BWOI(설치없이 빌드) 예제

*이 샘플은 Microsoft 게임 개발 키트와 호환 가능(2020년 6월)*

# 설명

개별 개발자는 일상적인 작업을 위해 컴퓨터에 컴파일러 도구 집합과 필수 SDK를 모두 설치해야 합니다. *Microsoft GDK(게임 개발 키트*)는 헤더 및 라이브러리 외에도 디버깅, MSBuild 플랫폼 정의 및 프로파일링 도구에 대한 Visual Studio 통합 기능을 제공합니다. 그러나 일일 빌드를 수행할 때 헤더 및 라이브러리에 대해 'xcopy 스타일' 배포를 사용할 수 있다면 빌드 서버 유지 관리 작업이 크게 단순화됩니다. 이 예제에서는 Microsoft GDK를 설치하지 않고도 **Gaming.\*.x64** 플랫폼을 사용하여 MSBuild 기반 프로젝트를 빌드하는 방법을 보여 줍니다. 또한 Windows 컨테이너를 사용하여 호스트 컴퓨터에 직접 Visual Studio 설치할 필요 없이 격리된 빌드 환경을 만드는 옵션도 제공합니다.

# 소프트웨어 설정

빌드 머신에는 일반적으로 정기적으로 유지 관리되는 이미지의 일부로 Visual Studio 도구 집합과 Windows SDK가 설치되어 있습니다. 이는 Azure DevOps "[Microsoft 호스팅](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/hosted?view=azure-devops)"에 해당하며 [자체 호스팅 Windows 에이전트](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops) 또는 기타 사용자 지정 빌드 컴퓨터를 설정하는 경우에 일반적입니다.

Microsoft GDK 프로젝트를 빌드하기 위해 [Visual Studio 2017](https://walbourn.github.io/vs-2017-15-9-update/)(v141 플랫폼 도구 세트 VC++ 프로젝트만 빌드 가능), [Visual Studio 2019](https://walbourn.github.io/visual-studio-2019/)(v141 및 v142 플랫폼 도구 세트 VC++ 프로젝트 빌드 가능) 또는 [Visual Studio 2022](https://walbourn.github.io/visual-studio-2022/)(v141, v142 및 v143 플랫폼 도구 세트 VC++ 프로젝트 빌드 가능)를 설정할 수 있습니다. 전체 Visual Studio 설치 또는 [Visual Studio Build Tools](https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2022)를 사용할 수도 있습니다. 다음 구성 요소를 설치해야 합니다.

**옵션 1: 전체 Visual Studio 설치**

|  |  |
| --- | --- |
| 워크로드 | 구성 요소 ID([명령줄 설치](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)의 경우) |
| C++를 사용한 게임 개발 | Microsoft.VisualStudio.Workload.NativeGame |
| C++를 사용한 데스크톱 개발  *필수 구성 요소, VS 2019/2022만 해당:* Windows 10 SDK(10.0.19041.0)  *선택적 구성 요소, VS 2019/2022만 해당:* MSVC v141 - VS 2017 C++ x64/x86 빌드 도구(v14.16)  *\* VS 2019/MSBuild 16.0 또는 VS 2022/MSBuild 17.0을 사용하여 v141 플랫폼 도구 집합 프로젝트를 빌드하는 경우에만 필요합니다.*  *선택적 구성 요소, VS 2022만 해당:* MSVC v142 - VS 2019 C++ x64/x86 빌드 도구(v14.29)  *\* VS 2022/MSBuild 17.0을 사용하여 v142 플랫폼 도구 집합 프로젝트를 빌드하는 경우에만 필요합니다.*  *선택적 구성 요소, VS 2019/2022만 해당:* Windowsdyd C++ Clang 도구(12.0.0 - x64/x86)  *\* Clang 도구 집합을 사용하여 빌드하는 경우에만 필요합니다.* | Microsoft.VisualStudio.Workload.NativeDesktop  *VS 2019/2022만 해당:* Microsoft.VisualStudio.Component.Windows10SDK.19041  *선택 사항, VS 2019/2022만 해당:* Microsoft.VisualStudio.Component.VC.v141.x86.x64  *선택 사항, VS 2022만 해당:* Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *선택 사항, VS 2019/2022만 해당:* Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

