**更复杂的光照**

**渲染路径**

**What:**

**Why:**

渲染路径决定了光照时如何应用到Unity Shader中的。

**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

**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。

**参考：前向渲染可以使用的内置光照变量**

**前向渲染可以使用的内置光照函数**

**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。

**光源**

**What:**

**Why:**

**How:**

**Unity支持哪些光源类型？**

平行光，点光源，聚光灯和面光源(用于烘焙)。

**如何计算光照的衰减？**

1. 使用纹理查找；
2. 使用数学公式计算；

**阴影**

**What:**

**Why:**

**How:**

**为什么会产生阴影？**

阴影区域的产生是因为光线无法到达这些区域。

**Unity使用的阴影技术及其原理**

Shadow Map，首先把摄像机的位置放在与光源重合的位置上，那么场景中的阴影区域就是那些摄像机看不到的地方。

Shadow Map阴影映射纹理本质上也是一张深度图，它记录了从该光源的位置出发，能看到的场景中距离它最近的表面位置。

**Unity中物体如何向其他物体投射阴影？**

Unity使用一个额外的Pass来专门更新光源的阴影映射纹理，这个Pass就是LightMode标签被设置为ShadowCaster的Pass。

因此，当开启了光源的阴影效果后，底层渲染引擎首先会在当前渲染物体的Unity Shader中找到LightMode为ShadowCaster的Pass，如果没有，它就会在Fallback指定的Unity Shader中继续寻找，如果仍然没有找到，该物体就无法向其他物体投射阴影(但它仍然可以接收来自其他物体的阴影)。当找到一个LightMode为ShadowCaster的Pass后，Unity会使用该Pass来更新光源的阴影映射纹理。

**屏幕空间的阴影映射技术**

在传统的阴影映射纹理的实现中，我们会在正常渲染的Pass中把顶点位置变换到光源空间下，以得到它在光源空间中的三维位置信息。然后，我们使用xy分量对阴影映射纹理进行采样，得到阴影映射纹理中该位置的深度信息。如果该深度值小于该顶点的深度值，那么说明该点位于阴影中。

但在Unity中，使用了屏幕空间的阴影映射技术。Unity首先会通过调用LightMode为ShadowCaster的Pass来得到可投射阴影的光源的阴影映射纹理以及摄像机的深度纹理。然后，根据光源的阴影映射纹理和摄像机的深度纹理来得到屏幕空间的阴影图。如果摄像机的深度图中记录的表面深度大于转换到阴影映射纹理中的深度值，就说明该表面虽然是可见的，但却处于该光源的阴影中。通过这样的方式，阴影图就包含了屏幕空间中所有有阴影的区域。如果我们想要一个物体接收来自其他物体的阴影，只需要在Shader中对阴影图进行采样。由于阴影图是屏幕空间下的，因此，我们首先需要把表面坐标从模型空间变换到屏幕空间中，然后使用这个坐标对阴影图进行采样即可。

**一个物体接收来自其他物体的阴影，以及它向其他物体投射阴影过程的总结。**

1. 如果我们想要一个物体接收来自其他物体的阴影，就必须在Shader中对阴影映射纹理进行采样，把采样结果和最后的光照结果相乘来产生阴影效果。
2. 如果我们想要一个物体向其他物体投射阴影，就必须把该物体加入到光源的阴影映射纹理的计算中，从而让其他物体在对阴影映射纹理采样时可以得到该物体的相关信息。在Unity中，这个过程是通过为该物体执行LightMode为ShadowCaster的Pass来实现的。如果使用了屏幕空间的投影映射技术，Unity还会使用这个Pass产生一张摄像机的深度纹理。

**透明物体的阴影**

透明度测试的阴影：把Fallback设置为Transparent/Cutout/VertexLit，把Cast Shadows属性设置为Two Sided。

透明度混合的阴影：Unity默认是不处理透明度混合的阴影，或者当成不透明物体处理。