

FrontPanelDolphin のサンプル

# *\*このサンプルは Xbox One XDK （2017 年 6月） とで動作可能です。*

# 概要

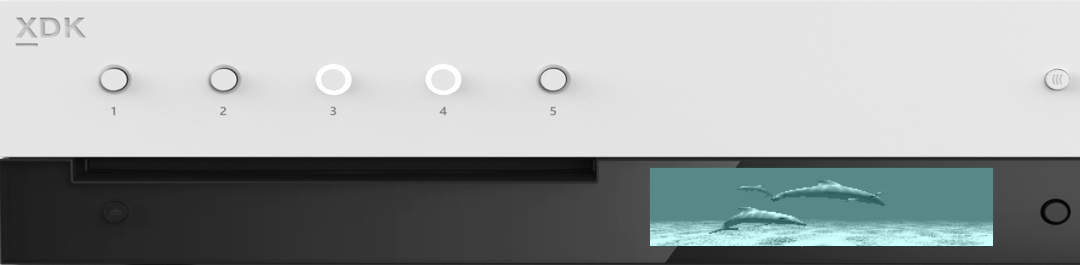
FrontPanelDolphin では、GPU を使用して FrontPanel にレンダリングする方法を紹介します。ほとんどのゲーム開発者は、GPU を使用して画面にレンダリングするためのコードを多数用意していると予想されます。したがって、このサンプルの目的は、フロント パネル ディスプレイを対象に、既存のコードを使用しやすくすることです。

以下にいくつかのユース ケースをご紹介します：

* 「ヘッドレス」開発キットのラボ設定でゲームを実行していて、画面に通常表示される内容をレンダリングするためにフロント パネルを使用します。一目で、ゲームが正常かどうかを判断できます。
* 多くのゲーム エンジンには既に「開発 HUD」があります。これは通常、製品版ではアクセスできないゲームの診断機能を有効にします。たとえば、シークレット コントローラー ボタンの組み合わせを実行すると、HUD が表示されます。HUD では、テスト中と開発中に役に立つ追加のオプションが提供されます。たとえば、モンスターを登場させたり、特定のレベルへのジャンプ、キャラクターを無敵にすることなどです。この HUD は通常 GPU によってレンダリングされるため、結果をフロント パネルにコピーする機能がある場合は、既存の HUD コードを再利用してフロント パネルに適応させることができます。フロント パネルに配置することにより、画面全体を有効に使うことができます。さらに、ゲームパッドを使用する代わりに、フロントパネルの DPAD とボタンを使用することもできます。

# サンプルの使用

FrontPanelDolphin のサンプルは、統合フロント パネルを備えた Xbox One X DevKit を対象としています。サンプルを起動すると、イルカのシーンがメイン ディスプレイとフロント パネルの LCD ディスプレイに同時にレンダリングされます。



ボタンごとに 1 つの LED

5 倍のプログラム可能ボタン

256 x 64 x 4 bpp の OLED ディスプレイ

方向パッド + 選択

## FrontPanelDolphin メイン ディスプレイ

## FrontPanelDolphin フロント パネル ディスプレイ



このサンプルでは、フロント パネルの DPAD 選択ボタン以外の入力を処理しません。選択ボタンを押すと、フロント パネル ディスプレイからバッファがキャプチャされ、結果が Title Scratch フォルダにある .dds ファイルに保存されます。

# 実装上の注意

このサンプルでは、FrontPanelRenderTarget と呼ばれるヘルパー クラスを使用します。名前が示すように、これはフロント パネル ディスプレイに適した画面外のレンダー ターゲットです。FrontPanelRenderTarget では、指定されたテクスチャをグレースケールに変換してからクワッドにレンダリングします。これは非常に単純な頂点シェーダーと非常に単純なピクセル シェーダーを使用して実現されます。頂点シェーダーではクワッドを生成し、ピクセル シェーダーでは指定されたテクスチャをサンプリングし、ドット積を使用して各ピクセルをグレースケールに変換します。レンダリング ステップは、GPUBlit() と呼ばれるメソッドで次のように実装されます。

// 指定されたシェーダー リソース ビューをソースとして使用してグレースケール イメージをレンダリングする

void GPUBlit(ID3D11DeviceContext \*context, ID3D11ShaderResourceView \*srcSRV);

