CMake 예제

*이 샘플은 Microsoft 게임 개발 키트(2020년 6월)와 호환됩니다.*

# 설명

[CMake](https://cmake.org/) 플랫폼 간 빌드 시스템을 사용하여 Ninja 생성기를 통해 Microsoft 게임 개발 키트로 실행 파일을 빌드하는 예제입니다.



*이 샘플의 주요 목적은 Gaming.\*.x64 플랫폼용으로 빌드하는 데 필요한 모든 경로와 설정을 명확하게 문서화하는 데 있습니다. 이렇게 하면 GDK에 의해 설치된 MSBuild 규칙에서 구현되는 대부분의 기능이 복제됩니다. Visual Studio 생성기를 통해 CMake 를 활용하는 다른 방법은* ***CMakeGDKExample*** *을 참조하세요.*

# 샘플 빌드(Visual Studio)

Visual Studio 2019 또는 2022를 사용하여 새 프로젝트 대화 상자 또는 “파일 -> 열기 -> 폴더...” 메뉴 명령에서 “로컬 폴더 열기…”를 선택하고 데스크톱, XboxOne 또는 Scarlett 폴더를 엽니다.

* 이렇게 하려면 “Windows 구성 요소용 C++ CMake 도구”(Microsoft.VisualStudio.Component.VC.CMake.Project)가 설치되어 있어야 합니다.

필요한 경우 **XdkEditionTarget** 변수(CMakePresets.json 또는 CMakeList.txt)를 편집하여 참조된 올바른 GDK 버전이 있는지 확인합니다.

CMake 도구를 열면 캐시가 자동으로 생성되어야 합니다. 그렇지 않으면 CMakeList.txt를 선택하고 오른쪽 단추 메뉴에서 “캐시 생성”을 선택합니다. 그런 다음 “빌드 -> 모두 다시 빌드” 메뉴 명령을 사용합니다. 빌드 제품은 “**out**” 하위 폴더에 있습니다.

Visual Studio의 CMake에 대한 자세한 내용은 [Microsoft Docs](https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio)를 참조하세요.

*이 샘플에서는 CMake 3.13 이상이 필요하도록 target\_link\_directories를 사용합니다. Visual Studio 2017(15.9 업데이트)에는 버전 3.12가 포함되어 있으므로 Visual Studio 2019에 대한 지침이 제공됩니다. 물론 Visual Studio 통합에 의존하는 대신 CMake 도구를 직접 사용할 수 있습니다. Visual Studio 2017을 사용하는 경우 VC 런타임 DLL을 찾기 위해 XboxOne 및 Scarlett CMakeList.txt에서 논리를 수정해야 합니다.*

기본 설정에는 대신 clang/LLVM을 사용하는 **x64-Debug**, **x64-Release**, **x64-Clang-Debug** 및 **x64-Clang-Release** 구성이 포함됩니다.

* 이렇게 하려면 “Windows 구성 요소용 C++ Clang 컴파일러”가 설치되어 있어야 합니다.

*Xbox One 또는 Xbox Series X|S 프로젝트에 대해 F5 키를 누르면 원격 콘솔이 아닌 개발 PC에서 실행하려고 시도하므로 실패합니다. 올바르게 실행하려면 아래 지침에 따라 프로그램을 배포해야 합니다.*

# 샘플 빌드(명령줄)

*VS 2019 또는 2022 개발자 명령 프롬프트*를 사용하여 명령줄에서 생성하고 빌드할 수도 있습니다.

cd CMakeExample\XboxOne\

cmake . -B out -DXdkEditionTarget=220300

cmake --build out

CMake 사전 설정도 있습니다(CMake 3.19에서 도입됨).

cmake --list-presets

cmake --preset=x64-Debug

cmake --build out\build\x64-Debug

# 샘플 사용

샘플을 배포하려면 *Xbox 게임 명령 프롬프트* 인스턴스를 열고 샘플 디렉터리로 변경합니다.

cd CMakeExample\XboxOne\out\build\<config>\bin

### 푸시 배포

푸시를 수행하려면 ‘느슨한’ 레이아웃을 배포합니다.

xbapp deploy Gaming.Xbox.XboxOne.x64

### Run-from-PC

PC에서 ‘느슨한’ 레이아웃을 실행하려면 다음을 수행합니다.

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

### 패키지된 배포

패키지를 만들려면 다음을 수행합니다.

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

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

데스크톱 패키징의 경우 두 번째 명령줄에 /pc를 추가합니다.

