CMake 示例

*此示例兼容于 Microsoft 游戏开发工具包(2020 年 6 月)*

# 说明

这是使用 [CMake](https://cmake.org/) 跨平台生成系统通过 Ninja 生成器使用 Microsoft 游戏开发工具包生成可执行文件的示例。



*此示例的主要用途是清楚地记录针对 Gaming.\*.x64 平台生成所需的所有路径和设置。这将复制 GDK 所安装 MSBuild 规则中实现的大部分功能。有关通过 Visual Studio 生成器使用 CMake 的替代方法，请参阅* ***CMakeGDKExample***

# 生成示例 (Visual Studio)

使用 Visual Studio 2019 或 2022 从“新建项目对话框”或使用“文件 -> 打开 -> 文件夹...”菜单命令选择“打开本地文件夹...”，然后打开桌面、XboxOne 或 Scarlett 文件夹。

* 这要求你安装“适用于 Windows 的 C++ CMake 工具”组件 (Microsoft.VisualStudio.Component.VC.CMake.Project)。

如果需要，请编辑 **XdkEditionTarget** 变量（在 CMakePresets.json 或 CMakeList.txt 中）以确保引用了正确的 GDK 版本。

CMake 工具应在打开时自动生成缓存。否则，请选择 CMakeList.txt 然后从右键菜单选择“生成缓存”。然后使用“生成 -> 全部重新生成”菜单命令。生成产品位于 "**Out**" 子文件夹中。

有关 Visual Studio 中的 CMake 的详细信息，请参阅 [Microsoft Docs](https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio)。

*此示例使用 target\_link\_directories，因此它需要 CMake 3.13 或更高版本。Visual Studio 2017（15.9 更新）包含版本 3.12，这就是为什么此说明适用于 Visual Studio 2019。当然可以直接使用 CMake 工具，而不依赖于 Visual Studio 集成。如果使用 Visual Studio 2017，则需要修改 XboxOne 和 Scarlett CMakeList.txt 中查找 VC 运行时 DLL 的逻辑。*

默认设置包括 **x64-Debug**、 **x64-Release**、 **x64-Clang-Debug** 和 **x64-Clang-Release** 配置，以改用 clang/LLVM。

* 这要求你安装“适用于 Windows 的 C++ Clang 编译器”组件。

*如果按 F5 查看 Xbox One或 Xbox Series X|S 项目，则尝试在开发电脑上而不是远程主机上运行，因此它将失败。你需要按照以下说明部署程序才能成功运行。*

# 生成示例（命令行）

还可以使用 *VS 2019 或 2022 开发人员命令提示符*从命令行生成和构建：

cd CMakeExample\XboxOne\

cmake . -B out -DXdkEditionTarget=220300

cmake --build out

还有 CMake 预设（在 CMake 3.19 中引入）：

cmake --list-presets

cmake --preset=x64-Debug

cmake --build out\build\x64-Debug

# 使用示例

若要部署示例，请打开 *Xbox 游戏命令提示*实例并更改为示例目录：

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

### 推送部署

若要执行推送部署 “松散”布局：

xbapp deploy Gaming.Xbox.XboxOne.x64

### 从电脑运行

如果要从电脑运行“松散”布局：

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

### 打包部署

如果要创建包：

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

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

对于桌面打包，还可将 /pc 添加到第二个命令行。

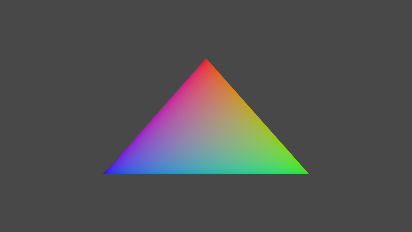
然后将生成的包安装到主机（确切的 .xvc 文件名会有所不同）：

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

对于桌面，扩展名为 “.msixvc” （确切的文件名会有所不同）：

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

运行时的示例创建设备和交换链，并绘制彩色三角形。它没有控件或其他行为。



*如果要打包其他版本，请参阅每个 CMakeLIst.txt 末尾的注释，了解要使用的特定命令行选项。*

# 实现详细信息

有关各种 Visual C++ 开关的详细信息，请参阅以下链接：

|  |  |
| --- | --- |
| /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> |

注意，[/Gm](https://docs.microsoft.com/en-us/cpp/build/reference/gm-enable-minimal-rebuild)（最小重建）已弃用，应从仍在使用它的项目中删除。

## 并行工具集

根据 [Visual C++ 博客](https://devblogs.microsoft.com/cppblog/side-by-side-minor-version-msvc-toolsets-in-visual-studio-2019/)，可以将较旧版本的编译器工具集与较新版本的 Visual Studio IDE 配合使用。对于 CMake，可通过 **CMakePresets.json** 执行此操作。例如，如果要使用版本为 VS 2019 (16.0) 编译器，请添加：

"environment":

[

{

"ClearDevCommandPromptEnvVars": "false",

"VCToolsVersion":"14.20.27508"

}

],

# 其他信息

此示例中的 CMake 项目支持选择加入生成选项，以使用生成并/并不安装 (BWOI)。如果启用，则需要 ExtractedFolder 环境变量，该变量指向由 *BWOIExample* 的 extractgdk.cmd 脚本创建的提取的 Microsoft GDK。还可以选择为 2020 年 5 月 GDK 或更高版本提取 Windows 10 SDK (19041)。CMake 项目不需要 vctargets.cmd 脚本的结果，因为它们不使用 Gaming.\*.x64 MSBuild 平台。

如果要启用此生成选项，请将 BUILD\_USING\_BWOI 设置为 True。或者，如果使用命令行生成的话，将 -DBUILD\_USING\_BWOI=True 添加到生成步骤中。

查看 **BWOIExample** 了解更多详情。

# 版本历史记录

|  |  |
| --- | --- |
| **2019 年 11 月** | 初始版本。 |
| **2020 年 2 月** | 向示例添加了 HLSL 着色器的使用。  已更新为可选支持 BWOI。 |
| **2020 年 4 月** | 使用 CMake 3.16 或更高版本时更新了 pch 支持。 |
| **2020 年 5 月** | 已更新以支持 2020 年 5 月版 GDK。 |
| **2020 年 6 月** | 已针对 2020 年 6 月 GDK FAL 版本进行了更新。 |
| **2020 年 8 月** | 更新了并行详细信息。 |
| **2020 年 11 月** | 向 Xbox 目标添加了 xmem.lib 和 xg\_\*.lib。  清理 CMake 文件。 |
| **2021 年 2 月** | CMake 注释的次要更新。 |
| **2021 年 4 月** | 添加 appnotify.lib 以解决桌面目标的链接问题。  Add LargeLogo.png。 |
| **2021 年 6 月** | 常规代码清理。 |
| **2021 年 8 月** | 工具链文件的改进。 |
| **2021 年 10 月** | BWOI 的更新。 |
| **2022 年 1 月** | 添加了 VS 2022 支持。  CMake 清理并添加了预设文件。 |