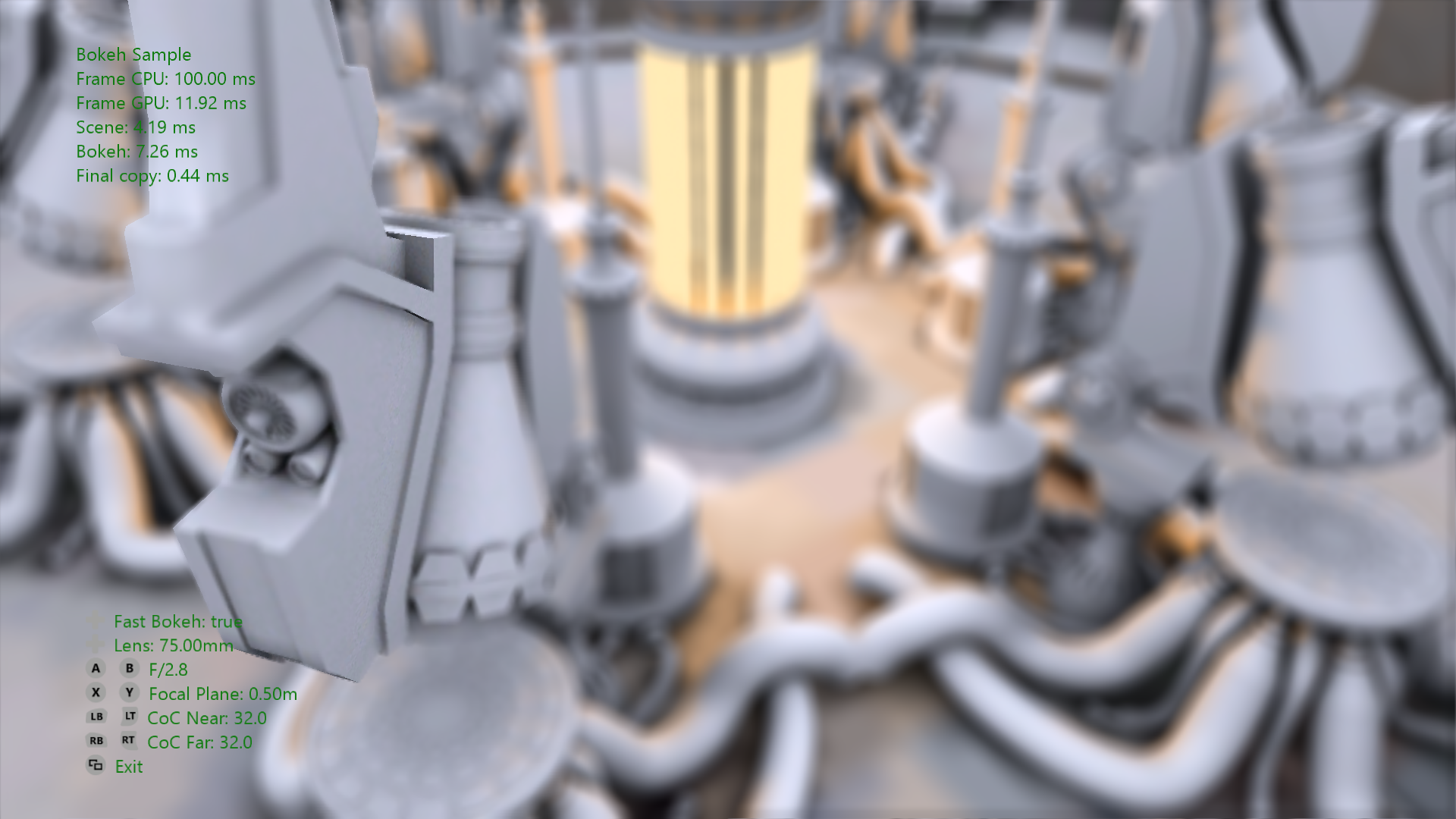


ボケのサンプル (DirectX 12)

*\* このサンプルは、2016 年 8 月の Xbox One XDK と互換性があります。*

# 概要

このサンプルは、ポイント スプライトのレンダリングを使用して被写界深度の効果を作成する方法を紹介します。



# サンプルの使用

このサンプルでは以下のコントロールを使用します。

## コントロール

|  |  |
| --- | --- |
| 作用 | ゲームパッド |
| カメラの回転 | 左サムスティックの X 軸 |
| カメラの仰角 | 左サムスティックの Y 軸 |
| カメラの距離 | 右サムスティックの Y 軸 |
| 焦点距離 | D-Pad 左/右 |
| 焦点面 | X ボタン/Y ボタン |
| 絞り値 | A ボタン/B ボタン |

|  |  |
| --- | --- |
| 作用 | ゲームパッド |
| 最大近距離 CoC サイズ | 左肩/左トリガー |
| 最大遠距離 CoC サイズ | 右肩/右トリガー |
| 高速ボケ シェーダーの切り替え | D-Pad 上 |
| サイクル プリセット ショット | D-Pad 下 |
| サンプルを終了 | 表示ボタン |

# 実装上の注意

カメラで撮影した写真の焦点が合っていない部分をぼかすことは、画像の特定の部分に注意を向けるための重要な効果です。画像の鮮明な領域は被写界深度 (DOF) と呼ばれ、画像の前後の領域がぼやけて見えます。

