

高度な ESRAM 12 サンプル

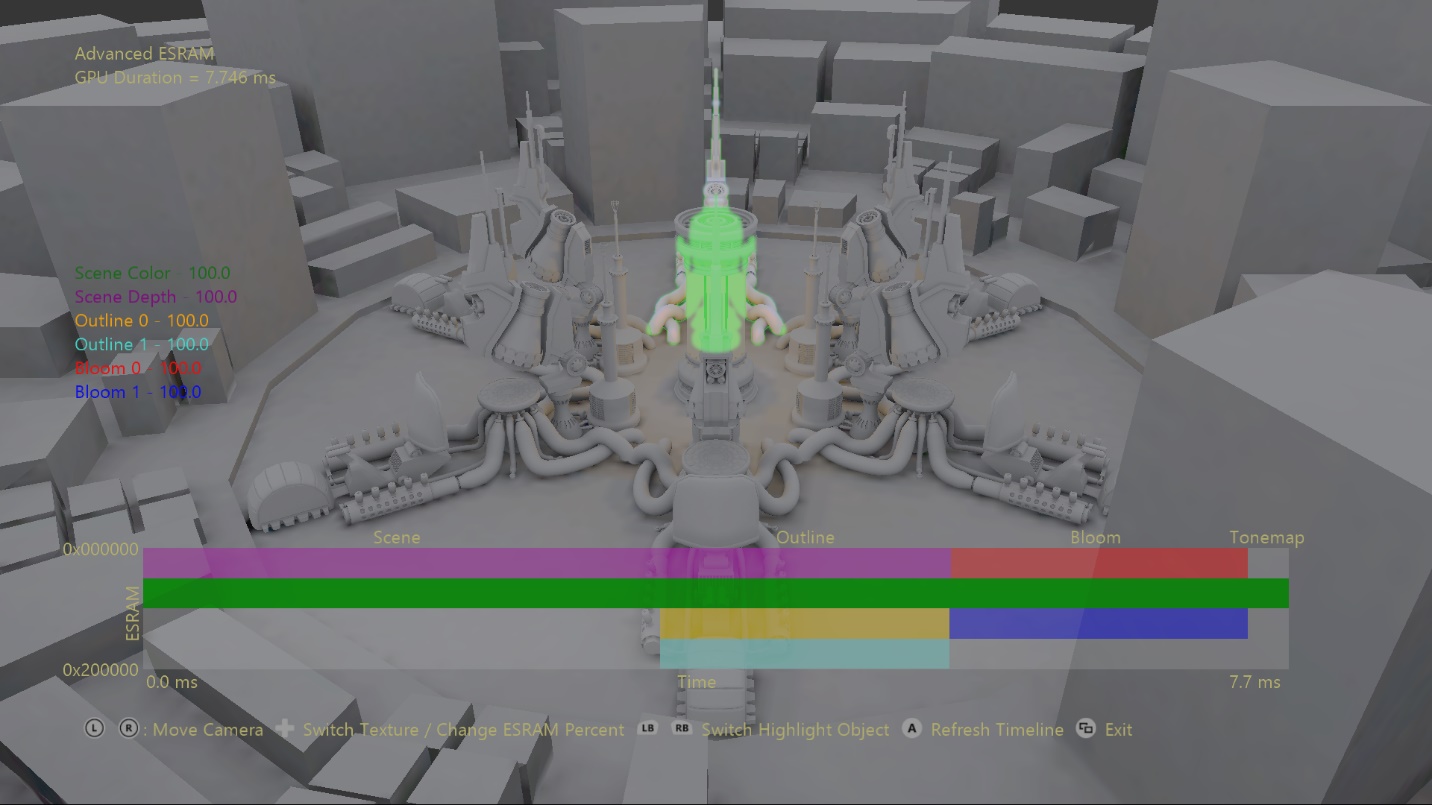
*\* このサンプルには、2016 年 10 月の Xbox One XDK との互換性があります。*

