

SimplePBR 示例

# *此示例与 Windows 10 2018 年 10 月更新 SDK （17763） 和 Microsoft 游戏开发工具包（2020 年 11 月）兼容。*

# 说明

此示例演示通过 GDK 使用 DirectX 12 在 Xbox 系列主机、Xbox One主机和电脑上基于物理的渲染 (PBR)。此示例使用以下参数将正向渲染的迪士尼样式 PBR 实现为独立着色器：

1. Albedo：无照明的底层 RGB 颜色
2. 普通贴图：未压缩的 3 通道普通贴图（Y 正），

RMA 贴图，其中指定了：

1. 粗糙度： [0...1], 普通分布表示反射突出显示大小和形状。粗糙度根据迪士尼纸张进行缩放。
2. 金属：（通常为 0 或 1，可以混合），控制 albedo 的重整索引、反射和漫射分布。
3. 环境遮挡：缩放反射和漫射贡献的值 [0...1]。

参数只能表示为常量，也可以仅表示为纹理（但不能表示混合）。着色器支持基于图像的照明（具有预先计算的漫射和反射贴图）和定向光。

有关 PBR 的更多详细信息，请参阅本文档末尾的实现/参考部分。

A picture containing text, indoor

Description automatically generated

# 生成示例

如果使用 Xbox One 开发工具包，请将活动解决方案平台设置为 Gaming.Xbox.XboxOne.x64。

Gaming.Xbox.Scarlett.x64 配置用于部署到 Xbox 系列设备。

此外，可以使用 Gaming.Xbox.Desktop.x64 活动解决方案平台在电脑上运行该示例。

*有关详细信息，请参阅 GDK 文档中的*“运行示例”*。*

# 使用示例

可以使用本文档“控件”部分中列出的旋转相机操作来导航示例中的渲染场景。所有设备都支持游戏板控件，而鼠标和键盘支持仅在电脑上可用。

# 控制

|  |  |  |
| --- | --- | --- |
| 操作 | 游戏板 | 键盘和鼠标 |
| 沿视图向量旋转/转换相机 | 左控制杆 | 鼠标滚轮 |
| 旋转相机 | 右控制杆 | 按住 LMB + 鼠标 |
| 平移相机 | 方向键 | WASD 或箭头键 |
| 退出 | “视图”按钮 | 退出 |

# 实施说明

PBREffect 类包装着色器的实现。着色器具有两个配置：常量和纹理化。常量配置主要用于调试。在纹理化配置中，输入参数（Albedo 和粗糙度、Metallic、AO）指定为纹理。

如果要创建纹理着色器，请使用 EffectFlags 枚举：

m\_effect = std::make\_unique<DirectX::PBREffect>(device, EffectFlags::Texture, pipelineState);

如果要设置纹理参数，只需为每个纹理和采样器传入描述符：

m\_effect->SetSurfaceTextures(m\_descriptors->GetGpuHandle(AlbedoIndex),

m\_descriptors->GetGpuHandle(NormalIndex),

m\_descriptors->GetGpuHandle(RoughnessMetallicAOIndex),

commonStates->AnisotropicWrap());

着色器作为 Visual Studio 项目的一部分进行编译，并分解为三个文件

1. PBREffect\_VSConstant – 共享顶点着色器
2. PBREffect\_PSConstant – 常量参数像素着色器
3. PBREffect\_PSTextured – 纹理化参数像素着色器

还有两个 HLSL，包括

1. PBREffect\_Math – BRDF 共享数学函数等
2. PBREffect\_Common – 根签名、常量和共享照明函数“PBR\_LightSurface”。

## 照明

PBREffect 支持定向照明和基于图像的照明。调用方必须提供预计算的辐射纹理（用于漫射环境照明）和辐射纹理（用于反射环境照明）。 纹理应为 HDR 格式的 DDS 立方体贴图。

调用方还必须指定辐射纹理中的 MIP 级别数。有关为 PBR 生成预计算贴图的更多详细信息 >请参阅“[用于基于物理渲染的AMD Cubemapgen](https://seblagarde.wordpress.com/2012/06/10/amd-cubemapgen-for-physically-based-rendering/) ”。

m\_effect->SetIBLTextures(

m\_descriptors->GetGpuHandle(m\_radTexDescIndex),

m\_radianceTexture->*GetDesc*().*MipLevels*,

m\_descriptors->GetGpuHandle(m\_irrTexDescIndex),

m\_commonStates->AnisotropicClamp());

可选，调用方还可以使用 SetLight\* 方法指定定向光。着色器将混合定向和图像照明。

## 引用

<https://www.allegorithmic.com/system/files/software/download/build/PBR_Guide_Vol.1.pdf>

<https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf>

<http://blog.selfshadow.com/publications/s2015-shading-course/>

<http://graphicrants.blogspot.com/2013/08/specular-brdf-reference.html>

<https://github.com/dariomanesku/cmftStudio>

# 更新历史记录

2021/09/20 –SimplePBR 示例初始发布

2021/10/15 –修复了退出示例后的 GPU 挂起问题，并添加了深色 UI 矩形以提高文本可读性。 还添加了对 1440p 的支持。

# 隐私声明

在编译和运行示例时，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。若要选择退出此数据收集，你可以删除 Main.cpp 中标记为“示例使用遥测”的代码块。

有关 Microsoft 的一般隐私策略的详细信息，请参阅 [Microsoft 隐私声明](https://privacy.microsoft.com/en-us/privacystatement/)。