

SimplePBR サンプル

# *このサンプルは Windows 10 October 2018 Update SDK (17763) および Microsoft Game Development Kit (2020 年 11 月)と互換性があります*

# 説明

このサンプルでは、Xbox Series 本体、Xbox One 本体、PC 上で GDK を介して DirectX 12 を使用して、物理ベース レンダリング (PBR) を行う方法を示します。このサンプルでは、次のパラメーターを使用して、スタンドアロン シェーダーとして前方レンダリングの Disney スタイル PBR を実装します。

1. Albedo: 照明なしの基本 RGB カラー
2. Normal map: 圧縮されていない 3 チャンネル法線マップ (Y 陽性)

次を指定する RMA マップも含まれます。

1. Roughness: [0...1]、法線分布は反射ハイライトのサイズと形状を意味します。粗さは、Disneyの論文に従ってスケーリングされます。
2. Metallic：(通常は 0 または 1 をブレンドできます)、アルベドの反射と拡散分布のインデックスを制御します。
3. Ambient occlusion:反射と拡散のコントリビューションをスケーリングする値 [0...1] です。

パラメーターは定数としてのみ、またはテクスチャとしてのみ表現できます (ただし、ミックスとして表現することはできません)。シェーダーは、画像ベースの照明 (事前計算済みの拡散マップと反射マップを使用) と方向ライトをサポートします。

PBR の詳細については、このドキュメントの最後にある実装/参照セクションを参照してください。

A picture containing text, indoor

Description automatically generated

# サンプルのビルド

Xbox One 開発キットを使用している場合、アクティブ ソリューション プラットフォームを Gaming.Xbox.XboxOne.x64 に設定します。

Gaming.Xbox.Scarlett.x64 構成は、Xbox Series デバイスに展開するための構成です。

さらに、このサンプルは、Gaming.Xbox.Desktop.x64 アクティブ ソリューション プラットフォームを使用して PC で実行できます。

*詳細については、GDK のドキュメントの*「サンプルの実行」*を参照してください。*

# サンプルの使用方法

サンプル内のレンダリングされたシーンは、このドキュメントの「コントロール」セクションに示されているように、オービット カメラ アクションを使用して移動できます。ゲームパッド コントロールはすべてのデバイスでサポートされていますが、マウスとキーボードのサポートは PC でのみ使用できます。

# コントロール

|  |  |  |
| --- | --- | --- |
| 操作 | ゲームパッド | キーボードとマウス |
| ビュー ベクターに沿ってカメラを回転/平行移動する | 左サムスティック | マウス ホイール |
| カメラの旋回 | 右スティック | LMB 長押し + マウス |
| [カメラ] をパンします。 | 方向パッド | WASD キーまたは方向キー |
| 終了 | ビュー ボタン | 脱出 |

# 実装上の注意

PBREffect クラスはシェーダーの実装をラップします。シェーダーには、定数とテクスチャの 2 つの構成があります。定数構成は主にデバッグ用です。テクスチャ構成では、入力パラメーター (Albedo、および Roughness、Metallic、AO) がテクスチャとして指定されます。

テクスチャ シェーダーを作成するには、EffectFlags 列挙型を使用します。

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

テクスチャ パラメーターを設定するには、各テクスチャとサンプラーの記述子を渡すだけです。

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

m\_descriptors->GetGpuHandle(NormalIndex),

m\_descriptors->GetGpuHandle(RoughnessMetallicAOIndex),

commonStates->AnisotropicWrap());

シェーダーはVisual Studio プロジェクトの一部としてコンパイルされ、3 つのファイルに分割されます

1. PBREffect\_VSConstant –共有頂点シェーダー
2. PBREffect\_PSConstant –定数パラメーターピクセル シェーダー
3. PBREffect\_PSTextured –テクスチャ パラメーターピクセル シェーダー

2 つの HLSL インクルードもあります

1. PBREffect\_Math – BRDF などの共有数式関数
2. PBREffect\_Common–ルートシグネチャ、定数、共有照明関数「PBR\_LightSurface」。

## 照明

PBREffect は、方向と画像ベースの両方の照明をサポートします。呼び出し元は、事前計算済みの非輝度テクスチャ (拡散環境の照明の場合) と輝度テクスチャ (反射環境の照明の場合) を指定する必要があります。 テクスチャは、HDR 形式の DDS キューブマップである必要があります。

呼び出し元は、輝度テクスチャ内の MIP レベルの数も指定する必要があります。PBR の事前計算済みマップの生成の詳細については、物理ベースのレンダリングに関する 「[AMD Cubemapgen](https://seblagarde.wordpress.com/2012/06/10/amd-cubemapgen-for-physically-based-rendering/)」 を参照してください。

m\_effect->SetIBLTextures(

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

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

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

m\_commonStates->AnisotropicClamp());

オプションでは、呼び出し元は SetLight\* メソッドを使用して方向ライトを指定することもできます。シェーダーは、方向とイメージの照明をブレンドします。

## 参考資料

<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>

# 更新履歴

2021/09/20 – SimplePBR サンプルの初期リリース

2021/10/15 – サンプルを終了した後の GPU ハングの問題を修正し、テキストの読みやすさを向上させるために UI 四角形に暗い色を追加しました。 1440p のサポートも追加されました。

# プライバシーに関する声明

サンプルをコンパイルして実行する場合、サンプルの使用状況を追跡するために、サンプル実行ファイルのファイル名が Microsoft に送信されます。このデータ コレクションからオプトアウトするには、Main.cpp の「Sample Usage Telemetry」というラベルの付いたコードのブロックを削除します。

全般的な Microsoft のプライバシー ポリシーの詳細については、「[Microsoft プライバシー ステートメント](https://privacy.microsoft.com/en-us/privacystatement/)」を参照してください。