的提升的), 第三步是将计算结果拷贝到Cubemap中. 那么TimeSlicing的三个选项分别有不同的三步计算所需帧数.  
All faces at once---9帧, 性能消耗中等. (9帧之内再次通过Via Scripting调用刷新是无效的)  
Individual faces---14帧(在9帧的基础上加了将六个面分为六帧的那多出来的五帧), 性能消耗最低.(14帧之内再次通过Via Scripting调用刷新是无效的)  
No time slicing---1帧. 所有动作都在同一帧执行, 这种性能消耗最大. (1帧之内再次通过Via Scripting调用刷新是无效的)  
  
  
Importance---影响了一个MeshRenderer中的多个ReflectionProbe的Weight的自动混合比例. 当一个物体处在多个Reflection Probe的”领地”内的时候, 首先会考虑每个Reflection Probe的Importance, 之后, 在此基础上才会考虑每个Reflection Probe与该物体之间的分别的交叉的体积的大小. 也就是说Importance的优先级高于交叉体积的计算.  
关于交叉体积的计算: For ex., if the first volume is 1.0 and the second is 2.0, then first probe’s influence will be 1.0 / (1.0 + 2.0) = 0.33, the second probe’s influence will be 2.0 / (1.0 + 2.0) = 0.67. 还有一种特殊情况, 就是就是一个小的Probe套在一个大的Probe里面, 而一个物体又完全在这个小的Probe里面, 那么对这个物体来说, 小的Probe的Weight会是1, 大的会是0. 而当这个物体逐渐脱离这个小的Probe的时候, 那么小的Probe和大的Probe对于这个物体的Weight会进行一个混合.  
  
  
Box Projection---如果选中这个选项, 那么Probe Size以及Probe Origin会影响Reflection Probe的映射效果.   
Size---该Reflection Probe的影响范围. 进入这个范围的MeshRenderer会自动和这个Reflection Probe产生关联. 同时, 当Box Projection开启的时候, 这个Size也会影响Reflection Probe的UV映射.  
Probe Origin---取样点相对于影响范围中心的偏移. 当勾选了Box Projection的时候, Probe Origin的位置会影响Reflection Probe的UV映射.  
  
  
Shadow Distance---阴影距离, 特性和Quality Setting中的特性一样, 数值越小, 阴影显示越近, 但阴影越精细, 调成0会完全关闭阴影.  
  
  
  
下面说一下GI方面的其他注意事项以及Rendering Path  
  
  
首先, 在Linear Color Space中PBR才会有最正确的效果  
  
  
  
  
然后是,  
光源的Baking模式有三种可选.  
`Realtime---对动态物体还是静态物体都使用实时光源.  
`Baked---只对静态物体使用烘焙光照. 对动态物体没有作用.  
`Mixed---对静态物体使用烘焙光照, 对动态物体使用实时光源.  
首先要明确的一点, 那就是提到Bounce ,就马上应该联想到静态物体, 因为Bounce对动态物体是无效的. 那么关于Bounce和光源的Baking模式之间的关系, 只有Realtime模式支持实时Bounce, 也就是更改Intensity或Bounce Intensity会对结果产生影响. 与此相对的, Baked以及Mixed都仅仅支持烘焙的Bounce.  
  
  
下面说一下Rendering Path, 请各位集中看一下下面的话, 或者反复读两遍. 因为个人觉得这个挺重要.  
  
  
Unity5多了一种RenderingPath, 可以取代过去的Legacy Deferred. 最大的特点是对于实时光影来说的性能消耗更低了.  
  
  
Deferred  
这个模式是最适合动态光影的. 对于次时代PC或者主机游戏, 当然要选择这个. 通过Viking Village以及”进化”这些最新游戏的观察我发现次时代游戏几乎不需要烘焙光照贴图了, 全都使用实时阴影是很好的选择. 通过阴影距离来控制性能消耗. 而在Viking Village的场景中,由于整个场景全部使用了动态光源, Forward的Rendering方面的性能消耗要比Deferred高出一倍! 因此在完全使用动态光源的项目中千万不能使用Forward.  
  
  
  
  
Forward  
对于实时光影来说, Forward的消耗比Deferred更高, 但是Forward更加适合用于半烘焙半实时的项目.  
Forward解决了一个Deferred没能解决的问题. 那就是Deferred不能让Mixed模式的DirectionalLight将动态阴影投射在一个经过烘焙了的静态物体上.  
  
比如这里这样, 立方体和右边的Plane是静态物体, 球体和左边的Plane是动态物体. 唯一的DirectionalLight是Mixed的. DirectionalLight烘焙了一下光照贴图, 但是球体不是静态物体, Mixed的DirectionalLight现在虽然可以照亮这个球体但是却无法将他的阴影投射在右边的经过烘焙了的静态的Plane上面. 但却可以将球体的阴影投射在左边的动态的Plane上面.   
而Forward却完美解决了这个问题.  
  
  
因此, Deferred适用于全动态阴影的主机以及PC游戏. Forward适用于烘焙阴影与实时阴影相结合的手机游戏.  
  
  
只要记住上面五句话就行了  
GI章节结束. 复习一下.  
`大部分GI功能仅能用于静态物体以及LightProbe上  
`动态物体应该受LightProbe所影响.  
`GI分为两种, 一种动态一种烘焙, 手游当然应该烘焙, 主机或PC可以选择用实时.  
`GI要和Standard 以及Standard(Specular Setup)配合使用才有最佳效果.  
`Deferred适用于全动态阴影的主机以及PC游戏. Forward适用于烘焙阴影与实时阴影相结合的手机游戏  
关于Box Projection, 官方文档说,If you have a room which you want to reflect, ensure that reflection bounding box’s is of similar dimensions, this way you’ll get the best results. 的确是这样, Reflection Probe的Bound刚好覆盖要反射的范围是最佳的效果.  
