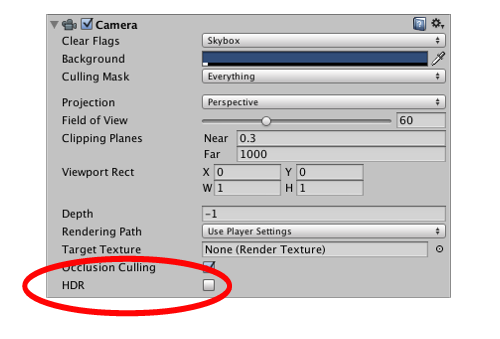
**高动态范围渲染**

在标准渲染中，像素的红色、绿色和蓝色值均由 0 到 1 范围内的分数表示，其中 0 表示零强度，1 表示显示设备的最大强度。虽然这很容易使用，但它并不能准确反映在现实生活场景中光照的运作方式。人眼倾向于适应局部光照条件，因此在光线昏暗的房间中看起来是白色的物体实际上可能还不如在强烈日光下看起来是灰色的物体明亮。此外，眼睛对昏暗范围的亮度差异比对明朗范围的亮度差异更敏感。

如果能调整渲染使像素值的范围来更准确地反映其在真实场景中的光照情况，则可以实现更真实的视觉效果。虽然这些值最终需要映射回显示设备上的可用范围，但任何中间计算（例如 Unity 的图像效果）都将提供更真实的结果。**高动态范围 (HDR)** 渲染的精髓是允许图形的内部表示使用 0 到 1 范围之外的值。

**使用 HDR**

应使用摄像机组件上的设置为每个摄像机单独启用 HDR：



当 HDR 处于激活状态时，场景将渲染为 HDR 图像缓冲区，该缓冲区可以容纳 0 到 1 范围之外的像素值。然后，该缓冲区由后期处理效果使用，例如[后期处理栈](file:///E:\\UnityDocumentation\\Manual\\PostProcessing-Stack.html)中的[泛光 (Bloom)](file:///E:\UnityDocumentation\Manual\PostProcessing-Bloom.html) 效果。HDR 图像随后被转换为标准低动态范围 (LDR) 图像以便发送给显示设备。这通常通过色调映射（属于[颜色分级](file:///E:\\UnityDocumentation\\Manual\\PostProcessing-ColorGrading.html)管线的一部分）来完成。转换为 LDR 必须在后期处理管线中的某个时刻进行，但如果之后有可能应用只针对 LDR 的后期处理效果，所以可能不是最后一步。为方便起见，一些后期处理效果可以在应用 HDR 效果后自动转换为 LDR（请参阅下面的“脚本”部分）。

**色调映射**

色调映射是指将 HDR 值映射回 LDR 范围的过程。有许多不同的技术，但对一个项目有益的技术对于另一个项目来说可能并非最佳选择。[后期处理栈](file:///E:\\UnityDocumentation\\Manual\\PostProcessing-Stack.html)中已经包含了多种色调映射技术。要使用这些技术，可从 [Asset Store](https://www.assetstore.unity3d.com/en/#!/content/83912) 下载**后期处理栈 (Post-processing stack)**。可在[颜色分级](file:///E:\\UnityDocumentation\\Manual\\PostProcessing-ColorGrading.html)文档中找到色调映射类型的详细描述。

在 HDR 中渲染的异常明亮的场景。如果不执行色调映射，大部分像素会看起来超出范围。 与上面相同的场景。但这一次，色调映射将大多数强度映射到了更合理的区间内。请注意，自适应色调映射甚至可以在上图和此图之间进行混合，从而模拟捕获介质（例如眼睛、摄像机）的自适应特性。

**HDR 的优点**

* 在高强度区域不会丢失颜色
* 更好地支持泛光和发光效果
* 减少低频光照区域的条带

**HDR 的缺点**

* 使用浮点渲染纹理（渲染速度较慢，需要更多 VRAM）
* 不支持硬件抗锯齿（但您可以使用[抗锯齿后期处理效果](file:///E:\\UnityDocumentation\\Manual\\PostProcessing-Antialiasing.html)来平滑边缘）
* 并非所有硬件都支持