**옵션 2: Visual Studio Build Tools**

|  |  |
| --- | --- |
| 워크로드 | 구성 요소 ID([명령줄 설치](https://docs.microsoft.com/en-us/visualstudio/install/use-command-line-parameters-to-install-visual-studio)의 경우) |
| C++ 빌드 도구  *필수 구성 요소, VS 2019/2022만 해당:* Windows 10 SDK(10.0.19041.0)  *필수 구성 요소, VS 2019/2022만 해당:* MSVC v142 - VS 2019 C++ x64/x86 빌드 도구(최신) 또는 MSVC v143 - VS 2022 C++ x64/x86 빌드 도구(최신)  *\* VS 2017에는 해당하는 구성 요소가 자동으로 포함됩니다.*  *선택적 구성 요소, VS 2019/2022만 해당:* MSVC v141 - VS 2017 C++ x64/x86 빌드 도구(v14.16)  *\* VS 2019/MSBuild 16.0 또는 VS 2022/MSBuild 17.0을 사용하여 v141 플랫폼 도구 집합 프로젝트를 빌드하는 경우에만 필요합니다.*  *선택적 구성 요소, VS 2022만 해당:* MSVC v142 - VS 2019 C++ x64/x86 빌드 도구(v14.29)  *\* VS 2022/MSBuild 17.0을 사용하여 v142 플랫폼 도구 집합 프로젝트를 빌드하는 경우에만 필요합니다.*  *선택적 구성 요소, VS 2019/2022만 해당:* Windowsdyd C++ Clang 도구(12.0.0 - x64/x86)  *\* Clang 도구 집합을 사용하여 빌드하는 경우에만 필요합니다.* | Microsoft.VisualStudio.Workload.VCTools  *VS 2019/2022만 해당:* Microsoft.VisualStudio.Component.Windows10SDK.19041  *VS 2019/2022만 해당:* Microsoft.VisualStudio.Component.VC.Tools.x86.x64  *선택 사항, VS 2019/2022만 해당:* Microsoft.VisualStudio.Component.VC.v141.x86.x64  *선택 사항, VS 2022만 해당:* Microsoft.VisualStudio.ComponentGroup.VC.Tools.142.x86.x64  *선택 사항, VS 2019/2022만 해당:* Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Llvm.Clang |

|  |
| --- |
| BWOIExample 프로젝트는 기본적으로 v142 도구 집합을 사용하므로 VS 2019 또는 VS 2022가 필요합니다. VS 2019로 빌드하려면 MSVC v142 - VS 2019 C++ x64/x86 빌드 도구(최신) 구성 요소가 필요하고 VS 2022에는 MSVC v142 - VS 2019 C++ x64/x86 빌드 도구(v14.29) 구성 요소가 필요합니다. |

VS 2017(15.9 업데이트)의 경우 Windows 10 SDK(17763)가 기본적으로 설치됩니다. Windows 10 SDK(19041)를 가져오려면 [독립 실행 형태로](https://developer.microsoft.com/en-US/windows/downloads/windows-10-sdk) 설치해야 합니다.

# 빌드 환경 설정

소프트웨어 요구 사항이 설치되면 설치가 필요하지 않은 추출된 GDK를 설정할 수 있습니다. 이 작업은 다음 두 가지 방법으로 수행할 수 있습니다. 원하는 경우 Windows 10 SDK를 추출할 수도 있습니다.

## 방법 1: 추출된 GDK 다운로드

이 방법은 가장 간단한 옵션으로 권장됩니다.

1. [Xbox 개발자 다운로드](https://aka.ms/gdkdl)로 이동합니다.
2. 파일 형식으로 "Game Core"를 선택합니다.
3. 빌드/버전 메뉴에서 사용하려는 GDK 빌드에 대해 "빌드 시스템에 대해 추출된 Microsoft GDK"를 선택합니다.
4. ZIP을 다운로드하여 빌드 컴퓨터의 폴더에 추출합니다. MAX\_PATH 문제를 방지하려면 짧은 경로가 있는 위치를 선택하세요.

