

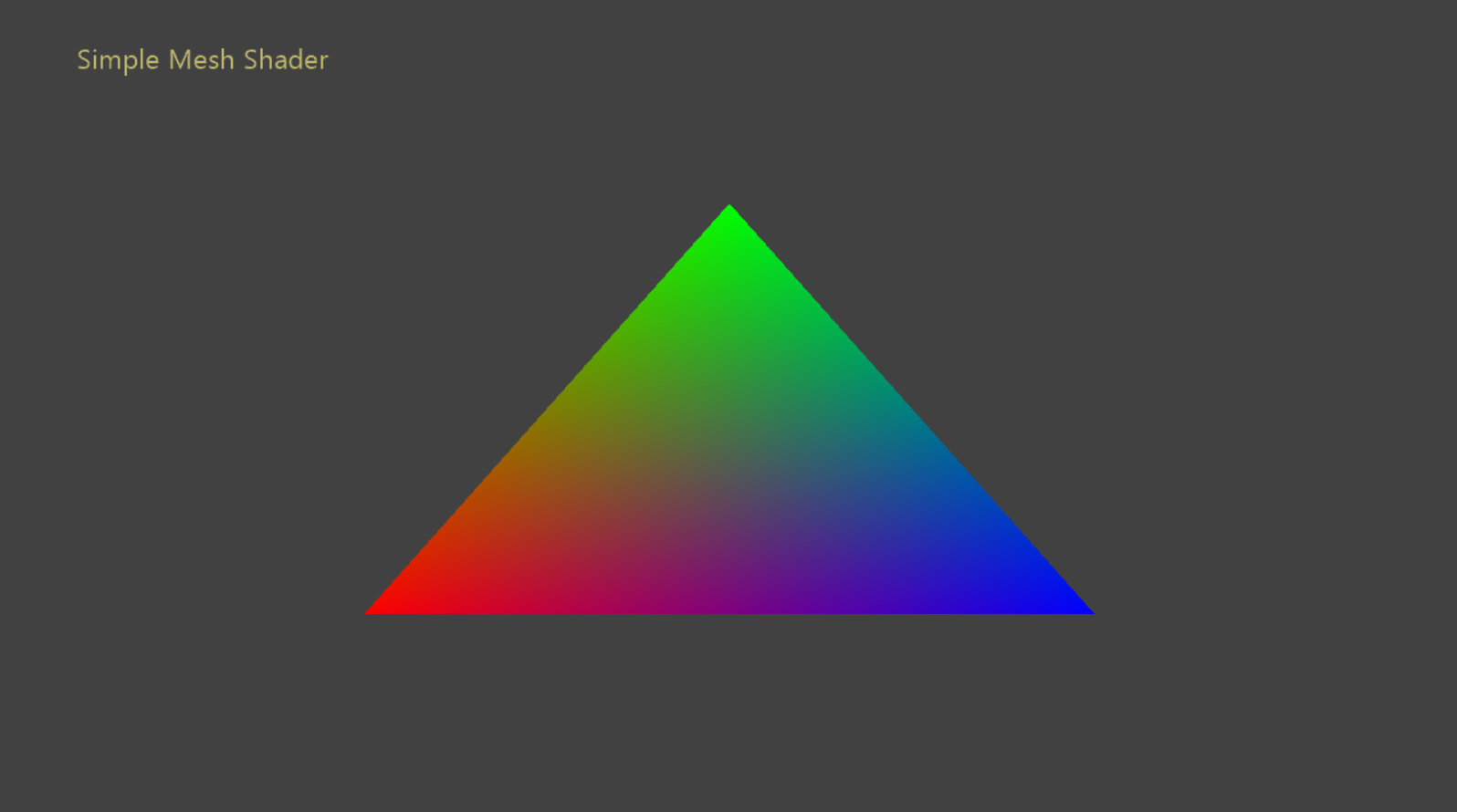
簡単なメッシュ シェーダーのサンプル

*このサンプルは、Microsoft ゲーム開発キットのプレビュー (2019 年 11 月) および PC 用 Windows 10 “20H1” Insider に対応しています。*

# 説明

このサンプルは、長らく利用されてきた「簡単な三角形のサンプル」と似ていますが、メッシュ シェーダーを使用する点が異なります。このサンプルの主な目的は、メッシュ シェーダー パイプラインを初期化して、PC と Scarlett GXDK の両方で動作させるのに必要なすべてのことを実演することです。

注:Xbox One シリーズのコンソールはメッシュ シェーダーをサポートしていないため、このようなビルドの構成はできません。



# サンプルのビルド

Project Scarlett を使用している場合は、アクティブなソリューション プラットフォームを Gaming.Xbox.Scarlett.x64 に設定します。

適切なハードウェアと Windows 10 リリースの PC を使用している場合は、アクティブなソリューション プラットフォームを Gaming.Desktop.x64 に設定します。

このサンプルは、Xbox One をサポートしていません。

*詳細については、GDK 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

|  |  |
| --- | --- |
| 動作 | コントローラー |
| 終了 | ビュー ボタン |

# 実装に関する注意事項

このサンプルでは次の手順を実演します。

1. DirectX12 でレンダリングするために ID3D12Device と request API オブジェクトを初期化します。
2. ID3D12Device::CheckFeatureSupport() 関数を使用して、メッシュ シェーダー機能のサポートを確認します。
3. ID3D12Device2::CreatePipelineState() 関数を使用して、メッシュ シェーダー パイプラインを作成します。
4. ルート署名、パイプラインの状態、リソースをコマンド リストにバインドします。必要なパラメーターを使用して、ID3D12GraphicsCommandList6::DispatchMesh() 関数を使用してメッシュを扱うパイプラインを実行します。

# 更新履歴

2019 年 10 月 31 日 – サンプルの作成。

# 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と、サンプルの使用状況を追跡するため、サンプル実行可能ファイルのファイル名が Microsoft に送信されます。このデータ収集を無効にするには、「Sample Usage Telemetry」とラベル付けされた Main.cpp 内のコードのブロックを削除します。

Microsoft のプライバシー方針の詳細については、「[Microsoft プライバシーに関する声明](https://privacy.microsoft.com/en-us/privacystatement/)」を参照してください。