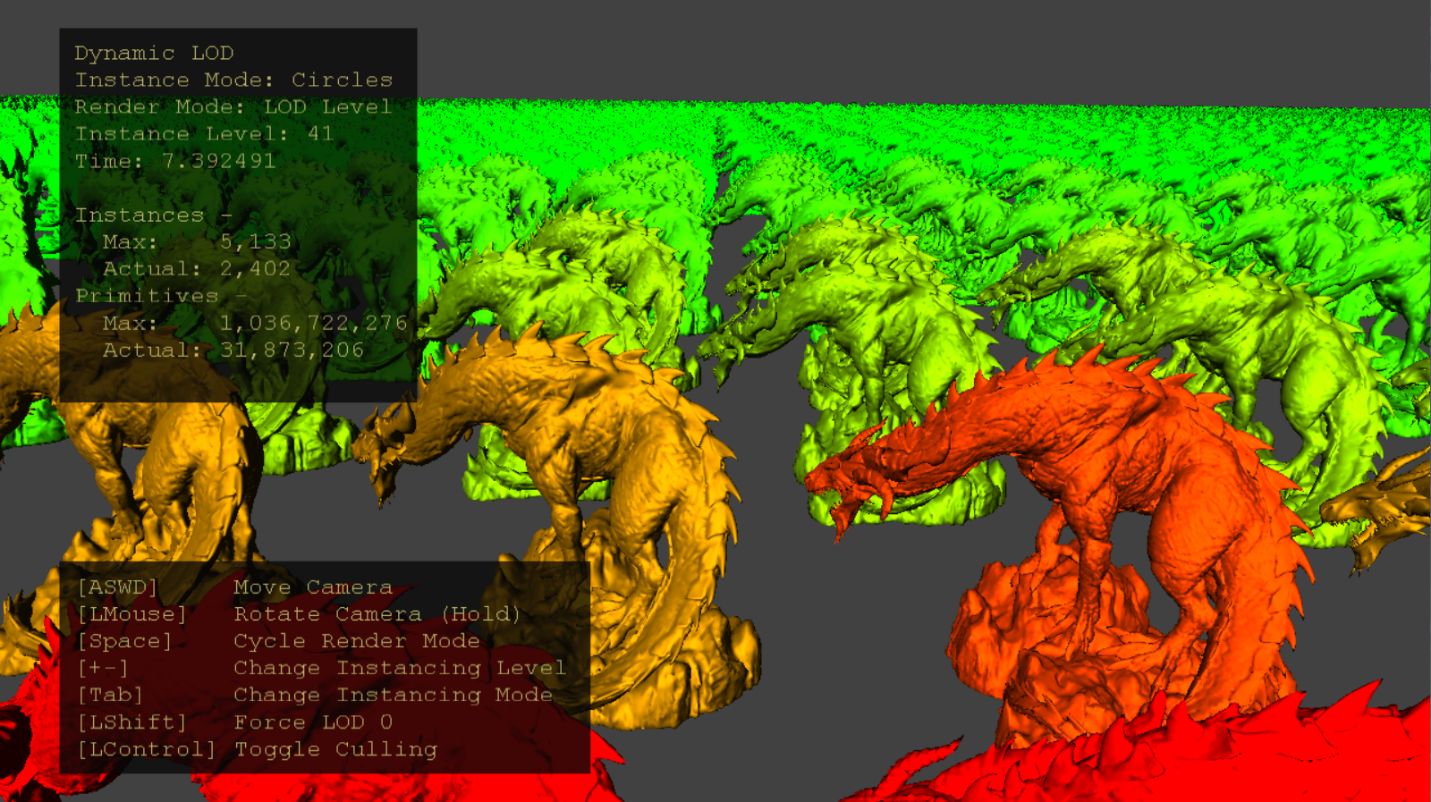


動的 LOD サンプル

*このサンプルは Microsoft Game Development Kit (2020 年 6 月) および Windows 10 (バージョン 2004) 2020 年 5 月更新プログラムと互換性があります*

