# Unity的渲染路径

渲染路径 决定了光照是如何应用到UnityShader中的。

在pass中设置了 渲染路径，Unity才会把光源和处理后的光照信息都放在某个数据中。

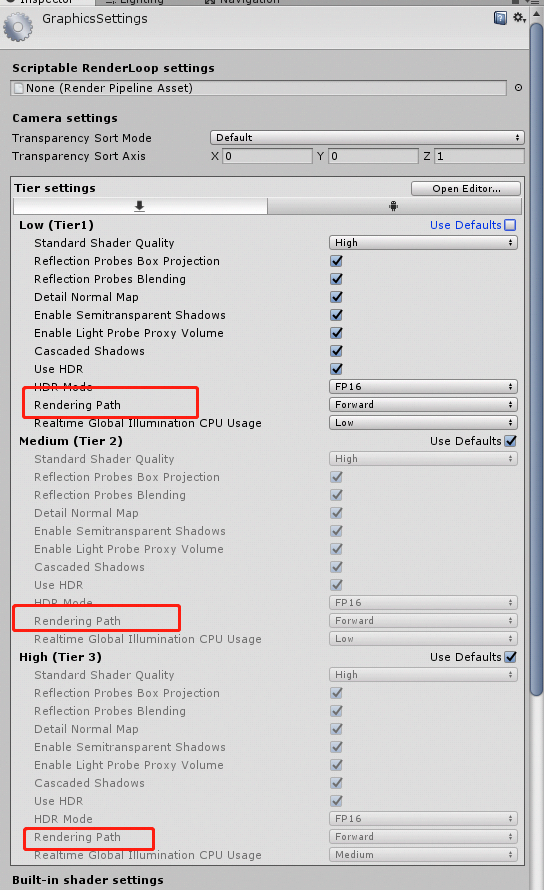
渲染路径：

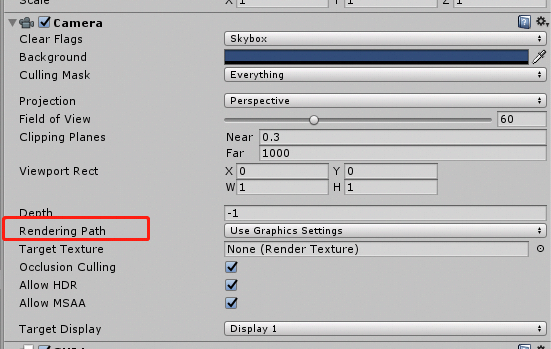
1 前向渲染路径(Forward)

2 延迟渲染路径(Deferred) ==> 更新为 新的延迟渲染路径，但兼容

3 顶点照明渲染路径(Vertex Lit) ==> 弃用，但兼容

设置渲染路径的位置：

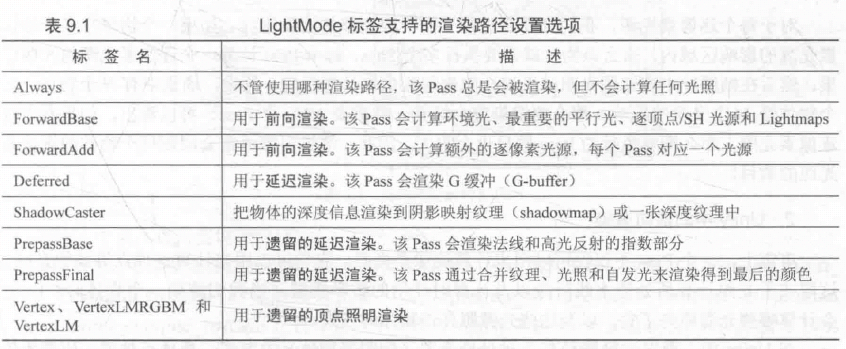
1 统一默认的设置

2 摄像机的单独设置

Pass的标签

Pass{

Tags {“LightMode”=”Forwardbase”}

}

**设置路径的过程，其实是告诉Unity要准备哪些数据，一会这些数据将会用在Pass中。**

**Shader的渲染位置主要在Pass中，pass中有的处理光照，有的不处理光照，有的需要两个处理光照。**

# 前向渲染路径

原理：每进行一次完整的前向渲染，我们需要渲染该对象的渲染图元，并计算两个缓冲区的信息(颜色缓冲区，深度缓冲区)。我们利用深度缓冲来决定一个片元是否可见，如果可见更新颜色缓冲区中的颜色值。

\*\*\* 对于 逐像素光源，我们都需要进行上面一次完整的渲染流程。如果有多个 逐像素光源，则进行多个Pass，每计算一个 逐像素光源的光照结果，就在帧缓冲中把这些光照结果混合起来。

