CMake 예제

*이 샘플은 Microsoft 게임 개발 키트 미리 보기(2020년 2월)와 호환됩니다.*

# 설명

다음은 Microsoft 게임 개발 키트를 사용하여 실행 파일을 만들기 위해 [CMake](https://cmake.org/) 플랫폼 간 빌드 시스템을 사용하는 예제입니다.



*이 샘플의 주요 목적은 Gaming.\*.x64 플랫폼용 빌드에 필요한 모든 경로와 설정을 명확하게 문서화하는 것입니다. 이렇게 하면 GDK에서 설치된 MSBuild 규칙에 구현된 많은 기능이 복제됩니다.*

# 샘플 빌드

Visual Studio 2019를 사용하여 새 프로젝트 대화 상자에서 "로컬 폴더 열기..."를 선택하거나 "파일 -> 열기 -> 폴더..." 메뉴 명령에서 Desktop, XboxOne 또는 Scarlett 폴더를 엽니다.

* 이 작업을 수행하려면 "C++ CMake tools for Windows" 구성 요소가 설치되어 있어야 합니다.

필요한 경우 CMakeList.txt를 편집하여 올바른 GDK 버전이 참조되는지 확인합니다.

set(XdkEditionTarget 200200)

CMake 도구를 열면 캐시를 자동으로 생성합니다. 그러지 않으면 CMakeList.txt를 선택하고 오른쪽 단추 메뉴에서 "캐시 생성"을 선택합니다. 그런 다음, "빌드 -> 모두 다시 빌드" 메뉴 명령을 사용합니다. 빌드 제품은 "**bin**" 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 샘플은 CMake 3.13 이상이 필요하므로 target\_link\_directories를 사용합니다. Visual Studio 2017(15.9 업데이트)에는 버전 3.12가 포함되어 있습니다. 이때문에 Visual Studio 2019용 지침이 제공되는 것입니다. 물론 Visual Studio 통합에 의존하는 대신 CMake 도구를 직접 사용할 수 있습니다. Visual Studio 2017을 사용하는 경우에는 VC 런타임 DLL을 찾기 위해 XboxOne 및 Scarlett CMakeList.txt에서 논리를 수정해야 합니다.*

기본 설정에는 Visual C++를 사용하는 **x64-Debug** 및 **x64-Release** 구성이 포함되어 있습니다**.** clang/LLVM을 대신 사용하기 위해 **x64-Clang-Debug** 및/또는 **x64-Clang-Release** 구성을 추가할 수 있습니다.

* 이 작업을 수행하려면 "C++ Clang Compiler for Windows" 구성 요소가 설치되어 있어야 합니다.

# 샘플 사용

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

cd CMakeExample\XboxOne\bin

### 푸시 배포

'느슨한' 레이아웃을 푸시 배포하려면 다음을 사용합니다.

xbapp deploy Gaming.Xbox.XboxOne.x64

### Run-from-PC

PC에서 '느슨한' 레이아웃을 실행하려면 다음을 사용합니다.

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

### 패키지된 배포

패키지를 만들려면 다음을 사용합니다.

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

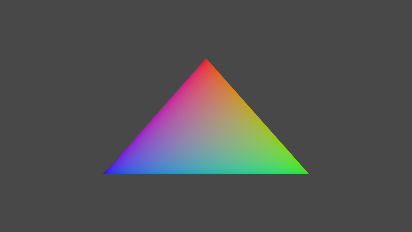
makepkg pack /f chunks.xml /lt /d Gaming.Xbox.XboxOne.x64 /pd .

데스크톱 패키지의 경우 두 번째 명령줄에 /pc도 추가합니다.

그런 다음, 결과 패키지를 콘솔에 설치합니다(정확한 .xvc 파일 이름은 다를 수 있음).

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

이 샘플을 실행하면 장치 및 Swapchain이 만들어지고 색상 삼각형이 그려집니다. 여기에는 컨트롤이나 다른 동작은 없습니다.



*다른 버전을 패키지하려면 각 CMakeLIst.txt 끝에 있는 주석에서 사용할 특정 명령줄 옵션을 확인하세요.*

# 구현 세부 정보

다양한 Visual C++ 스위치에 대한 자세한 내용은 다음 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/> |
| /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 않으므로 아직 사용하는 프로젝트에서 제거해야 합니다.

# 추가 정보

이 샘플의 CMake 프로젝트는 BWOI(설치 없는 빌드)를 사용하기 위한 옵트인(opt-in) 빌드 옵션을 지원합니다. 이 기능을 사용하도록 설정하면 *BWOIExample*의 extractgdk.cmd 스크립트를 사용하여 만든 추출된 Microsoft GDK를 가리키는 ExtractedFolder 환경 변수가 필요합니다. CMake 프로젝트는 Gaming.\*.x64 MSBuild 플랫폼을 사용하지 않으므로 vctargets.cmd 스크립트의 결과가 필요하지 않습니다.

자세한 내용은 **BWOIExample**을 참조하세요.

# 버전 기록

2019년 11월 15일 - 초기 버전

2020년 1월 24일 - 예제에 HLSL 셰이더 사용을 추가함

2020년 2월 5일 - 선택적으로 BWOI를 지원하도록 업데이트함

2020년 4월 1일 - CMake 3.16 이상을 사용하는 경우 pch를 지원하도록 업데이트함