烘焙光照

预计算的光照信息烘焙到哪里？

静态物体表面颜色信息烘焙到光照贴图，动态物体的照明信息烘焙到光照探针。

烘焙光照的设置

Light的Mode设为Mixed；

所有混合模式光源使用同一种混合光照模式，Lighting Mode有3个选项：Baked Indirect，Shadowmask和Subtractive。Baked Indirect光照模式仅对光源提供的间接照明部分进行预计算烘焙。我们使用Baked Indirect。

Lightmapping Setting的设置，Lightmap Resolution设置为20，取消Compress Lightmaps勾选，Direcional Mode选择Non-Directional。

勾选对象Mesh Renderer组件上的Contribute Global illumination复选框。勾选Contribute GI的static选项。

采样烘焙光照

间接光照的来源不固定，只能用于漫反射照明，镜面反射通过反射探针实现的。

要获取光照贴图的UV坐标，需要由Unity将其发送到着色器中，我们需要指示渲染管线对每个被烘焙了光照信息的对象都这样做。

光照贴图的UV坐标是顶点数据的一部分，应该在顶点和片元输入结构体中都定义它，在顶点函数中将其转换到片元函数中用于贴图采样。

光照贴图的UV坐标通常由Unity给每个Mesh自动生成，或者在建模软件中设置好后作为Mesh数据的一部分导入进来。

在GI.hlsl中把源码库中的EnityLighting.hlsl文件include进来，从中获取光照贴图和它的采样器。

光照探针

光照探针的原理是什么？

在光照探针的所在位置点上对光照信息进行采样，然后从该光照探针相邻的其它光照探针的位置上对光照信息进行采样，把这些光照信息进行插值运算。

什么是光照探针组？

光照探针组件不能直接挂到游戏对象上面，通常需要依赖光照探针组（Light Probe Group）组件挂接。光照探针组默认情况下在一个立方体空间中包含六个探针，场景中用黄色小球表示光照探针，它可以通过组件上的Edit Light Probes来编辑探针，你可以移动、复制或删除单个探针，场景中光照探针要达到一定数量才能被正确烘焙。

光照探针的布局

最简单的布局方式是将光照探针排列成一个规则的3D网格样式，这样的设置方式简单高效，但会消耗大量内存，因为每一个光照探针本质上是一个球形的且记录了当前采样点周围环境的纹理图像。如果一片区域的照明信息都差不多，那么就没必要使用大量光照探针了。光照探针通常用于照明效果突然改变的场合。

采样光照探针

1. 光照探针的插值数据需要逐对象的传递给GPU。

perObjectData = PerObjectData.Lightmaps | PerObjectData.LightProbe;

2. 在UnityInput.hlsl文件的UnityPerDraw缓冲区中定义7个float4类型变量来接收CPU传递来的光探针数据，它们是代表红色、绿色和蓝光的多项式组件。

3. 在GI.hlsl文件中创建SampleLightProbe方法对光照探针进行采样。首先判断，若该对象正在使用光照贴图就直接返回0，否则返回0和使用SampleSH9()方法得到的光照数据之间最大值。

4. 在GetGI中将光照探针的采样结果和光照贴图的采样结果相加得到最终的漫反射照明。

光照探针代理体(LPPV)

在3D空间的一个位置点上，因为有且只使用一个球面表达式用于描述光照，所以光照探针照明还不适合用于描述光线穿过一个很大的物体时的情况，这种情况下光照会发生很多变动，从而无法精准地进行模拟。光照探针适合小物体，它的照明是基于一个点，因此不适用于大的物体。另一个限制就是，因为球谐函数是在一个球面上对光照信息进行编码，所以对于一个大型的有着平坦表面的物体，或者是一个有着凹面的物体，光照探针照明技术也不适用。如果想在一个大物体上应用光照探针照明，则需要使用光照探针代理体（Light Probe Proxy Volume）组件辅助实现。光探针代理体是一个“解决无法直接使用光探针技术去处理的大型动态游戏对象问题”的组件。

采样LPPV

1. LPPV也需要将每个对象的数据发给GPU。

perObjectData = PerObjectData.Lightmaps | PerObjectData.LightProbe

| PerObjectData.LightProbeProxyVolume

2. 在UnityInput.hlsl的UnityPerDraw缓冲区中添加4个相关属性。

3. 光照探针代理体数据会存储在一个名为unity\_ProbeVolumeSH的3D float纹理中，在GI.hlsl中通过TEXTURE3D\_FLOAT宏获取该纹理，并获取它的采样器。

4. 通过unity\_ProbeVolumeParams的X分量的值判断物体是否使用了LPPV或插值光照探针，如果使用了，必须使用SampleProbeVolumeSH4方法对光探针代理体进行采样，传参分别是对应纹理和采样器、世界空间的顶点位置和法线、一个转换矩阵、unity\_ProbeVolumeParams的Y和Z分量，最后是unity\_ProbeVolumeMin和unity\_ProbeVolumeSizeInv的XYZ分量。