## 방법 2: 수동으로 GDK 추출

이 방법은 더 복잡하지만 별도의 다운로드가 필요하지는 않습니다. 표준 GDK 설치 관리자의 복사본이 필요합니다.

|  |
| --- |
| 이 메서드는 추출된 다운로드 옵션이 없는 공용 GDK와 함께 사용할 수 있습니다. |

1. **명령 프롬프트를 엽니다**(VS 또는 GDK에 대한 *개발자 명령 프롬프트*일 필요 없음).
2. BWOIExample 샘플 폴더로 **cd**합니다.
3. VS 2022, 2019 또는 2017에 대한 환경 변수를 설정하고 대상 버전 번호를 제공하세요. 추출된 GDK에 대한 사용자 지정 경로를 지정하는 경우 짧고 절대적인 따옴표가 없는 경로를 사용하여 MAX\_PATH 초과와 같은 문제를 방지하세요.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
4. 설치 관리자 이미지에서 GDK를 추출합니다.

extractgdk <path-to-gdk-installer>\Installers

|  |
| --- |
| MSIEXEC의 모든 사용은 전역 잠금을 사용하므로 추출 작업만 수행하더라도 다른 MSIEXEC 인스턴스가 동시에 실행 중인 경우(Windows 업데이트 또는 동일한 스크립트의 다른 인스턴스) 해당 작업이 실패합니다.  동일한 VM에서 실행되는 빌드 파이프라인의 경우 Global\\_MSIExecute 뮤텍스 및 사용자 고유의 전역 잠금을 사용하여 외부 잠금/잠금 해제 주기를 제공해야 합니다.  일반적으로 개발자 컴퓨터에서 MSI를 한 번만 추출하고 에이전트 액세스 가능 폴더에 결과를 복사하는 것이 더 쉽습니다. |

## 선택 사항: Windows 10 SDK 추출

원하는 경우 Windows 10 SDK를 추출하여 빌드 컴퓨터에서 올바른 버전을 항상 사용할 수 있도록 할 수도 있습니다. Visual Studio 설치로 Windows 10 SDK(19041)를 설치하는 경우 이 작업은 일반적으로 필요하지 않습니다.

이 프로세스에는 Windows 10 SDK 설치 관리자 이미지의 복사본이 필요합니다. 이를 가져오는 가장 쉬운 방법은 [Windows 개발자 센터](https://developer.microsoft.com/windows/downloads/windows-10-sdk)에서 Windows 10 SDK .ISO를 다운로드하는 것입니다(버전 10.0.19041.0이 필요함).

1. **명령 프롬프트**를 열고 BWOIExample 폴더에 **cd**합니다.
2. 환경 변수를 설정합니다. 추출된 GDK와 동일한 경로를 사용합니다.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. 설치 관리자 이미지에서 Windows 10 SDK를 추출합니다.  
     
   extractsdk <path-to-sdk-installer>\Installers

## VS 2019/2022만 해당: VCTargets 병합

VS 2019 및 2022 BWOI는 GDK의 플랫 파일 디렉터리를 설정하는 것 외에도 표준 Microsoft.Cpp MSBuild 규칙을 GDK의 MSBuild 규칙과 병합하는 결합된 VCTargets 폴더를 사용합니다. VS 2017의 경우 내부 변수를 통해 처리할 수 있지만 VS 2019 및 2022의 경우 강력한 솔루션은 추출된 GDK와 함께 병합된 폴더를 만드는 것입니다.

1. **명령 프롬프트**를 열고 BWOIExample 폴더에 **cd**합니다.
2. 환경 변수를 설정합니다. 다운로드하거나 수동으로 추출한 GDK의 경로를 제공합니다.  
     
   setenv vs2022 220300 [path-for-extracted-sdks]
3. 병합된 VC++ MSBuild 대상 디렉터리를 빌드하고 추출된 GDK 옆에 배치합니다.  
     
   vctargets

이러한 단계를 실행한 후 ExtractedFolder 환경 변수는 샘플을 빌드할 추출된 이식 가능한 GDK(및 선택적 Windows SDK 및 VCTargets 디렉터리)를 가리킵니다. 이 폴더는 다른 빌드 머신에도 복사할 수 있습니다.

# 샘플 빌드

빌드의 나머지 작업은 정상적으로 수행됩니다. 이 BWOI 예제는 Directory.Build.props 파일에 의해 구동됩니다. 대상 vcxproj 자체는 완전히 "stock"이며 Directory.Build.props 파일을 제거하면 GDK가 설치된 일반 개발자 시스템에서 예상한 대로 정확하게 작동합니다.

1. **명령 프롬프트를 엽니다**(VS 또는 GDK에 대한 개발자 명령 프롬프트일 필요는 없음).
2. BWOIExample 샘플 폴더로 **cd**합니다.
3. VS 2022, 2019 또는 VS 2017 및 GDK 버전 대상에 대해 **setenv**를 실행합니다.

setenv vs2022 220300 [path-for-extracted-sdks]

|  |
| --- |
| setenv를 실행하지 않으면 빌드가 Directory.Build.props에서 지정된 기본값으로 대체됩니다. 원하는 경우 파일에서 직접 수정할 수 있습니다. 또한 setenv를 사용하지 않는 경우 MSBuild가 경로에 있는지 확인해야 합니다. |

1. 명령줄에서 프로젝트를 빌드하세요.