**使用注意事项**

**前向渲染**

在前向渲染模式下，仅当您具有后期处理效果时才支持 HDR。这是出于性能考虑的原因。如果您没有后期处理效果，则不存在色调映射，并且将发生强度截断。在这种情况下，场景将直接渲染到不支持 HDR 的后备缓冲区。

**延迟渲染**

在 HDR 模式下，光照缓冲区也被分配为浮点缓冲区。这样可减少光照缓冲区中的条带。即使没有后期处理效果，延迟渲染也支持 HDR。

**脚本**

可向后期处理效果脚本中添加 **ImageEffectTransformsToLDR** 属性以表明目标缓冲区应在 LDR 中，而非在 HDR 中。从本质上讲，这意味着脚本可以在应用其 HDR 后期处理效果后自动转换为 LDR。有关更多详细信息，请参阅[编写后期处理效果](file:///E:\\UnityDocumentation\\Manual\\PostProcessingWritingEffects.html)。

# 渲染路径

Unity 支持不同的渲染路径。您应该根据自己的游戏内容和目标平台/硬件而选择使用哪一个渲染路径。不同的渲染路径具有不同的性能特征，主要影响光照和阴影。请参阅[渲染管线](file:///E:\\UnityDocumentation\\Manual\\SL-RenderPipeline.html)以了解技术细节。

项目所用的渲染路径是在[图形设置 (Graphics Settings)](file:///E:\\UnityDocumentation\\Manual\\class-GraphicsSettings.html)中选择的。此外，可针对每个[摄像机](file:///E:\\UnityDocumentation\\Manual\\class-Camera.html)重写渲染路径。

如果显卡无法处理选定的渲染路径，Unity 将自动使用较低保真度的渲染路径。例如，在无法处理延迟着色 (Deferred Shading) 的 GPU 上将使用前向渲染 (Forward Rendering)。

## 延迟着色

延迟着色是具有最大光照和阴影保真度的渲染路径，最适合有许多实时光源的情况。此渲染路径需要一定程度的硬件支持。

有关更多详细信息，请参阅[延迟着色页面](file:///E:\\UnityDocumentation\\Manual\\RenderTech-DeferredShading.html)。

## 前向渲染

前向渲染是传统的渲染路径。它支持所有典型的 Unity 图形功能（法线贴图、每像素光照、阴影等）。但是，在默认设置下，仅少量最亮的光线才在每像素光照模式下渲染。其余光线是在对象顶点上或每个对象上计算的。

有关更多详细信息，请参阅[前向渲染页面](file:///E:\UnityDocumentation\Manual\RenderTech-ForwardRendering.html)。

## 旧版延迟 (Legacy Deferred)

旧版延迟（Light Prepass，即光照 prepass）类似于延迟着色，只是采用不同的技术并进行不同的折中。它不支持 Unity 5 基于物理的标准着色器。

有关更多详细信息，请参阅[延迟光照页面](file:///E:\UnityDocumentation\Manual\RenderTech-DeferredLighting.html)。

## 旧版顶点光照 (Legacy Vertex Lit)

旧版顶点光照是具有最低光照保真度且不支持实时阴影的渲染路径。它是前向渲染路径的一个子集。

有关更多详细信息，请参阅[顶点光照页面](file:///E:\UnityDocumentation\Manual\RenderTech-VertexLit.html)。

