

简单照明样本

*此示例与 2015 年 11 月的 Xbox One XDK 或更高版本兼容。*

# 描述

此示例演示如何创建静态Direct3D 11顶点，索引和常量缓冲区，以绘制使用静态和动态Lambertian光照点亮的索引几何体。

该示例渲染了一个由两个灯光点亮的大型立方体，一个白色和一个红色，也表示为立方体。白光是静止的，而红光围绕中心立方体旋转。中央立方体也旋转。该动作允许您从不同角度观察彩色灯光的效果。



# 使用样本

|  |  |  |
| --- | --- | --- |
| 操作 | 游戏手柄 | 键盘 |
| 退出 | 查看按键 | 退出 |

# 实施说明

该示例使用三个着色器渲染场景 - 顶点着色器（“TriangleVS”）和两个像素着色器（“LambertPS”，“SolidColorPS”。）已编译的着色器块在 CreateDeviceDependentResources 中加载，然后，在创建着色器资源时引用。 所有着色器都在同一个HLSL包含文件中定义，“SimpleLighting.hlsli”和三个存根着色器包含此文件。为不同的入口点编译每个存根着色器，以便创建三个着色器blob。

该场景的几何体由静态顶点和索引缓冲区组成，每个缓冲区填充有 24 个顶点的数据，代表立方体的六个四边形。这两个缓冲区在 CreateDeviceDependentResources 中创建，并立即填充有在 D3D11\_SUBRESOURCE\_DATA 结构的初始化数据中提供的数据。这是最高效的 DirectX 11 初始化方法。

对于这个非常简单的场景，所有着色器常量都集中在一个包含以下内容的常量缓冲区中：

* 世界，视图和投影矩阵
* 光线方向和颜色
* 纯色

对于更复杂的场景，通常会根据常量更新的频率将常量拆分为多个缓冲区。

由于大立方体和红光是动画的，因此，必须每帧更新一些常量。使用以下策略更新常量缓冲区:

1. 使用 D3D11\_MAP\_DISCARD 调用 ID3D11DeviceContext::Map，以创建 D3D11\_MAPPED\_SUBRESOURCE。
2. 将所有常量的值复制到映射的资源中
3. 在调用 DrawIndexed 之前取消映射资源

使用 D3D11\_MAP\_DISCARD 时

* 驱动程序本质上会分配一个新的缓冲区，因此，您应该假设您必须更新着色器将使用的缓冲区中的所有常量。相比之下，使用 D3D11\_MAP\_WRITE\_NO\_OVERWRITE 时缓冲区可以包含来自先前更新的有效数据。
* 如果 GPU 仍在使用先前映射的缓冲区，则您将获得一个新缓冲区。如果您过度使用 D3D11\_MAP\_DISCARD，则可能最终会停止运行，因为您必须等待更多的内存可用。

# 更新历史记录

2016 年 5 月，已弃用 ATG 示例框架的所有依赖项删除。2017 年 4 月，它更新为在 Xbox One X 上运行时支持 4k 交换链。

# 隐私声明

编译和运行示例时，示例可执行文件的文件名将发送给Microsoft以帮助跟踪示例使用情况。要选择退出此数据收集，您可以删除Main.cpp中标记为“Sample Usage Telemetry”的代码块。

有关 Microsoft 隐私政策的更多信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/zh-cn/privacystatement/)。