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Desktop.x64

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.XboxOne.x64

msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64

|  |
| --- |
| VS 2019의 경우 v142 플랫폼 도구 집합 프로젝트만 지원하려고 하고 Microsoft.VisualStudio.Component.VC.v141.x86.x64를 설치하지 *않은* 경우 Directory.Build.Props를 편집하여 VCTargetsPath15의 설정을 제거해야 합니다. 마찬가지로 VS 2022의 경우 v143 플랫폼 도구 집합에 대한 지원만 설치한 경우 VCTargetsPath15 및 VCTargetsPath16을 모두 제거합니다. |

# Windows 컨테이너에서 빌드

또는 Docker를 사용하여 실행되는 Windows 컨테이너를 사용하여 격리되고 재현 가능한 빌드 환경을 만들 수 있습니다. 일관된 빌드 프로세스를 보장하기 위해 빌드 서버 또는 로컬 개발자 빌드에 사용할 수 있습니다. 이 샘플에는 Visual Studio 2022 빌드 도구를 사용하여 BWOI 빌드 환경을 설정하는 Dockerfile이 포함되어 있습니다.

|  |
| --- |
| 여기에 설명된 프로세스를 수행하려면 프로젝트에서 v142 도구 집합 이상을 사용해야 합니다.  Windows 컨테이너에 대한 자세한 내용은 [Windows 설명서의 컨테이너](https://docs.microsoft.com/en-us/virtualization/windowscontainers/)를 참조하세요. |

Dockerfile을 사용하려면 추출된 GDK를 여전히 제공해야 하며, 선택적으로 추출된 Windows SDK를 제공할 수 있습니다. 그러나 호스트 컴퓨터에 Visual Studio를 설치할 필요는 없습니다.

1. [여기](https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/set-up-environment)에 설명된 대로 Docker가 설치되어 있고 Windows 컨테이너를 사용하도록 설정되어 있는지 확인합니다.
2. BWOIExample 프로젝트와 추출된 SDK가 모두 포함된 부모 디렉터리로 Dockerfile을 이동합니다. 다음은 그 예시입니다.

<부모 디렉터리>

-> Dockerfile

-> BWOIExample

-> <프로젝트 및 스크립트 파일>

-> sdks

-> Microsoft GDK

-> <추출된 GDK 파일>

-> <선택적으로 추출된 Windows SDK>

|  |
| --- |
| Docker는 컨테이너를 빌드할 때 setenv.cmd, vctargets.cmd 및 추출된 SDK에 대한 액세스만 필요합니다. 원하는 경우 실제 프로젝트 소스를 다른 곳에 배치할 수 있습니다. |

1. Dockerfile이 포함된 디렉터리로 이동하여 다음을 실행합니다.  
     
   docker build -t gdkbwoi:latest -m 2GB --build-arg ExtractedSDKDir="sdks" --build-arg ScriptDir="BWOIExample" --build-arg GDKVer="220300" .

|  |
| --- |
| 컨테이너에서 추가 CPU 코어를 사용하도록 허용하려면 --cpus=*N* 플래그를 사용합니다. 추가 메모리를 사용하려면 -m 2GB 플래그의 값을 변경하세요. |

Docker는 컨테이너 생성, VS Build Tools 다운로드 및 설치, 필요한 \*.cmd 스크립트 및 추출된 SDK 복사, VCTargets 병합 프로세스를 자동화합니다.

1. 컨테이너가 빌드되면 다음을 사용하여 실행합니다.  
     
   cmd.exe 사용:

docker run --rm -v %cd%\BWOIExample:c:\Project -w c:\Project gdkbwoi msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64  
  
PowerShell 사용:

docker run --rm -v ${pwd}\BWOIExample:c:\Project -w c:\Project gdkbwoi msbuild BWOIExample.vcxproj /p:Configuration=Debug /p:Platform=Gaming.Xbox.Scarlett.x64  
  
이 명령은 컨테이너를 시작하고 그 안에 프로젝트 디렉터리를 탑재하고 지정된 매개 변수를 사용하여 msbuild를 실행합니다. 필요에 따라 구성 및 플랫폼을 변경할 수 있습니다. 다른 프로젝트를 빌드하려면 "%cd%\BWOIExample"을 프로젝트 위치로 변경합니다.

