CMake のサンプル

*このサンプルは、Microsoft ゲーム開発キットのプレビュー (2020 年 2 月) に対応しています。*

# 説明

この例では、[CMake](https://cmake.org/) クロスプラットフォーム ビルド システムを使用して、Microsoft ゲーム開発キットで実行可能ファイルをビルドします。



*このサンプルの主な目的は、Gaming.\*.x64 プラットフォーム用のビルドに必要なすべてのパスと設定を明確に文書化することです。GDK によってインストールされる MSBuild ルールに実装されている多くの機能を再現しています。*

# サンプルのビルド

Visual Studio 2019 を使用している場合、[新しいプロジェクト] ダイアログから [ローカル フォルダーを開く] を選択します。または [ファイル] > [開く] -> [フォルダー] メニュー コマンドで、デスクトップ、XboxOne、または Scarlett フォルダーを開きます。

* これを行うには、"C++ CMake tools for Windows" コンポーネントがインストールされている必要があります。

必要な場合、CMakeList.txt を編集して、正しい GDK エディションを参照してください。

set(XdkEditionTarget 200200)

CMake ツールを開くと、通常は自動的にキャッシュが生成されます。そうでない場合、CMakeList.txt を選択し、右ボタン メニューから [Generate Cache]\(キャッシュの生成\) を選択します。次に、[Build]\(ビルド\) > [Rebuild All]\(すべてリビルド\) を使用します。ビルドの成果物は、**"bin**" サブフォルダーにあります。

Visual Studio での CMake の詳細については、[Microsoft Docs](https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio) を参照してください。

*このサンプルでは target\_link\_directories を使用しているため、CMake 3.13 以降が必要です。Visual Studio 2017 (15.9 アップデート) に含まれているのはバージョン 3.12 であるため、この手順は Visual Studio 2019 用のものとなっています。もちろん、Visual Studio 統合を使用する代わりに、CMake ツールを直接使用することもできます。Visual Studio 2017 を使用している場合は、VC ランタイム DLL を見つけるために XboxOne および Scarlett CMakeList.txt のロジックを変更する必要があります。*

既定のセットアップには、Visual C++ を使用する **x64-Debug** および **x64-Release** の構成が含まれます。代わりに、clang/LLVM を使用するために、**x64-Clang-Debug** および **x64-Clang-Release** の構成を追加できます。

* これを行うには、"C++ Clang Compiler for Windows" コンポーネントがインストールされている必要があります。

# サンプルの使用

サンプルを展開するには、*Xbox Gaming コマンド プロンプト*のインスタンスを開き、サンプル ディレクトリに移動します。

cd CMakeExample\XboxOne\bin

### プッシュ展開

この 'ルーズ' レイアウトをプッシュ展開するには、次を実行します。

xbapp deploy Gaming.Xbox.XboxOne.x64

### PC からの実行

PC からこの 'ルーズ' レイアウトを実行するには、次を実行します。

xbpp launch Gaming.Xbox.XboxOne.x64\CMakeExampleXboxOne.exe

### パッケージ展開

パッケージを作成するには、次を実行します。

makepkg genmap /f chunks.xml /d Gaming.Xbox.XboxOne.x64

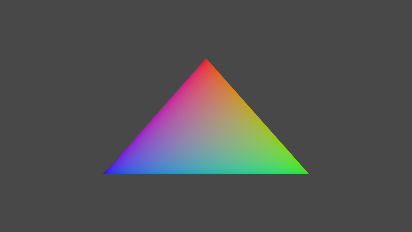
makepkg pack /f chunks.xml /lt /d Gaming.Xbox.XboxOne.x64 /pd .

デスクトップ パッケージの場合は、2 番目のコマンドラインに /pc も追加します。

その後、結果として得られたパッケージをコンソールにインストールします (.xvc ファイルの正確な名前は異なる場合があります)

xbapp install CMakeExampleXboxOne\_1.0.0.0\_neutral\_\_zjr0dfhgjwvde.xvc

サンプルが実行されると、デバイスおよびスワップチェーンが作成され、色付きの三角形が描画されます。コントロールやその他の動作はありません。



*他のバージョンをパッケージ化する場合、使用する具体的なコマンド ライン オプションについては、各 CMakeLIst.txt の最後のコメントを参照してください。*

# 実装に関する詳細情報

さまざまな Visual C++ スイッチの詳細については、以下のリンクを参照してください。

|  |  |
| --- | --- |
| /GR | <https://docs.microsoft.com/en-us/cpp/build/reference/gr-enable-run-time-type-information> |
| /GS  /RTC  /sdl    /DYNAMICBASE  /NXCOMPAT | <https://aka.ms/msvcsecurity> |
| /DEBUG:fastlink | <https://devblogs.microsoft.com/cppblog/faster-c-build-cycle-in-vs-15-with-debugfastlink/> |
| /EHsc | <https://devblogs.microsoft.com/cppblog/making-cpp-exception-handling-smaller-x64/> |
| /fp | <https://docs.microsoft.com/en-us/cpp/build/reference/fp-specify-floating-point-behavior>  <https://devblogs.microsoft.com/cppblog/game-performance-improvements-in-visual-studio-2019-version-16-2/> |
| /FS | <https://docs.microsoft.com/en-us/cpp/build/reference/fs-force-synchronous-pdb-writes> |
| /GL  /Gw  /LTCG | <https://devblogs.microsoft.com/cppblog/tag/link-time-code-generation/>  <https://devblogs.microsoft.com/cppblog/introducing-gw-compiler-switch/> |
| /Gy | <https://docs.microsoft.com/en-us/cpp/build/reference/gy-enable-function-level-linking> |
| /JMC | <https://devblogs.microsoft.com/cppblog/announcing-jmc-stepping-in-visual-studio/> |
| /permissive- | <https://devblogs.microsoft.com/cppblog/permissive-switch/> |
| /std:c++14 | <https://devblogs.microsoft.com/cppblog/standards-version-switches-in-the-compiler/> |

|  |  |
| --- | --- |
| /Yc  /Yu  /Fp  /FI | <https://docs.microsoft.com/en-us/cpp/build/creating-precompiled-header-files>  <https://devblogs.microsoft.com/cppblog/shared-pch-usage-sample-in-visual-studio/> |
| /Zc:\_\_cplusplus | <https://devblogs.microsoft.com/cppblog/msvc-now-correctly-reports-__cplusplus/> |
| /Z7, /Zi, /ZI | <https://docs.microsoft.com/en-us/cpp/build/reference/z7-zi-zi-debug-information-format> |

[/Gm](https://docs.microsoft.com/en-us/cpp/build/reference/gm-enable-minimal-rebuild) (最小リビルド) は非推奨であり、これをまだ使用しているプロジェクトから削除する必要があることに注意してください。

# 追加情報

このサンプルの CMake プロジェクトは、インストールしないビルド (BWOI) を使用するためのオプトインのビルド オプションをサポートしています。この機能を有効にすると、*BWOIExample* の extractgdk.cmd スクリプトによって作成された、抽出された Microsoft GDK を指す ExtractedFolder 環境変数が必要になります。CMake プロジェクトでは、Gaming.\*.x64 MSBuild プラットフォームを使用しないため、vctargets.cmd スクリプトの結果は必要ありません。

詳細については、**BWOIExample** を参照してください。

# バージョン履歴

2019 年 11 月 15 日 – 初期バージョン

2020 年 1 月 24 日 – HLSL シェーダーの使用をサンプルに追加

2020 年 2 月 5 日 – BWOI をオプションでサポートするように更新

2020 年 4 月 1 日 – CMake 3.16 以降を使用するときの pch サポートの更新