# 説明

このサンプルでは、増幅シェーダーを活用して、任意の数のインスタンスに対して、インスタンスごとの視錐台カリングとメッシュ詳細レベル (LOD) の選択を GPU 上で完全に行う方法を示します。メッシュ シェーダー パイプラインを使用すると、この手法は 1 つの増幅とメッシュ シェーダー パイプラインの状態オブジェクトに適切に収まります。



# 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 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 リリースで 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 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

標準コントロールを使用して、カメラを駆動します。可視化モードとインスタンスの外挿方法で調整します。インスタンスの視錐台カリングと LOD 選択が GPU に与える影響を観察し、カリングを無効にするか、LOD 選択をレベル 0 (最も詳細) に強制します。

# コントロール

|  |  |  |
| --- | --- | --- |
| 操作 | ゲームパッド | キーボード |
| カメラを移動する | 左サム スティック | WASD キーまたは方向キー |
| カメラを回転する | 右スティック | LMB 長押し + マウス |
| カメラのリセット | 右サムスティック (プッシュ) | - |
| インスタンス化モードの変更 | 確認が完了していないエイリアスの横には、 | Tab |
| レンダリング モードの変更 | X | Space |
| インスタンス化レベルの変更 | 右ショルダーとトリガー | +/- |
| Toggle Force LOD 0 | Y | 左 Shift キー |
| カリングの切り替え | B | 左コントロール |
| 終了 | ビュー ボタン | Esc キー |

# 実装上の注意

増幅シェーダー ステージは、メッシュ シェーダー パイプラインのメッシュ シェーダー ステージの前にあります。これはコンピューティングに似たシェーダー ステージで、その目的は、未処理のジオメトリ ワークロードを決定し、データのペイロード バッファーを設定し、ジオメトリを処理するために必要な数のメッシュ シェーダー スレッドグループを起動することです。

CPU 側のコードは、ユーザー入力によってインスタンスを生成し、インスタンス バッファーを GPU リソースにアップロードします。メッシュ LOD の配列は、シェーダー コードが LOD インデックスによって動的にインデックス付けするための記述子テーブルを設定します。増幅シェーダー (AS) は、1 つのシェーダー ウェーブのグループ サイズを持つように構成されているため、ウェーブの同期とメモリ バリアの必要性を回避できます。CPU は、スレッドごとに 1 つのインスタンスをスケジュールするのに十分な AS ウェーブをディスパッチします。

AS スレッドグループの各スレッドは、ビューの視錐台に対してカリングを行い、LOD 計算を実行する 1 つのインスタンスを処理します。LOD インスタンス数を取得し、ウェーブ組み込み関数を使用して再編成し、ディスパッチされたメッシュ シェーダーのペイロード データを生成します (図 2、左)。これは、LOD ごとのいくつかの異なるインスタンス数、オフセット、およびインスタンス インデックス リストで構成されます。レンダリングするメッシュの合計数によって、DispatchMesh 増幅シェーダー組み込み関数を使用してディスパッチするメッシュ シェーダー スレッド グループの数が決まります。