빌드가 완료되면 컨테이너가 종료됩니다. 프로젝트 디렉터리가 컨테이너에 마운트되었기 때문에 빌드 결과는 호스트 시스템의 프로젝트 디렉터리에 나타납니다.

# 추가 정보

Microsoft GDK 설명서에서는 MSBuild "BWOI" 속성에 대한 내용이 자세히 설명되어 있습니다.

Microsoft 게임 개발 키트 문서

-> 개발 및 도구

-> 설치하지 않고 Microsoft GDK(게임 개발 키트) 사용

<https://aka.ms/GDK_BWOI>

*CMakeExample* 샘플은 MSBuild 기반이 아닌 빌드 시스템을 사용하는 경우 모든 특정 규격 및 링커 스위치에 대한 세부 정보를 제공합니다. 해당 샘플은 이 샘플의 extractgdk.cmd 스크립트에 의해 생성된 동일한 BWOI 이미지를 사용하도록 설정하는 빌드 옵션(기본적으로 해제됨)을 지원합니다. CMake 예제는 Gaming.\*.x64 MSBuild 플랫폼을 사용하지 않으므로 vctargets.cmd 결과가 필요하지 않습니다.

자세한 내용은 **CMakeExample**를 참조하세요.

# 알려진 문제

DisableInstalledVCTargetsUse=true를 사용하고 프로젝트에 <MinimumVisualStudioVersion>16.0</MinimumVisualStudioVersion>가 포함되어 있는 경우 VS 2019의 일부 버전에서는 다음을 사용하여 빌드하지 못할 수 있습니다.

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Current\Bin\Microsoft.Common.CurrentVersion.targets(816,5): error : 'X.vcxproj' 프로젝트에 대해 BaseOutputPath/OutputPath 속성이 설정되지 않았습니다. 이 프로젝트에 유효한 구성 및 플랫폼 조합을 지정했는지 확인하세요. Configuration='Debug' Platform='Gaming.XBox.Scarlett.x64'. 이 메시지는 솔루션 파일 없이 프로젝트를 빌드하려고 하고 이 프로젝트에 존재하지 않는 기본이 아닌 구성 또는 플랫폼을 지정했기 때문에 표시될 수 있습니다.

해결 방법은 **Directory.Build.props**에 재정의를 추가하는 것입니다.

<PropertyGroup>

<ExtractedFolder Condition="'$(ExtractedFolder)'==''">C:\xtracted\</ExtractedFolder>

<ExtractedFolder Condition="!HasTrailingSlash('$(ExtractedFolder)')">$(ExtractedFolder)\</ExtractedFolder>

<\_AlternativeVCTargetsPath160>$(ExtractedFolder)VCTargets160\</\_AlternativeVCTargetsPath160>

<\_AlternativeVCTargetsPath150>$(ExtractedFolder)VCTargets150\</\_AlternativeVCTargetsPath150>

<!-- VS 버그에 대한 해결 방법 -->

<MinimumVisualStudioVersion>15.0</MinimumVisualStudioVersion>

</PropertyGroup>

이 문제는 Visual Studio 2019 버전 16.11에서 [해결](https://developercommunity.visualstudio.com/t/1695-or-later-fails-when-using-disableinstalledvct/1435971) 되었습니다.

# 버전 기록

|  |  |
| --- | --- |
| **2020년 2월** | 초기 버전. |
| **2020년 5월** | 선택적으로 추출된 Windows 10 SDK에 대해 업데이트됨 |
| **2020년 6월** | 2006 GDK FAL 릴리스에 대해 업데이트됨. |
| **2021년 4월** | LargeLogo.png 추가. |
| **2021년 6월** | 사용되지 않는 GDK에 대한 정보를 제거하고, 추가 설명을 추가하고, DisableInstalledVCTargetsUse 사용을 추가함.  일반 코드 정리. |
| **2021년 10월** | Dockerfile 및 Windows 컨테이너의 빌드에 관한 지침을 추가함. |
| **2022년 3월** | Visual Studio 2022를 지원하도록 업데이트되었습니다.  기본적으로 v142 도구 세트를 사용하도록 프로젝트 파일을 업데이트했습니다.  탑재된 디렉터리를 사용하도록 Dockerfile을 변경했습니다. |