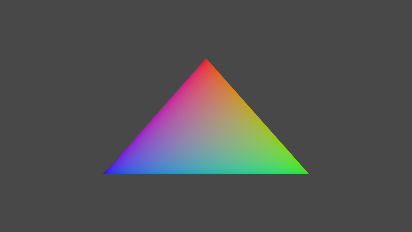
그런 다음 결과 패키지를 콘솔에 설치합니다(정확한 .xvc 파일 이름은 다름).

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

데스크톱의 경우 확장명은 “.msixvc”입니다(정확한 파일 이름은 다름).

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

샘플을 실행하면 디바이스 및 swapchain이 만들어지고 색이 지정된 삼각형이 그려집니다. 컨트롤이나 다른 동작은 없습니다.



*다른 버전을 패키징하려면 각 CMakeLIst.txt의 끝에 있는 주석에서 사용할 특정 명령줄 옵션을 참조하세요.*

# 구현 세부 정보

다양한 Visual C++ 스위치에 대한 자세한 내용은 아래 링크를 참조하세요.

|  |  |
| --- | --- |
| /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/> |
| /Zc:preprocessor | <https://devblogs.microsoft.com/cppblog/announcing-full-support-for-a-c-c-conformant-preprocessor-in-msvc/> |
| /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)(최소 다시 빌드)은 더 이상 사용되지 않으며 여전히 사용하는 프로젝트에서는 제거해야 합니다.

## 병렬 도구 집합

[Visual C++ 블로그](https://devblogs.microsoft.com/cppblog/side-by-side-minor-version-msvc-toolsets-in-visual-studio-2019/)에 따라 최신 버전의 Visual Studio IDE와 이전 버전의 컴파일러 도구 집합을 함께 사용할 수 있습니다. CMake의 경우 **CMakePresets.json**을 통해 이 작업을 수행합니다. 예를 들어 VS 2019(16.0) 버전의 컴파일러를 사용하려면 다음을 추가합니다.

"environment":

[

{

"ClearDevCommandPromptEnvVars": "false",

"VCToolsVersion": "14.20.27508"

}

],

# 추가 정보

이 샘플의 CMake 프로젝트는 BWOI(Build With/Out 설치)를 사용하는 옵트인 빌드 옵션을 지원합니다. 사용하도록 설정된 경우 *BWOIExample*의 extractgdk.cmd 스크립트에서 만든 추출된 Microsoft GDK를 가리키는 ExtractedFolder 환경 변수가 필요합니다. 필요에 따라 2020년 5월 GDK 이상에 대해 추출된 Windows 10 SDK(19041)를 가질 수도 있습니다. CMake 프로젝트는 Gaming.\*.x64 MSBuild 플랫폼을 사용하지 않으므로 vctargets.cmd 스크립트의 결과가 필요하지 않습니다.

이 빌드 옵션을 사용하도록 설정하려면 BUILD\_USING\_BWOI를 True로 설정합니다. 또는 명령줄을 사용하여 빌드하는 경우 생성 단계에 -DBUILD\_USING\_BWOI=True를 추가합니다.

자세한 내용은 **BWOIExample**을 참조하세요.

# 버전 기록

|  |  |
| --- | --- |
| **2019년 11월** | 초기 버전. |
| **2020년 2월** | 예제에 HLSL 셰이더 사용을 추가했습니다.  필요에 따라 BWOI를 지원하도록 업데이트되었습니다. |
| **2020년 4월** | CMake 3.16 이상을 사용할 경우 pch 지원으로 업데이트되었습니다. |
| **2020년 5월** | 2020년 5월 GDK를 지원하도록 업데이트되었습니다. |
| **2020년 6월** | 2020년 6월 GDK FAL 릴리스에 대해 업데이트되었습니다. |
| **2020년 8월** | 병렬 세부 정보로 업데이트되었습니다. |
| **2020년 11월** | Xbox 대상에 대해 xmem.lib 및 xg\_\*.lib가 추가되었습니다.  CMake 파일을 정리했습니다. |
| **2021년 2월** | CMake 주석에 대한 사소한 업데이트입니다. |
| **2021년 4월** | 데스크톱 대상의 링크 문제를 해결하기 위해 appnotify.lib를 추가했습니다.  LargeLogo.png를 추가했습니다. |
| **2021년 6월** | 일반 코드 정리입니다. |
| **2021년 8월** | 도구 체인 파일의 향상된 기능입니다. |
| **2021년 10월** | BWOI에 대한 업데이트입니다. |
| **2022년 1월** | VS 2022 지원이 추가되었습니다.  CMake를 정리하고 사전 설정 파일이 추가되었습니다. |