指定された ID3D11ShaderResource ビューは、レンダリングするテクスチャから作成する必要があります。

GPUBlit() を呼び出した後、結果を CPU のバッファにコピーし、そのバッファをフロント パネル ディスプレイに表示する必要があります。これを簡単にするために、FrontPanelRendertTarget クラスでは次の 2 つのメソッドが提供されます。

// レンダー ターゲットをステージング テクスチャにコピーしてから CPU にコピーする

void CopyToBuffer(ID3D11Device \*device,

ID3D11DeviceContext \*context, ATG::BufferDesc &desc);

// レンダー ターゲットをステージング テクスチャにコピーしてから結果を CPU にコピーする

// その後、フォント パネル ディスプレイに表示する

// 呼び出されるたびに追加のバッファが割り当てられることに注意してください

void PresentToFrontPanel(ID3D11Device \*device, ID3D11DeviceContext \*context);

BufferDesc は、CPU バッファの幅と高さを追跡する構造体です。FrontPanelRenderTarget::CopyToBuffer では、メモリ内の任意のアドレスにコピーできます。必要なのは、バッファの大きさを記述する BufferDesc だけです。このサンプルでは、フロントパネル用のバッファを管理し、FrontPanelDisplay::GetBufferDescriptor() を使用してフロントパネル用の BufferDesc を取得する FrontPanelDisplay クラスを使用します。次に、FrontPanelRenderTarget からイメージをコピーする FrontPanelRenderTarget::CopyToBuffer() を呼び出します。最後に、FrontPanelDisplay::Present() を呼び出して、イメージを実際にフロント パネル ディスプレイに表示します。

FrontPanelRenderTarget::PresentToFrontPanel() メソッドでは、CPU へのコピーとバッファの表示の両方のステップが処理されることに注意してください。これは、まだ FrontPanelDisplay クラスを使用していない場合に便利です。FrontPanelRenderTarget::PresentToFrontPanel() の 1 つの欠点は、呼び出すたびにバッファを割り当てる必要があるということです。

フロント パネルにレンダリングするための Dolphin サンプルの適応は、実際には FrontPanelRenderTarget を使用すると非常に簡単です。わずかな変更を加えるだけで、最小限の適応が可能です。

**In Sample::Sample：**

// フロント パネルのレンダー ターゲットを作成する

m\_frontPanelRenderTarget = std::make\_unique<FrontPanelRenderTarget>();

//既定のフロント パネルを取得する

DX::ThrowIfFailed(GetDefaultXboxFrontPanel(m\_frontPanelControl.ReleaseAndGetAddressOf()));

// FrontPanelDisplay オブジェクトを初期化する

m\_frontPanelDisplay = std::make\_unique<FrontPanelDisplay>(m\_frontPanelControl.Get());

**In Sample::CreateDeviceDependentResources ：**

// GPU を介してフロント パネルにレンダリングするように設定する

ComPtr<IXboxFrontPanelControl> frontPanelControl;

DX::ThrowIfFailed(GetDefaultXboxFrontPanel(frontPanelControl.ReleaseAndGetAddressOf()));

// フロント パネルのレンダー ターゲット リソースを作成する

m\_frontPanelRenderTarget->CreateDeviceDependentResources(frontPanelControl.Get(),

device);

**In Sample::CreateWindowSizeDependentResources ：**

// メインのレンダー ターゲット用のシェーダー リソース ビューを作成する

auto device = m\_deviceResources->GetD3DDevice();

DX::ThrowIfFailed(device->CreateShaderResourceView(m\_deviceResources->GetRenderTarget(),

nullptr, m\_mainRenderTargetSRV.GetAddressOf()));

**In Sample::Render ：**

// フロント パネルのレンダー ターゲットに転送してからフロント パネルに表示する

m\_frontPanelRenderTarget->GPUBlit(m\_deviceResources->GetD3DDeviceContext(),

m\_mainRenderTargetSRV.Get());

auto fpDesc = m\_frontPanelDisplay->GetBufferDescriptor();

m\_frontPanelRenderTarget->CopyToBuffer(m\_deviceResources->GetD3DDevice(),

m\_deviceResources->GetD3DDeviceContext(), fpDesc);

m\_frontPanelDisplay->Present();

# 更新履歴

2017 年 4 月、サンプルの初回リリース

# 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。