

単純な DMA 圧縮解除のサンプル

*\* このサンプルには、2017 年 6 月の Xbox One XDK との互換性があります*

# 概要



# このサンプルは、2 つの異なるシナリオでハードウェア LZ 圧縮および圧縮解除機能を使用する方法を紹介します。

# Xbox One GPU では、DEFLATE アルゴリズム (<http://www.rfc-base.org/rfc-1951.html>) と Jpeg デコードを実装した DMA エンジンに直接アクセスできます。 これらは、CPU を集中的に使用するタスクをオフロードするために使用できます。

# 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:

|  |  |
| --- | --- |
| 作用 | ゲームパッド |
| 読み込まれたテクスチャをを順番に切り替える | DPad 上/下 |
| ハードウェア圧縮/ソフトウェア圧縮を切り替える | Aボタン |
| ハードウェア圧縮解除/ソフトウェア圧縮解除を切り替える | B ボタン |
| 終了 | 表示ボタン |

# 実装上の注意

## シナリオ 1: 実行時デコード/エンコード

サンプルが読み込まれると、サンプルの Media\Textures\ フォルダにある DDS テクスチャ ファイルがすべて選択されます。 MediaList.txt ファイルを修正して再配置することで、さらにファイルを追加できます。

読み込まれると、Dma ハードウェアによる圧縮 (1 KB ウィンドウの LZ77) か、圧縮ライブラリ Zlib または Zopfli の参照を使用するかを変更できます。 後者のソフトウェア圧縮オプションを使用する場合、4 KB ウィンドウを使用する圧縮解除エンジンでサポートされる最適なオプションに制限されます。 これらのオプションを切り替えると、利用可能な圧縮オプションを簡単に比較できます。 ハードウェアとの互換性を維持するために、大きなファイルは 4 MB より少し小さなチャンクに断片化されます。

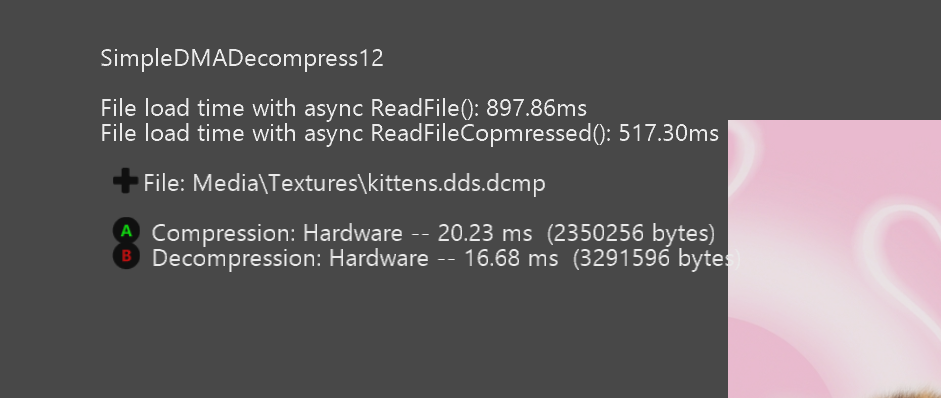
## シナリオ 2: ストリーミング DMA 圧縮解除

このシナリオではほとんどのタイトルと同様に、コンテンツ構築パイプラインの一部として圧縮されたディスクからストリーミングされているコンテンツを圧縮解除します。 このバリエーションでは、すべての圧縮解除の作業が、CPU サイクルを犠牲にすることなく、専用のハードウェアにオフロードされます。

これを達成するために、StreamingDmaCompressionLib プロジェクトでは、DMA ハードウェアへの非同期オフロードによって、効率的な I/O スループットを向上させ、タイトルのパッケージ サイズを小さくすることを可能にする、圧縮ブロックで構成される単純なファイル形式を定義します。 利便性のために、 XboxDmaCompression (PCのみ) を使用すると、ハードウェアでサポートされている最適な方法でファイル全体を圧縮できます。 あるいは、ライブラリを直接使用してストリーム レベルで圧縮し、圧縮されたコンテンツをより大きな複合ファイルに埋め込むこともできます。

生成されたファイル (または埋め込まれた圧縮ストリーム) は、ReadFileCompressed() を使用して読み取られます。これは、本質的に Win32 ReadFile() api との互換性がある API です。 これはオーバーラップした非同期I I/O を対象としており、バッファ済みの読み取りおよびバッファなしの読み取りがサポートされます。後者のほうがパフォーマンスが向上します。

このシステムのスループットの特性を説明するために、既定の MediaList.txt では、含まれているテクスチャの 20 個の圧縮コピーと 20 個の非圧縮コピーをレプリケートし、initialization\load パスで一度に 1 つのバッチを読み込みます。 各バッチの合計非圧縮バイトは 59.6MB で、非圧縮読み込み (66 MB/秒) と圧縮読み込み (115 MB/秒) の有効スループットは大幅に異なります。 これを DmaCompressionTool と組み合わせると、DMA 圧縮解除ハードウェアがタイトルに適しているかどうかを簡単に判断できます。 ライブラリは完全に非同期であるため、基になる Windows の overlapped\async I/O と同様に拡張され、実行中のリクエスト数が増えます。



ReadFileCompressed () api を使用する場合、主な違いは API が呼び出された時点では圧縮解除されたバッファのサイズがわからないことです。 この情報を事前に知ることは要求されません。そのためには他のブロッキング読み取りが必要になるためです。API でカスタム アロケーターの受け渡しがサポートされるか、既定のアロケーターを使用できます。割り当てが完了すると、ファイル ヘッダーが含まれているブロックが読み込まれます。

ReadFileCompressed が含まれるオーバーラップした I/O のその他の点は ReadFile と同じですが、いくつか例外があります。 ReadFileCompressed は、バッファなしのファイル位置をセクターに合わせる必要がない読み取りを処理し、事前と事後のバッファリングを内部的に処理します。 ただし、ReadFileCompressed では、ストレージ コントローラーからの dma の移動と、圧縮解除プログラムを介した dma の移動の両方を、CPU によってメモリがコピーされることなく実行できるように、読み取りを 4 バイトに揃える必要があります。 StreamingDmaCompressionLib を使用して、サイズの大きい複合ファイル内のストリームを圧縮する場合は、圧縮ストリームの最初のバイトがファイル内で 4 バイトに揃っていることを確認する必要があります。

# 既知の問題点

* このサンプルでは、中断/再開が正しく実装されていません。

# 更新履歴

このサンプルの最初のバージョンは 2012 年に作成されました。2018 年 9 月に修正されました。

# 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。