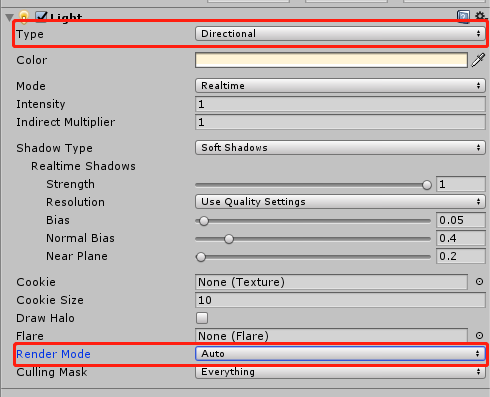
例如，如果有N个物体，M个逐像素光源，需要进行N\*M个Pass，数目很大，需要限制物体的逐像素光照的数目。

前向渲染路径有三种光照方式，即照亮物体的方式。

逐顶点处理、逐像素处理、球谐函数(SH)

光照计算所处流水线阶段 和 计算时使用的数学模型 决定了 Pass 计算那种光照方式

当我们渲染一个物体时， Unity会计算 有哪些光，以及这些光照亮物体的方式。

决定一个光源 的 光照模式 是 光源的类型 和 渲染模式。光源类型：平行光，点光等；渲染模式：Auto, Important, Not Important。

例如，如果设置为 Important，则采用 逐像素光源处理。

**光源排序**

前向渲染中，Unity会对光源排序，排序参考 光源的设置以及对物体的影响程度：

1 场景中 最亮的平行光 逐像素处理

2 如果是 Important 的光源，逐像素处理

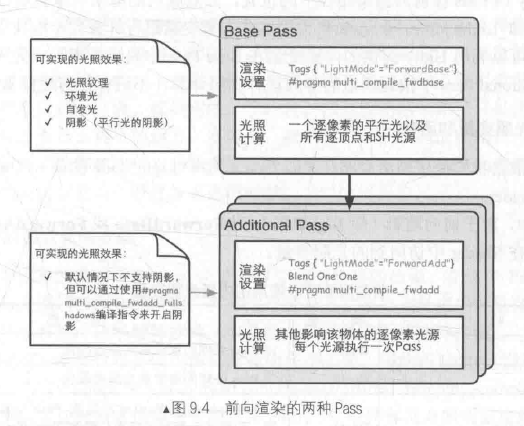
3 如果是 Not Important 的光源，逐顶点 或者 SH处理

4 逐像素的光源数量 不能小于 QualitySetting中的PixelLightCount，否则，会有更多的光源以逐像素的方式渲染。

**光源排序后分类**

排序完成后，一部分光源 按照逐像素处理，不超过4个光源 按照逐顶点处理，剩下的按照SH处理

**光源的处理位置** BasePass 和 AddPass

**注意：**

1 前向渲染，一个Shader通常会定义一个BasePass和一个AddPass，**BasePass一般只会执行一次**，除非进行双面渲染。**AddPass会根据其他逐像素光源的数目多次调用**。

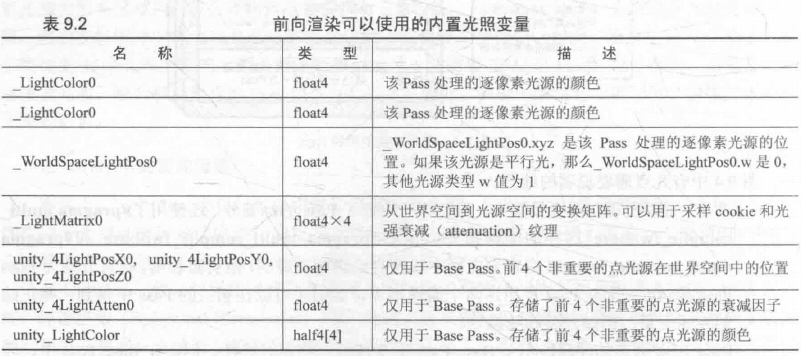
2 **AddPass 需要开启 混合**，因为希望得到剩余的光的总渲染效果。

3 B**asePass 计算 纹理，环境光，自发光，和阴影**。

AddPass默认不计算阴影。除非把

#pragma multi\_compile\_fwdadd 换为 #pragma\_multi\_compile\_fwdadd\_fullshadows

#### 内置的光照变量和函数



**\_LightColor0 该Pass处理的逐像素光源的颜色**

**\_WorldSpaceLightPos0 该Pass处理逐像素光源的位置。如果是平行光w为0，其他为1**

**\_LightMatrix0 类型是float4x4，从世界空间到光源空间的变换**

前四个非重要的**点光源**，只能在**前向渲染**。

**unity\_4LightPosX0** float4 //.

**unity\_4LightPosY0** float4 //

**unity\_4LightPosZ0** float4 //前四个非重要的点光源的世界空间中的位置

**unity\_LightColor** half4[4] //前四个非重要的点光源的颜色。

**unity\_4LightAtten0**  float4 //前四个非重要的点光源的衰减因子。

只能在前向渲染的函数:

float3 **WorldSpaceLightDir**(float4 v);

float3 **UnityWorldSpaceLightDir**(float4 v);

float3 **ObjSpaceLightDir**(float4 v);

float3 **Shade4PointLights**(...); 计算四个点光源的光照