CMake 示例

*此示例与 Microsoft 游戏开发工具包预览版（2020 年 2 月）兼容*

# 说明

下面的示例是通过 [CMake](https://cmake.org/) 跨平台生成系统，使用 Microsoft 游戏开发工具包构建可执行文件。



该示例的主要目的是清楚地记录构建 *Gaming.\*.x64* 平台所需的所有路径和设置。*这将复制 GDK所安装 MSBuild规则中实现的许多功能。*

# 构建示例

使用 Visual Studio 2019，从“新建项目”对话框中选择“打开本地文件夹...”或“文件->打开->文件夹...”菜单命令，然后打开 Desktop、XboxOne 或 Scarlett 文件夹。

* 这要求安装“C++ CMake tools for Windows”组件。

如果需要，请编辑 CMakeList.txt，确保引用了正确的 GDK 版本：

set(XdkEditionTarget 200200)

CMake 工具应在打开后自动生成缓存。否则，选择 CMakeList.txt，然后从右键菜单中选择 “Generate Cache”。然后使用“ Build-> Rebuild All”菜单命令。构建产品位于“ **bin**”子文件夹中。

有关 Visual Studio 中 CMake 的更多信息，请参见 [Microsoft Docs](https://docs.microsoft.com/en-us/cpp/build/cmake-projects-in-visual-studio)。

*此示例使用了 target\_link\_directories，因此需要 CMake 3.13 或更高版本。Visual Studio 2017（15.9更新）包括版本3.12，这就是说明适用于 Visual Studio 2019 的原因。当然，您可以直接使用 CMake工具，不必依赖 Visual Studio集成。如果使用 Visual Studio 2017，则需要修改 XboxOne 和 Scarlett CMakeList.txt中的逻辑以查找 VC Runtime DLL。*

默认设置包括使用 Visual C++ 的 **x64-Debug** 和 **x64-Release** 配置。可添加 **x64-Clang-Debug** 和/或 **x64-Clang-Release** 配置，以改用 clang/LLVM。

* 这要求安装 “C++ Clang Compiler for Windows” 组件。

# 使用示例

若要部署示例，请打开 *Xbox 游戏命令提示符*实例，然后转到示例目录：

cd CMakeExample\XboxOne\bin

### Push deploy

若要推送部署“宽松”布局，请执行以下操作：

xbapp deploy Gaming.Xbox.XboxOne.x64

### Run-from-PC

若要从电脑运行“宽松”布局，请执行以下操作：

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

### Packaged deployment

创建程序包：

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

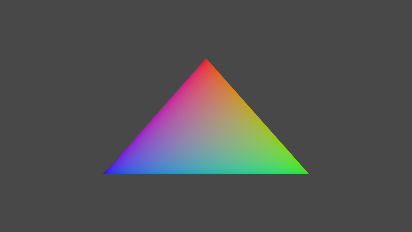
makepkg pack /f chunks.xml /lt /d Gaming.Xbox.XboxOne.x64 /pd .

对于桌面打包，还可将 /pc 添加至第二个命令行。

然后将生成的程序包安装到控制台（具体的 .xvc 文件名可能有所不同）

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

示例在运行时会创建设备和交换链，并绘制一个彩色三角形。它没有任何控件或其他行为。



*若要打包其他版本，请参阅每个 CMakeLIst.txt 末尾的注释，以获取要使用的特定命令行选项。*

# 实现详细信息

有关各种 Visual C ++ 开关的更多信息，请参见下面的链接：

|  |  |
| --- | --- |
| /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/> |
| /Z7, /Zi, /ZI | <https://docs.microsoft.com/en-us/cpp/build/reference/z7-zi-zi-debug-information-format> |

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

# 其他信息

此示例中的 CMake 项目支持选择使用的生成选项，以使用“有/无安装”（BWOI）构建。如果启用，则需要 ExtractedFolder 环境变量，此变量指向由 *BWOIExample 的* extractgdk.cmd 脚本所创建的已提取 Microsoft GDK。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。

有关更多详细信息，请参见 **BWOIExample**。

# 版本历史记录

2019 /11/15 – 初始版本

2020 / 1/ 24 – 向示例增加了使用 HLSL 着色器

2020 / 2/ 5 – 更新为可选择支持 BWOI

2020 /4/1 – 更新使用 CMake 3.16 或更高版本时提供 pch 支持