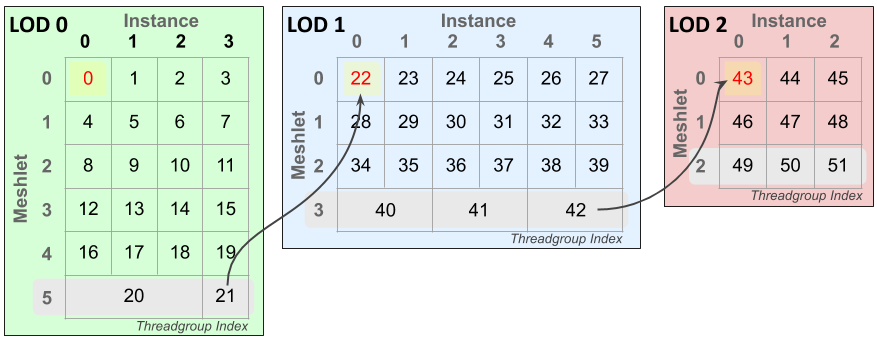


図 1:インスタンスとメッシュレットにまたがるメッシュシェーダー スレッド グループのインデックス作成レイアウトの例。このシナリオでは、4 つのインスタンスが LOD 0、6 つのインスタンスが LOD 1、および 3 つのインスタンス LOD 2 で、52 個のメッシュ シェーダー スレッド グループが必要です。各 LOD の最後のメッシュ (灰色で強調表示) は、複数のインスタンスを 1 つのメッシュ シェーダー スレッド グループにパックできます (頂点とプリミティブ カウントで許可されます)。黄色で強調表示されたセルは各 LOD の最初のスレッドグループ、赤色の値は各 LOD レベルのグローバル オフセットです。スレッド グループの LOD インデックスを決定するには、スレッド グループ インデックスから差し引く必要があります。矢印は、LOD 境界を越えたスレッド グループ ID の継続を示すものです。

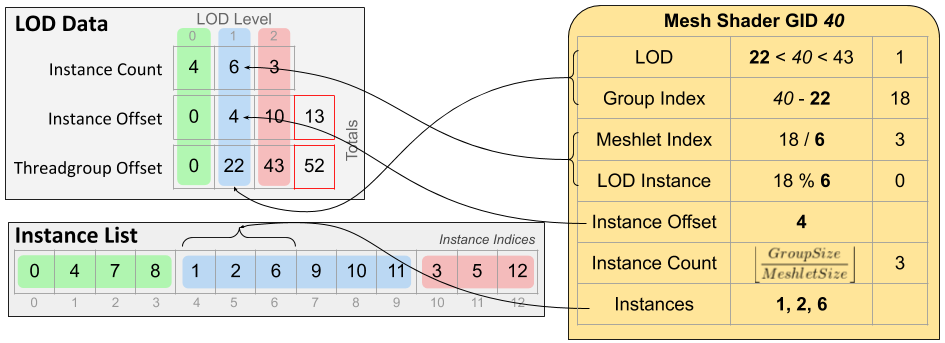


図 2:ペイロードデータ(左)は、各増幅シェーダースレッドグループからディスパッチされたメッシュシェーダースレッドグループ(右側のスレッドグループの例)に渡されます。 シェーダーは、スレッド グループのインデックスを使用して、処理対象の LOD、メッシュレット インデックス、インスタンス インデックスを計算できます。グループ サイズはスレッド グループ内のスレッド数で、メシュレット サイズはメッシュレットの頂点数とプリミティブ数の最大値です。

メッシュ シェーダーは単純なメッシュレット レンダリング シェーダーですが、ペイロード データを使用して LOD、メッシュレット、インスタンス インデックスを計算するコードが含まれています。メッシュレットと頂点のデータは、LOD インデックスによって SRV 配列にインデックスを付けることによって読み取られます。複数のインスタンスは、LOD レベルの最後のメッシュレットでスレッドグループ使用率が最大になる場合、1 つのスレッドグループで処理できます。

# 既知の問題

最適化 (-Od) を無効にすると、PC 上で InstancedLodMS.hlsl が破損します。これは、Windows SDK (10.0.19041) に同梱されているバージョンのシェーダー コンパイラ (dxc.exe) のバグが原因です。 この問題は、[GitHub](https://github.com/microsoft/DirectXShaderCompiler) で利用可能な最新リリースで修正されています。

# 更新履歴

2020 年 4 月 20 日 – サンプル作成。

2020 年 4 月 28 日 - メッシュ シェーダー パイプラインの作成に D3DX12 ヘルパーを使用するように更新

2021 年 9 月 2 日 - Windows SDK dx を使用して最適化を無効にすると PC が破損することにご注意ください。

# 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。