

Meshlet インスタンス作成のサンプル

*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。*

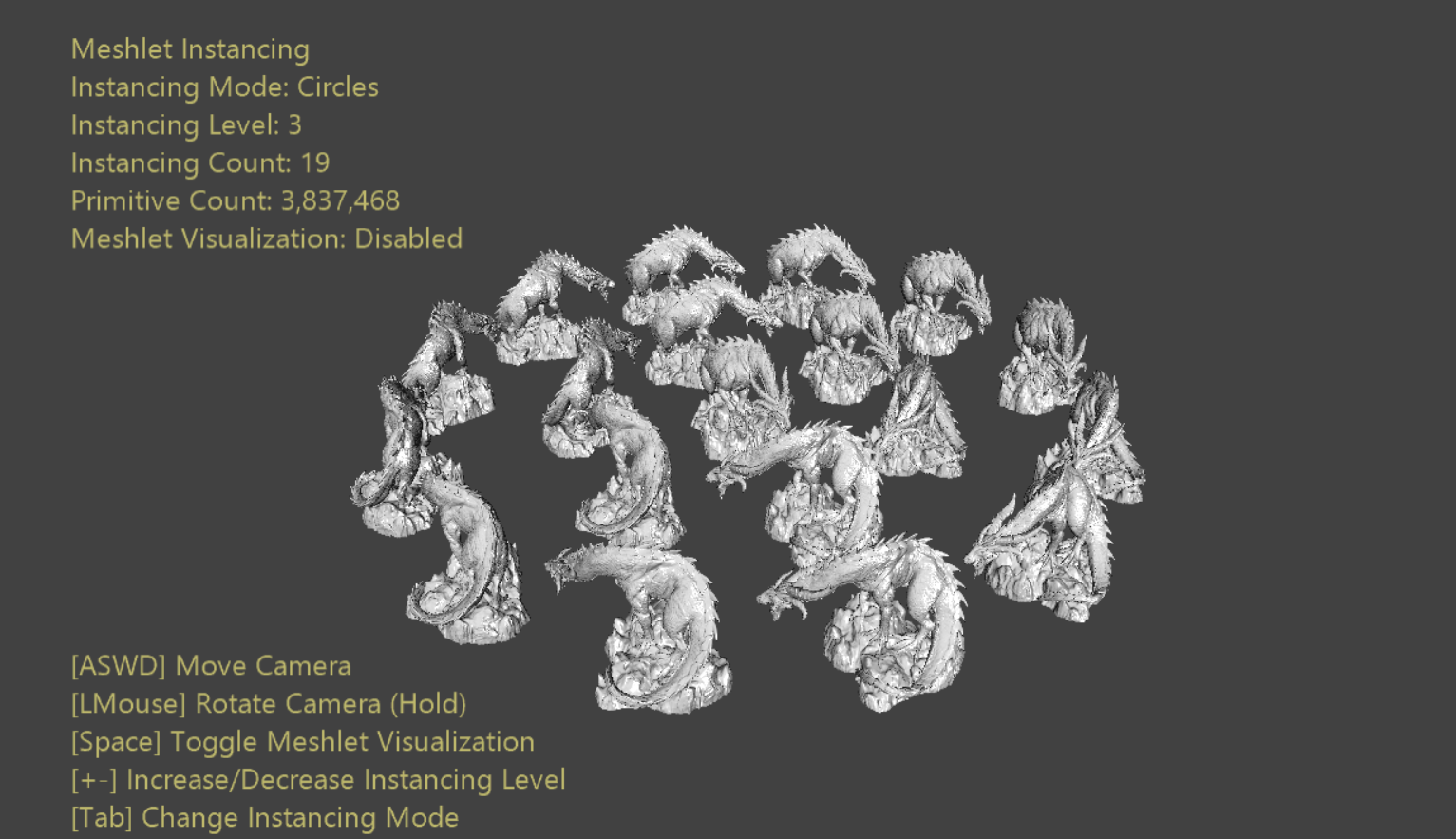
# 説明

メッシュ シェーダー パイプラインは、インデックス処理およびインスタンス作成機能を実行するために必要な入力アセンブラーを使用しません。このため、メッシュ シェーダー パイプラインでは、従来のグラフィック パイプラインのように、インスタンス作成のための高度なインターフェースが表示されず、開発者はシェーダー スレッド ID を使って自身のインスタンス作成ソリューションを実行する必要があります。