# 概要

このサンプルでは、D3D リソースのメモリを効果的にエイリアシングするための高度な DirectX 12.x メモリ機能の使用方法を紹介します。このサンプルの中心となる API は、ID3D12CommandQueue の「CopyPageMappingsX」および「CopyPageMappingsBatchX」です。これらの関数は、GPU タイムライン上で CPU ページ テーブル エントリを GPU にコピーする機能を提供し、完全仮想 D3D リソースをその場でメモリ ページにマッピングすることを可能にします。

この機能を使用して、サンプルでは 64 KB のページ単位で ESRAM と DRAM をステージングする一時リソース アロケーターを実装しています。これは、ESRAM の全機能を活用するために使用されるフレームの GPU メモリ使用量を最適にコンパクトに保ちます。インターフェースには、XG Memory ライブラリからの XGMemoryLayout のページ マッピング関数のポイントが反映されています。

注: Xbox One X には ESRAM がなく、代わりに高帯域幅の DRAM を利用しています。このコンソールでは、サンプルですべての ESRAM オプションとビジュアル化を無効にしてシーンをレンダリングします。



# サンプルの使用

サンプルの主な機能は、一時的なテクスチャ リソースが割り当てられている場所の操作を可能にすることです。フレームで使用されるリソースは、アウトライン、ブルームのシーンの色とシーンの深さのテクスチャです。リソースの ESRAM 構成は、左端にリソース メモリの割合として表示されます。ESRAM のレイアウトを紹介するビジュアル化は、各テクスチャの ESRAM と DRAM の構成に対する変更を即座に視覚的にフィードバックします。各リソースの ESRAM フットプリントは Y 軸に沿って表示され、ライフタイムは X 軸に沿って表示されます。時間軸に沿って使用される ＧＰＵ タイミングは、ボタンを押して更新できます。

## コントロール

|  |  |
| --- | --- |
| 作用 | ゲームパッド |
| カメラを原点に近づけたり遠ざけたりする | 左サムスティック 上/下 |
| 軌道カメラ | 右サムスティック |
| カメラのリセット | 右サムスティック (クリック) |
| サイクル遷移テクスチャ | D-Pad 左/右 |
| ESRAM の割合の変更 | D-Pad 上/下 |
| 強調表示されたオブジェクトのサイクル | 左/右のバンパー |
| タイムラインの更新 | Aボタン |
| 終了 | 表示ボタン |

# 実装上の注意

ESRAM と DRAM から割り当てられた 64 KB のメモリページ (ページ プール) のブロックをマップするために、大きな仮想アドレス領域 (VAS) が作成されます。ページ プールは、ID3D12Device の「RegisterPagePoolX」関数を使用して DirectX 12 に登録されます。使用されなくなった場合は、「UnregisterPagePoolX」を使用して登録解除する必要があります。このマッピングは、GPU ページ テーブルに直接コピーするために CPU ページ テーブル エントリをステージングするのに役立ちます。

ID3D12CommandQueue の「CopyPageMappingsX」関数または「CopyPageMappingsBatchX」関数を使用すると、これらのページ プール内のページ範囲を GPU タイムライン上の指定した GPU 仮想アドレスにマップできます。これにより、完全仮想 D3D リソースを 64 KB の物理ページに柔軟にマッピングできます。この機能により、リソース間のメモリのエイリアシングが簡単になり、ヒープ全体が向上します。