Meta Pass

因为间接漫反射光照会从表面反射出来，因此还应该受到这些表面的漫反射率的影响。Unity使用一个特殊的Meta Pass来确定烘焙时从表面反射出来的光照，然后提供给烘焙系统，从而计算间接光照。

在Lit.shader和Unlit.shader中添加Meta Pass的定义，该Pass的LightMode设置为Meta，关闭剔除功能，include名为MetaPass.hlsl的文件。

Meta Pass可用于生成不同的数据，通过定义一个bool4类型的标记向量unity\_MetaFragmentControl进行通信。在片元函数中进行判断，如果标记了X分量，则需要漫反射率。

自发光表面

有些表面可以发出光，即使场景中没用任何照明，因为它不是真正的光源，所以它不会影响其它表面，但可以参与烘焙光照贴图的计算中，从而照明周围的静态物体。

烘焙自发光

在Meta Pass文件的片元函数中进行判断，如果unity\_MetaFragmentControl的Y分量被标记，则返回自发光的颜色，Alpha为1。

自发光是通过一个单独的Pass进行烘焙的，我们需要对每个材质进行烘焙自发光的设置才行。

我们需要对每个自发光的物体材质的Global Illumination属性设置为Baked，这将在烘焙光照贴图时使用单独的Pass来烘焙自发光。但只是这样还不能起作用，当Global Illumination的切换选项发生改变时，我们应该更新每个材质的Global Illumination Flags标志，这是一个枚举。

烘焙透明物体

1. 不幸的是，Unity的烘焙系统对透明的处理是硬编码的，首先它会根据材质的渲染队列来判断该材质是透明、不透明还是裁切材质。接着将\_MainTex和\_Color属性的Alpha相乘，然后通过\_Cutoff属性对该透明度进行裁剪。我们的Shader中目前有定义\_Cutoff属性，所以还需要定义\_MainTex和\_Color属性（虽然我们有定义\_BaseMap和\_BaseColor属性，它们作用是一样的，只是命名不一样，但是这也没办法，烘焙系统对透明的处理就是硬编码的），然后将这两个属性通过HideInInspector标签使它们不在材质面板中显示，因为我们不希望这两个属性被调节。

2. 我们要确保\_MainTex、\_Color的属性值和\_BaseMap、\_BaseColor属性值保持一致，在CustomShaderGUI脚本中定义一个CopyLightMappingProperties方法，若\_BaseMap、\_BaseColor属性值有修改，则应将其同步到\_MainTex和\_Color中。在OnGUI方法的最后面调用该方法进行追踪。

Mesh Ball

最后我们也对Mesh Ball脚本生成的多个对象实例来添加全局照明的支持，因为对象实例是在运行模式下生成的，因此它们无法被烘焙，但可以使用光照探针存储照明信息。

光照探针的支持

1. 在Mesh Ball脚本中的Graphics.DrawMeshInstanced方法调用中添加额外的5个参数来使用光照探针，第1个参数代表是否投射阴影，我们启用它。第2个布尔参数代表是否接收阴影，我们设为true。第3个参数代表层级，我们使用默认的0。第4个参数代表提供一个渲染相机，我们设置null为所有相机渲染它们。第5个参数代表光照探针插值类型，我们使用CustomProvided。

2. 我们需要为所有对象实例生成插值光照探针，并将它们添加到材料属性块（MaterialPropertyBlock）中。这意味着在配置块时，我们需要访问实例位置。我们可以获取转换矩阵的最后一列来得到实例位置，并将它们存储在临时数组中。然后通过实例化一个SphericalHarmonicsL2类型的数组来创建每个对象实例的光照探针，并使用LightProbes.CalculateInterpolatedLightAndOcclusionProbes来填充数据，该方法需要传递三个参数，对象实例的位置和光照探针数据，第三个参数用于遮挡，我们设置为空。最后通过block.CopySHCoefficientArraysFrom方法将光照探针数据复制到材质属性块中。

也可以对对象实例添加LPPV的支持，因为实例都存在于狭小空间中，这样就不必计算和存储插值光照探针。我们添加一个LightProbeProxyVolume配置字段，如果该字段正在使用，则不需要将光照探针数据添加到材质属性块中，且将LightProbeUsage.UseProxyVolume作为DrawMeshInstanced方法的最后一个参数。

将LPPV 组件添加到挂载Mesh Ball脚本的对象中，并将组件传递给LightProbeProxyVolume字段。可以将LPPV组件的Bounding Box Mode设置为Custom自定义代理体的世界空间区域，能够将所有对象实例包裹进来。