GPUは *Waves* と呼ばスレッドの固定サイズのチャンクにワークロードをディスパッチします。このサイズはアーキテクチャ固有ですが、ディスパッチできるスレッドの最小値です。入力アセンブラーが実行した機能の 1 つは、スレッドの使用率を最適化するために、これらの Wave をパックすることです*。Wave 使用率* は、実際のデータを処理している Wave スレッドと、ディスパッチされた Wave の合計数を比較した割合です。

メッシュ シェーダー パイプラインは入力アセンブラーの使用を省略しているため、作業付きの Wave をパックするというこの問題は、開発者に委ねられています。meshlet は、メッシュを使いやすい作業のチャンクに前処理し、Wave の使用を最適化するツールです。ただし、meshlet には完全にフルになるような厳しい制約はないため、次善のウェーブ使用率となります。特に、メッシュの最後の meshlet がいっぱいになることはほとんどありません (メッシュが塗りつぶしを実行される前にプリミティブを使い果たしたため)。 これは、メッシュ の meshlet 数がゼロに近づくにつれ、より大きな問題となります。これらの小さなメッシュのヘビーインスタンスは、樹木、髪、粒子などで一般的です。

このサンプルでは、塗りつぶされていない最後の meshlet のインスタンスの多くを 1 つのスレッドグループにパックして効率を高める一般的な手法を示します。これにより、ディスパッチされる停止状態のスレッドの数を最小限に抑え、Wave の使用率は 100% 近くに保ちます。



# サンプルの作成

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

このサンプルには、2 つのインスタンス レイアウト (同心円と立方体) を選ぶためのオプションが用意されています。これにより、異なるレートでの拡大縮小が可能になります。また、インスタンスの基礎となる meshlet 構造のフラット シェーディングと 視覚化を切り替えることもできます。

# コントロール​​

|  |  |  |
| --- | --- | --- |
| 動作 | ゲームパッド | キーボード |
| カメラを移動する | 左スティック | WASD または方向キー |
| カメラを回転させる | 右スティック | LMB を押しながらマウスの操作 |
| カメラをリセットする | 右スティック (押す) | - |
| インスタンス作成モードを変更する | A | Tab キー |
| Meshlet の視覚エフェクトを切り替える | X | Space キー |
| インスタンス作成レベルを上げる | 右ショルダー | + |
| インスタンス作成レベルを下げる | 右トリガー | - |
| 終了する | 表示ボタン | Escape キー |

# 実装に関する注意事項

メッシュシェーダーのインスタンス化は、作業を完了するために必要なシェーダーインスタンスをディスパッチするだけで、指定された Id を使用して、処理する正しい meshlet & インスタンスを決定するだけです。選択されたインデックス作成スキームにより、どのスレッドグループが作業を行うかが決定されます。

スレッドグループごとに 1 つの Meshlet とした場合、最も簡単な実装は MeshletCount \* InstanceCount スレッドグループのディスパッチです。この作業を完了するために必要なスレッドグループがあることを確認してください。これには、非常に分かりやすいインデックス作成スキームもあります。

* *MeshletIndex* = *GroupID.x* / *InstanceCount*
* *InstanceIndex* = *GroupID.x* & *InstanceCount*

これは、最初の *InstanceCount* スレッドグループが異なるインスタンスを除いた Meshlet *0*  のすべての処理を実施することを意味します。重要なのは最後の *InstanceCount* スレッドグループがMeshlet *(MeshletCount – 1)*  をすべて処理するということです。最初の *(MeshletCount – 1)* meshlet は「フル」 (頂点および/またはプリミティブの最大値に近い) で、最後の meshlet はそれよりも空きがあります。そのため、最後の *(MeshletCount – 1)* スレッドグループは最初の *(MeshletCount – 1) \* InstanceCount* スレッドグループよりも Wave 使用率が最適ではありません。十分に疎な場合、複数のインスタンスを 1 つのスレッドグループにパックできます。

実際には、スレッドグループのサイズが最大 meshlet サイズと同じであることを前提として、インスタンスを 1 つのスレッドグループにフィットさせることができます。したがって、最後の meshlet を処理するには、パックされたスレッドグループが必要です。これにより、シェーダーにこれらのスレッドグループの ALU がいくつか追加されますが、最適な Wave 効率を実現できます。

# 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 ステートメント](https://privacy.microsoft.com/en-us/privacystatement/)」を参照してください。