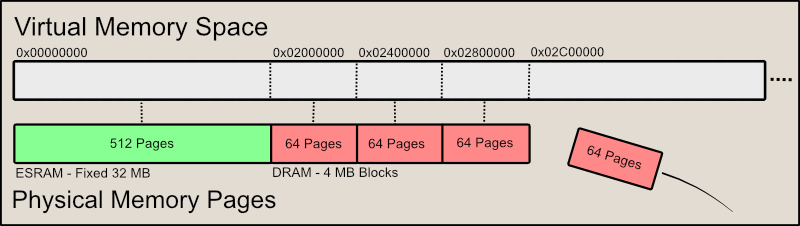


図 1: サンプルで使用されているメモリ マッピング パラダイム。ESRAM (使用可能な場合) は VAS の最初の 32 MB にマップされ、4 MB の DRAM ページ プールは必要に応じて追加されます。

ページ ブロックの作成と管理は、PageAllocator クラスによって実行されます。仮想アドレス範囲が指定され、アロケーターは必要に応じてページ プールをこの範囲に順次マッピングします。その後、「RegisterPagePoolX」を使用してページ プールが DirectX 12 に登録されます。アロケーター ページの使用状況は完全に追跡されます。最初から最後までページが割り当てられ、解放されるとページが置き換えられます。

TransientCache は、完全に仮想の D3D リソースを管理します。これらはオンデマンドで作成されますが、不要なオーバーヘッドによって共通リソースが再作成されるのを避けるためにキャッシュされます。仮想アドレス空間を割り当てるだけなので、これらのリソースをキャッシュすることによるメモリ オーバーヘッドは事実上ゼロです。各リソースは、フレームごとに 1 回だけ割り当てられます。

この TransientAllocator クラスではページ アロケーターと一時キャッシュを使用して、ユーザーへのリソース要求を処理します。リソースが要求されると、TransientCache からインスタンスを取得します。次に、PageAllocators から必要なページ数を割り当て、トークンを解析し、ページレベルの細分度で ESRAM と DRAM のどちらを使用するかを決定します。その後、後で「CopyPageMappingsBatchX」に提供される適切な構造、つまり D3D12XBOX\_PAGE\_MAPPING\_BATCH 構造および D3D12XBOX\_PAGE\_MAPPING\_RANGE 構造のベクトルが生成されます。

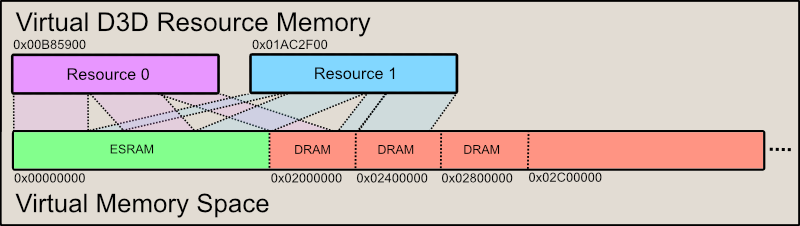


図 2: メモリの要件を満たすために、仮想 D3D リソースがページ プール内のページ範囲にマッピングされます。これらのマッピングは、CopyPageMappingsX および CopyPageMappingsBatchX の呼び出しの結果です。見やすくするために、このビジュアル化では 2 つのリソースだけが使用され、メモリのエイリアスもありません。ただし、メモリのエイリアシングは、この手法で期待される利点です。

メモリのエイリアシングによって、TransientAllocator では必要に応じてシェーダーとキャッシュのフラッシュを実行します。DirectX 12 では、フラッシュがリソース バリアの一部として挿入されます。このシステムでメモリのエイリアシングを回避したため、手動で独自のフラッシュを挿入する必要があります。TransientAllocator では、リソースに関連付けられているビューを調べて、どのシェーダー ステージとキャッシュをフラッシュするかを決定します。

最後に、割り当てられた一時リソースを使用するコマンド リストがそのコマンド キューに送信される前に、TransientAllocator で「Finalize」を呼び出してリソース マッピングを完了する必要があります。この時点で、「CopyPageMappingsBatchX」の呼び出しがコマンド キューに配置され、これによって後続のコマンド リストで使用されるリソースのメモリ マッピングが設定されます。

# 更新履歴

2018 年 8 月 6 日 - サンプルの作成。

# 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。