Cmake Xbox コンソール アプリ

*このサンプルは Microsoft Game Development Kit (2020 年 6 月) と互換性があります*

# 説明

これは、[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 GDK を使用する Xbox ハードウェアで実行できる「Win32 コンソール」アプリケーションを生成する例です。これは、“printf”形式の出力を使用したグラフィックス以外の開発者ユニットのテストに適しています。



*標準の Microsoft GDK アプリケーションのビルドに CMake を使用する方法についての詳細を検索する場合は、「****CMakeExample*** *と* ***CMakeGDKExample****」を参照してください。*

# サンプルのビルド (Visual Studio)

Visual Studio 2019 を使用して、[新しいプロジェクト] ダイアログまたは “[ファイル] -> [開く]　-> [フォルダー…]” メニュー コマンドから “[ローカル フォルダーを開く…]” を選択し、サンプル フォルダーを開きます。

* ここでは、“C++ CMake tools for Windows” コンポーネント (Microsoft.VisualStudio.Component.VC.CMake.Project) がインストールされている必要があります。

必要に応じて、CMake **XdkEditionTarget** 変数 (CMakePresets.json または CMakeList.txt) を編集して、正しい GDK エディションが参照されていることを確認します。

CMake ツールは開くときに、自動的にキャッシュを生成する必要があります。それ以外の場合は、CMakeList.txt を選択し、右ボタン メニューから “[キャッシュの生成]” を選択します。次に、“[ビルド] -> [すべて再ビルド]” メニュー コマンドを使用します。ビルドされた製品は “**out**” サブフォルダーにあります。

Visual Studio 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 を使用するため、CMake 3.13 以降が必要です。Visual Studio 2017 (15.9 更新プログラム) にはバージョン 3.12 が含まれているため、この手順は、Visual Studio 2019 用です。もちろん、Visual Studio 統合に依存せずに、CMake ツールを直接使用できます。Visual Studio 2017 を使用している場合は、VC ランタイム DLL を検索するために 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、CMake のプリセットとして定義される **x64-Debug**、**x64-Release**、**x64-Clang-Debug**、**および x64-Clang-Release** 構成が含まれます。

* これには、“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。

*[F5] を押すと、それは開発 PC での実行を試みますが、リモート コンソールではこれが行われないため、失敗する可能性があるかも知れません、また、ないかも知れません。正常に実行するには、以下の手順に従ってプログラムをデプロイする必要があります。*

# サンプルのビルド (コマンド ライン)

*VS 2019 開発者コマンド プロンプト*を使用して、コマンド ラインから生成およびビルドすることもできます。

cd CMakeXboxConsoleApp

cmake -B out -DXdkEditionTarget=220300

cmake --build out

CMake プリセットも提供されます

cmake --list-presets

cmake --preset=x64-Debug

cmake --build out\build\x64-Debug

# サンプルの使用方法

サンプルをデプロイするには、*Xbox ゲーム コマンド プロンプト* インスタンスを開き、サンプル ディレクトリに変更します。

cd CMakeXboxConsoleApp\out\build\<config>

xbcp bin\Console\\*.exe xd:\

xbcp bin\Console\\*.dll xd:\

サンプルを実行するには。

xbrun /O D:\CMakeXboxConsoleApp.exe

このプログラムはシステム OS のコンテキストで実行されます。

代わりにゲーム OS のコンテキストで実行する場合は、同様の手続きが使用できます。まず、ターゲットのコンソールでゲーム OS タイトルを実行して開始します。適切な候補は、Visual Studio の [新しいプロジェクト] ダイアログを使用し、Microsoft GDK を使用して既定 の “Direct3D 12 Xbox Game プロジェクト” を作成することです。それをビルドおよびデプロイして、実行状態のままにしておきます。

次に、以下を使用します。

xbcp /x/title bin\Console\\*.exe xd:\

xbcp /x/title bin\Console\\*.dll xd:\

サンプルを実行するには。

xbrun /x/title /O D:\CMakeXboxConsoleApp.exe

これは、Game OS VM にプロセスを挿入して動作することに注意してください。現時点では、マルチプロセス ゲーム タイトルはサポートされていません、またグラフィックス、オーディオ、GameRuntime を含むいくつかのコンポーネントは、複数のプロセス シナリオでの使用に関して、テストまたはサポートもされていません。また、ホスティング タイトルを非常にシンプルに保ち、CPU リソースの使用を制限することをお勧めします。

# 実装の詳細

デスクトップ PC の場合、Win32 コンソール exe (つまり、/SUBSYSTEM:CONSOLE) の **CMakeLists.txt** は次のようになります。

cmake\_minimum\_required (VERSION 3.13)

project(CMakeExampleWindowsConsole LANGUAGES CXX)

option(BUILD\_USING\_LTCG "Enable Whole Program Optimization" ON)

set(CMAKE\_CXX\_STANDARD 14)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

add\_executable(${PROJECT\_NAME} Main.cpp)

target\_compile\_definitions(${PROJECT\_NAME} PRIVATE "$<$<CONFIG:DEBUG>:\_DEBUG>" "$<$<CONFIG:RELEASE>:NDEBUG>")

target\_compile\_definitions(${PROJECT\_NAME} PRIVATE \_CONSOLE \_UNICODE UNICODE)

