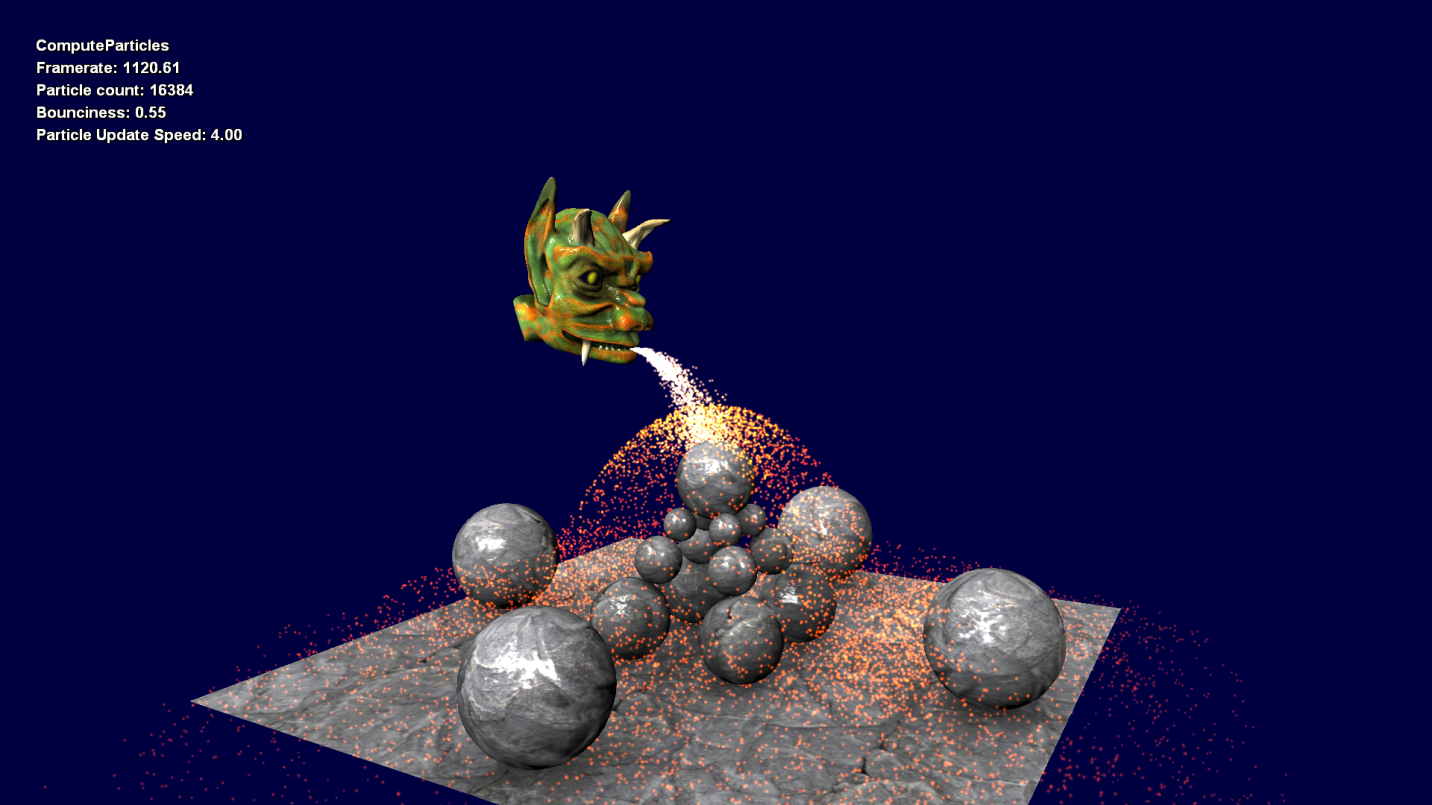


粒子の計算

*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) に対応しています。*

# 説明

このサンプルでは、計算シェーダーおよび追加バッファーを使用して、基本的な粒子シミュレーションを実行し、非常に多くの数の粒子を高いパフォーマンスでレンダリングする方法を示します。



# サンプルのビルド

Xbox One の devkit 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.XboxOne.x64 に設定します。