**注意：**使用正交投影 (Orthographic projection) 时不支持延迟渲染。如果摄像机的投影模式设置为正交模式，则会覆盖这些值，并且摄像机将始终使用前向渲染。

## 渲染路径比较

|  | **延迟** | **前向** | **旧版延迟** | **顶点光照** |
| --- | --- | --- | --- | --- |
| **功能** |  |  |  |  |
| 每像素光照（法线贴图、光照剪影） | 是 | 是 | 是 | - |
| 实时阴影 | 是 | 带有警告 | 是 | - |
| 反射探针 | 是 | 是 | - | - |
| 深度和法线缓冲区 | 是 | 其他渲染pass | 是 | - |
| 软粒子 | 是 | - | 是 | - |
| 半透明对象 | - | 是 | - | 是 |
| 抗锯齿 | - | 是 | - | 是 |
| 光照剔除遮罩 | 受限 | 是 | 受限 | 是 |
| 光照保真度 | 全部每像素 | 部分每像素 | 全部每像素 | 全部每顶点 |
| **性能** |  |  |  |  |
| 每像素光照的成本 | 照射像素数量 | 像素数量 \* 照射对象数量 | 照射像素数量 | - |
| 正常渲染对象的次数 | 1 | 每像素光照的数量 | 2 | 1 |
| 简单场景的开销 | 高 | 无 | 中 | 无 |
| **平台支持** |  |  |  |  |
| PC (Windows/Mac) | Shader Model 3.0+ 和 MRT | 所有 | Shader Model 3.0+ | 所有 |
| 移动端 (iOS/Android) | OpenGL ES 3.0 和 MRT、Metal（在搭载 A8 或更高版本 SoC 的设备上） | 所有 | OpenGL ES 2.0 | 所有 |
| 游戏主机 | XB1、PS4 | 所有 | XB1、PS4、360 | - |

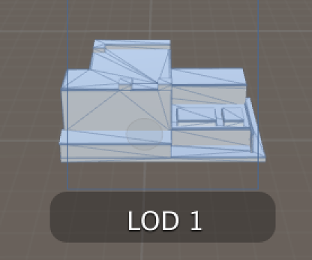
# 细节级别 (LOD)

当场景中的某个游戏对象距离摄像机很远时，该游戏对象的细节的可见程度将大大降低。但是，即使对象的细节不会让人注意到，系统也会使用相同数量的三角形来渲染对象。使用一种称为\_细节级别\_ (LOD) 的优化技术进行渲染，则可以随着某个对象与摄像机的距离增加而减少渲染对象的三角形数量。只要全部的对象不是同时靠近摄像机，LOD 便能降低硬件的负载并提高渲染性能。

在 Unity 中，可使用 **LOD Group** 组件为对象设置 LOD 渲染。[组件参考页面](file:///E:\UnityDocumentation\Manual\class-LODGroup.html)提供了完整详解，但下图显示了用于渲染对象的 LOD 级别如何随着对象与摄像机的距离的变化而变化。第一张图显示的是 LOD 级别 0（细节化程度最高）。请注意网格中大量的小三角形：

摄像机处于 LOD 0

第二张图显示了对象距离变远时使用的较低级别的细节模型。请注意，网格的细节程度已降低（三角形更大，数量更少）：

摄像机处于 LOD 1

由于 LOD 级别的设置在某种程度上取决于目标平台和可用的渲染性能，因此 Unity 允许您在[质量设置 (Quality Settings)](file:///E:\UnityDocumentation\Manual\class-QualitySettings.html) 中设置最高 LOD 级别和 LOD 偏差偏好（即，处于阈值距离时提高还是降低 LOD 级别）。

## 导入对象时的 LOD 命名规则

如果创建一组名称以 \_LOD0、\_LOD1、\_LOD2 等结尾的网格，结尾数字对应于所需的 LOD 级别，在将这些模型导入时，系统将自动创建并将其设置为对象的 LOD 组。例如，如果网格的基本名称是 Player\_，则可以创建名为 Player\_LOD0、Player\_LOD1 和 Player\_LOD2 的文件，以生成具有三个 LOD 级别的对象。编号规则为 LOD 0 是细节程度最高的模型，而数字越大，对应的细节程度越低。