# 警告レベル 4 を使用する

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS ${CMAKE\_CXX\_FLAGS})

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS\_DEBUG ${CMAKE\_CXX\_FLAGS\_DEBUG})

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS\_RELEASE ${CMAKE\_CXX\_FLAGS\_RELEASE})

# typeid または dynamic\_cast を使用していない場合は、RTTI を無効にしてバイナリ サイズを保存できます

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS ${CMAKE\_CXX\_FLAGS})

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS\_DEBUG ${CMAKE\_CXX\_FLAGS\_DEBUG})

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS\_RELEASE ${CMAKE\_CXX\_FLAGS\_RELEASE})

target\_compile\_options(${PROJECT\_NAME} PRIVATE /fp:fast /GS /Gy)

if(CMAKE\_CXX\_COMPILER\_ID MATCHES "MSVC")

target\_compile\_options(${PROJECT\_NAME} PRIVATE /permissive- /Zc:\_\_cplusplus)

if(CMAKE\_BUILD\_TYPE MATCHES "Debug")

elseif(BUILD\_USING\_LTCG MATCHES ON)

target\_compile\_options(${PROJECT\_NAME} PRIVATE /GL /Gw)

target\_link\_options(${PROJECT\_NAME} PRIVATE /IGNORE:4075 /LTCG)

endif()

endif()

Xbox ハードウェアのシステムとゲーム OS の場合は、別のリンク ライブラリセットを使用し、サポートされていないライブラリを確実に選択しないようにする必要があります。また、サポートされていない API を使用しないよう、適切な API の分割を有効にする必要があり、このサンプルでは、プラットフォーム ヘッダーとライブラリを使用して確実にビルドします。

また、Xbox ハードウェアで実行されているアプリケーションでは、必要な Microsoft Visual C++ランタイム DLL と、Debug 用にビルドされた場合には ucrtbased.lib も提供する必要があります。

このサンプルの Xbox “コンソール” CMake は、Xbox Series X|S または Xbox One ハードウェアで実行する EXE をビルドするように設定されています。コンソール アプリには Direct3D を使用できないため、プラットフォームでの API の大きな違いを回避し、両方のプラットフォームで同じ EXE を実行することが当然に必要です。また、これは、特定の XboxOne および Scarlett include/lib パスが CMakeLists.txt で設定されていないということでもあります。

必要に応じて、Xbox Series X|S ハードウェア専用の追加のコンパイラ CPU ターゲットを有効にすることができます。これを行うには、ビルド オプション OPTIMIZE\_FOR\_SCARLETTを ON に設定します。結果の EXE は Xbox Series X|S では以前と同じように実行されますが、Xbox One では実行できません。これを説明するために、サンプルでは、関連する CPUID チェックを実行する DirectXMath XMVerifyCPUSupport 関数を使用しています。

# 追加情報

この例で使用されているすべてのコンパイラおよびリンカー スイッチの詳細については、「**CMakeExample**」を参照してください。

このサンプルの CMake プロジェクトでは、Build With/Out Installing (BWOI) を使用するためのオプトイン ビルド オプションがサポートされています。有効にすると、*BWOIExamples の* extractgdk.cmd スクリプトによって作成された抽出済みの Microsoft GDK を指す ExtractedFolder 環境変数が必要になります。また、2020 年 5月以降の GDK 用に抽出された Windows 10 SDK (19041) を任意で持つこともできます。CMake プロジェクトは、Gaming.\*.x64 MSBuild プラットフォームを使用しないため、vctargets.cmd スクリプトの結果を必要としません。

このビルド オプションを有効にするには、CMakeSettings.json を使用して BUILD\_USING\_BWOI を True に設定します。または、コマンド ラインを使用してビルドする場合は、生成ステップに -DBUILD\_USING\_BWOI=True を追加します。

詳細については、「**BWOIExample**」を参照してください。

# 既知の問題

clang/LLVM ツールセットを使用する場合は、DirectXMath 3.14 を含む Windows 10 SDK (19041) を使用していることを確認してください。DirectXMath 3.13 以前では、XMVerifyCPUSupport 実装がそのツールセットに対して正しくビルドされません。詳細については、[「https://walbourn.github.io/directxmath-3.14/」](https://walbourn.github.io/directxmath-3.14/)を参照してください。

# バージョン履歴

|  |  |
| --- | --- |
| **2020 年 5 月** | 初期バージョン。 |
| **2020 年 6 月** | 2020 年 6 月の GDK FAL リリース用に更新されました。 |
| **2020 年 11 月** | CMake ファイルをクリーンアップし、\_CONSOLE 定義を追加しました。 |
| **2021 年 2 月** | マイナーな CMake のクリーンアップ。 |
| **2021 年 8 月** | ツールチェーン ファイルの機能強化。 |
| **2021 年 10 月** | BWOI の更新プログラム。 |
| **2022 年 1 月** | VS 2022 のサポートを追加しました。  CMake のクリーンアップと CMake プリセット ファイルの追加。 |