**渲染路径**

**What:**

渲染路径决定了光照是如何应用到Unity Shader中的。

**Why:**

**How:**

**Unity支持哪些类型的渲染路径？**

前向渲染路径，延迟渲染路径和顶点照明渲染路径(废弃)。

**渲染路径的数量**

大多数情况下一个项目只使用一种渲染路径。但有时，我们希望可以使用多个渲染路径，我们可以在每个摄像机的渲染路径设置中进行设置。

**LightMode标签支持哪些渲染路径设置选项？**

Always：该Pass会被渲染，但不会计算任何光照

**ForwardBase**：用于前向渲染，该Pass会计算环境光，最重要的平行光，逐顶点/SH光源和Lightmaps。

**ForwardAdd**：用于前向渲染，该Pass会计算额外的逐像素光源，每个Pass对应一个光源。

Deferred：用于延迟渲染，该Pass会渲染G缓冲。

ShadowCaster：把物体的深度信息渲染到阴影映射纹理或一张深度纹理中。

PrepassBase，PrepassFinal，Vertex：遗留渲染

**前向渲染**

**What:**

**Why:**

**How:**

**前向渲染路径的原理**

对于每个逐像素光照，我们都需要进行一次完整的渲染流程。如果一个物体在多个逐像素光源的影响区域内，那么该物体就需要执行多个Pass，每个Pass计算一个逐像素光源的光照结果，然后在帧缓冲区中把这些光照结果混合起来得到最终的颜色值。

**前向渲染路径有哪3种处理光照的方式？**

逐顶点处理，逐像素处理和球谐函数

**如何决定一个光源使用哪种处理模式？**

1. 光源类型：该光源是平行光还是其他类型的光源；
2. 光源的渲染模式：该光源是否是重要的(Important)。

**Unity判断光源的规则。**

1. 场景中最亮的平行光总是按逐像素处理的；
2. 渲染模式被设置为Not Important的光源，会按逐顶点或者SH处理；
3. 渲染模式被设置为Important的光源，会按逐像素处理；
4. 如果根据以上规则得到的逐像素光源数量小于QualitySetting中设置的PixelLightCount，会有更多的光源以逐像素的方式进行渲染。

**前向渲染有哪两种Pass？**

Base Pass和Additional Pass。

**Base Pass和Additional Pass和执行次数。**

一个UnityShader通常会定义一个Base Pass(可以定义多个，例如需要双面渲染等情况)以及一个Additional Pass。一个BasePass仅会执行一次，而一个Additional Pass会根据影响该物体的其他逐像素的数目被多次调用。

**Base Pass**

**What:**

**Why:**

**How:**

**Base Pass的渲染设置**

Tags{“LightMode” = “ForwardBase”}

#progma multi\_compile\_fwdbase

**Base Pass的光照计算**

一个逐像素的平行光以及所有逐顶点和SH光源。

对于Base Pass来说，它处理的逐像素光源类型一定是平行光。

**可实现的光照效果**

光照纹理，环境光，自发光和阴影(平行光的阴影)

**Additional Pass**

**What:**

**Why:**

**How:**

**渲染设置**

Tags{“LightMode” = ”ForwardAdd”}

Blend One One

#progma multi\_compile\_fwdadd

**光照计算**

其他影响该物体的逐像素光源，每个光源执行一次Pass。

**可实现的光照效果**

默认情况下不支持阴影，但可以通过使用#progma multi\_compile\_fwdadd\_fullshadows编译指令来开启阴影。

**延迟渲染**

**What:**

**Why:**

延迟渲染的效率不依赖于场景的复杂度，而是和我们使用的屏幕空间的大小有关。延迟渲染适合光源数目很多的场景。

**How:**

**延迟渲染的原理。**

延迟渲染主要包含了两个Pass。在第一个Pass中，我们不进行任何光照计算，而是仅仅计算哪些片元是可见的，这主要是通过深度缓冲技术来实现的。当发现一个片元是可见的，我们就把它的相关信息存储到G缓冲区中。然后，在第二个Pass中，我们利用G缓冲区的各个片元信息，例如表面法线，视角方向，漫反射系数等，进行真正的光照计算。

**延迟渲染的一些缺点。**

不支持真正的抗锯齿功能。

不能处理半透明物体。

对显卡有一定要求。