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。

*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

このサンプルでは、次のコントロールが使用されます。

|  |  |
| --- | --- |
| 操作 | コントローラー |
| サンプルを終了する。 | 選択 |
| 粒子の弾力性を増やす/減らす | 右/左トリガー |
| カメラを回転させる | 左/右スティック |
| 粒子エミッタを移動する | 左/右スティック + 右ショルダー |
| 粒子のレンダリングを切り替える | A ボタン |
| 粒子の更新を切り替える | B ボタン |
| 粒子数を増やす/減らす | 方向パッド 上/下 |

# 実装に関する注意事項

このサンプルでは、D3D11 および計算シェーダーで使用できる、興味深いいくつかの奥義について説明します。このサンプルには、3 つの興味深い点があります。最初の 2 つは粒子の更新およびカリング方法に関連し、3 つ目は計算シェーダー ステージの結果を通常のレンダリング パイプラインで使用する方法に関連します。

1. **粒子シミュレーション**

AdvanceParticlesCS 計算シェーダーのシミュレーション フェーズには、主に 2 つのステップがあります。まず、粒子の位置、速度、および経過時間が UAV バッファーから読み取られ、ワールド空間でシミュレートされます。次に、単純化された環境内ジオメトリ (基面および球体) への衝突が総当たり手法で計算されます。新しい位置、速度、および経過時間が、同じ UAV バッファーに書き戻され、以前に読み取られたデータが上書きされます。

1. **粒子のカリングと書き込み**

粒子が視錐台内で見えるかどうかを検討する単純平面カリング アルゴリズムが個々の粒子に適用されます。粒子が見える場合、粒子の位置がレンダリング用の追加バッファーに追加されます。

追加バッファーを使用するには、ID3D12Device::CreateUnorderedAccessView(…) API を使用し、2 番目のリソースを ‘pCounterResource’ パラメーターとして指定して、バッファー リソースの順序指定されていないアクセス ビュー (UAV) を作成します。追加バッファーの現在のカウントを格納するために、カウンター リソースは少なくとも 4 バイトである必要があります (32 ビットの符号なし整数が 1 つ)。 また、ID3D12GraphicsCommandList::ClearUnorderedAccessViewUint(…) API を使用して、各フレームのカウントをクリアするために使用するカウンター リソースの UAV も作成します。作成されたら、AppendStructuredBuffer<…> として宣言される UAV シェーダー スロットにバッファーをバインドします。

AdvanceParticlesCS の計算シェーダーは、アクティブな粒子のインスタンスをシミュレートし、追加バッファーに追加します。完了したら、ID3D12GraphicsCommandList::CopyBufferRegion(…) API を使用して、カウンター リソースの粒子数を間接的な引数バッファー リソースにコピーします。これは、ID3D12GraphicsCommandList::ExecuteIndirect(…) API への入力値として使用できます。これにより、AdvanceParticlesCS の視錐台カリング テストで決定された可視の粒子だけを描画できます。

1. **レンダリング**

ID3D12GraphicsCommandList::ExecuteIndirect(…) API を使用して、粒子のレンダリングをディスパッチします。 ExecuteIndirect がディスパッチするコマンドの種類を指定するために、ID3D12Device::CreateCommandSignature(…) を使用して作成されたコマンド シグネチャが必要です。コマンドの種類によって、間接的な引数バッファーの内容の解釈方法が決まります。この例では、コマンドの種類は ‘Draw’ で、これは ID3D12GraphicsCommandList::DrawInstanced(…) API (4 つの 32 ビット符号なし整数) に相関します。VertexCountPerInstance を 4 にハードコーディングし、粒子数を各フレームの InstanceCount の位置にコピーします。

頂点の属性は、頂点シェーダーの定数参照テーブルとしてハードコードされています。頂点 ID (SV\_VertexID) を使用して、この参照テーブルにインデックスを付けて各頂点の属性にアクセスします。インスタンス ID (SV\_InstanceID) を使用して、粒子のインスタンス バッファーにインデックスを付けて各インスタンスのプロパティにアクセスします。

# 更新履歴

2019 年 3 月 – 従来の Xbox サンプル フレームワークから新しいテンプレートに移植。

# 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、サンプルの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 のプライバシーに関する声明の詳細については、「[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。