そのような重要なアート ツールがコンピューターで生成されたグラフィック イメージに役立つことは驚くことではありません。写真の DOF は、レンズの焦点距離、絞り、被写体までの距離によって決まり、概ね薄いレンズの方程式によって決まります。コンピューター グラフィックスでは、DOF に任意のパラメーターと式を使用できますが、写真の場合と同様にそれらを定義すると便利です。

レンズと他のイメージング システムを通して見たときに、世界のある点がフィルムまたはカメラのセンサー上に円形に投影されるため、DOF が存在します。これは常に円ですが、領域内のきちんと焦点が合っている小さな円であるか、または非常に大きなぼけた部分がある領域内で切り取られています。この円は Circle of Confusion (CoC) と呼ばれます。

一般に、DOF のリアルタイム グラフィックス実装では、ピクセルあたりの CoC を計算し、それに応じて画像をぼかします (オフライン DOF 実装は別の方法で行います)。そのようなリアルタイムのぼかしは、実物のように見えて役立てられるために、以下の制限を満たす必要があります。

1. 各点は、カメラの虹彩の画像として画面に投影されます。これは CoC のサイズです。その結果、外観は一般的にボケ DOF と呼ばれます。

2. 少なくとも焦点の合った画像が、カメラから遠いピクセルによって生じるぼやけた円およびカメラに近いピクセルによって生じるぼやけた円によってぼやけることがないように、ぼやけた点を前後にブレンドする必要があります。焦点の合った画像と焦点のずれた画像の上にブレンドします。この機能がないと、画像の焦点が合っている部分の縁の周りに色にじみが生じます。

3. 部分的なオクルージョン - 大きなアパーチャにより、光学システムで障害物の周囲を表示できます。これは、近距離の CoC が大きいほど、その下にあるオブジェクトが見えることを意味します。

4. 非常に大きなサイズの CoC があり、高速にする必要があります。

サンプルで紹介されているアルゴリズムは、3 以外の上記のすべての点を満たしています。部分的なオクルージョンを正しく処理するには、オクルージョンの背後のシーンをレンダリングするか (深度ピーリングと同様)、オクルージョンしたオブジェクトから部分的なオクルージョン領域に色を塗りつぶす必要があります。このサンプルでは、パフォーマンス上の理由でこの手順を実行するのではなく、基になるピンホール画像を撮影しています。これによって、一部にわずかな視覚的な不具合が生じることがあります。

## アルゴリズム:

この概念はシンプルです。入力のピクセルを取得し、その CoC を計算して、CoC のサイズであり、ソース ピクセルとして色付けされたポイント スプライトを出力します。それらをオクルージョン順に並べ替え、スプライトをすべて累計して、正しくぼかした画像を取得します。

このような直接的な方法で実装された場合、効果は問題がないように見えますが、最新のハードウェア上でリアルタイムで実行することはできません。以下は最適化されたアルゴリズムです。

1. ソース カラーと深度を RGBZ テクスチャに変換します。ソース テクスチャは、サイズが W\*2 x H\*2 です。

2. テクスチャを 1 回ダウンサンプリングし、4 色の平均と深度の最小値 4 を取得します。

3. WxH ダウンサンプリング テクスチャの場合は、6 つのビューポート (それぞれ 2 つのペア) でレンダー ターゲットを設定します。WxH、W/2xH/2、および W/4xH/4

4. W\*H/4 ポイント プリミティブをレンダリングします

1. 各ポイント プリミティブに対して、ダウンサンプリングされたソース テクスチャの 4 つの RGBZ 値を読み取ります。

2. 4 ピクセルがそれほど変わらない場合は 1 スプライトを出力し、それ以外の場合は 4 スプライトを出力します。

3. スプライトのサイズに応じて、対応するビューポートにルーティングされます。焦点の近い領域よりも近くのスプライトは、「近い」ビューポートに入り、遠くのスプライトは「遠い」ビューポートに入ります。

5. 6 つのビューポートのレンダリングが終了したら、それらを焦点が合っている画像と再結合し、焦点が合っているピクセルが遠いぼやけた領域をオーバーライドし、近いピクセルが焦点が合っているピクセルと遠いピクセルをブレンドします。

近いぼやけているピクセルのブレンドを向上させるために、画像の省エネが重要です。各ソース ピクセルには単位エネルギーがあるので、それをポイント スプライトでスプラッティングすると、エネルギーがスプライトの領域に分散されます。さまざまな虹彩テクスチャとラスター化の規則があるため、すべてのサイズとスプライトの原点について、閉じたフォームで重みを計算することは不可能です。サンプルでは、虹彩スプライトをさまざまなサイズでレンダリングし、その結果の重みを計算してから、正規化係数を計算します。

# 更新履歴

2015: オリジナル サンプルのオーサリング

2018 年 11 月: 新しいサンプル テンプレート用に書き直されました

# プライバシーステートメント

サンプルをコンパイルして実行すると、サンプルの実行可能ファイルの名前がMicrosoftに送信され、サンプルの使用状況の追跡に役立ちます。このデータ収集を無効にするには、Main.cppの「Sample Usage Telemetry」というラベルの付いたコードブロックを削除します。

マイクロソフトのプライバシーポリシー全般に関する詳細については、 [Microsoftのプライバシーステートメント](https://privacy.microsoft.com/ja-jp/privacystatement/)をご参照ください。