Cmake Xbox 主机应用

*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/) 跨平台生成系统生成“Win32 控制台”应用程序的示例，可使用 Microsoft GDK 在 Xbox 硬件上执行此应用程序。它适用于使用“printf”样式输出的非图形开发人员单元测试。



*如果你在查找使用 CMake 构建标准 Microsoft GDK 应用程序的详细信息, 请参阅* ***CMakeExample*** *and* ***CMakeGDKExample****.*

# 生成示例 (Visual Studio)

使用 Visual Studio 2019 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...”，然后打开示例文件夹：

* 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 CMakeList.txt 中查找 VC 运行时 DLL 的逻辑。*

默认设置包括定义为 CMake 预设的 **x64-Debug** 和 **x64-Release**，**x64-Clang-Debug** 和 **x64-Clang-Release** 配置。

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

*如果按 F5，则它会尝试在开发电脑而不是远程主机上运行，因此它可能会失败或可能不会失败。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 开发人员命令提示*通过命令行生成和构建：

cd CMakeXboxConsoleApp

cmake -B out -DXdkEditionTarget=220300

cmake --build out

还提供了 CMake 预设

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 CMakeXboxConsoleApp\out\build\<config>

xbcp bin\Console\\*.exe xd:\

xbcp bin\Console\\*.dll xd:\

若要运行示例：

xbrun /O D:\CMakeXboxConsoleApp.exe

程序将在 System OS 的上下文中运行。

如果希望改为在 Game OS 的上下文中运行，可以使用类似的过程。首先在目标主机上运行 Game OS 游戏。一个很好的候选项是使用 Visual Studio 的新项目对话框并使用 Microsoft GDK 创建默认的“Direct3D 12 Xbox 游戏项目”。生成并部署它，并使其保持运行状态。

然后使用：

xbcp /x/title bin\Console\\*.exe xd:\

xbcp /x/title bin\Console\\*.dll xd:\

若要运行示例：

xbrun /x/title /O D:\CMakeXboxConsoleApp.exe

请记住，此步骤可以通过将进程注入游戏 OS VM 来实现。目前不支持多进程游戏，且多个组件（包括图形、音频、GameRuntime）未经过测试，也不支持在多个进程方案中使用。让“承载”的游戏非常简单，并限制其对 CPU 资源的使用也是一个好主意。

# 实现详细信息

对于 PC 桌面设备，Win32 控制台 exe（即 /SUBSYSTEM：CONSOLE）的 **CMakeLists.txt** 如下所示：

cmake\_minimum\_required (版本 3.13)

project(CMakeExampleWindowsConsole LANGUAGES CXX)

option(BUILD\_USING\_LTCG "Enable Whole Program Optimization" ON)

set(CMAKE\_CXX\_STANDARD 14)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_EXTENSIONS OFF)

add\_executable(${PROJECT\_NAME} Main.cpp)

target\_compile\_definitions(${PROJECT\_NAME} PRIVATE "$<$<CONFIG:DEBUG>:\_DEBUG>" "$<$<CONFIG:RELEASE>:NDEBUG>")

target\_compile\_definitions(${PROJECT\_NAME} PRIVATE \_CONSOLE \_UNICODE UNICODE)

# 使用警告级别 4

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS ${CMAKE\_CXX\_FLAGS})

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS\_DEBUG ${CMAKE\_CXX\_FLAGS\_DEBUG})

string(REPLACE "/W3 " "/W4 " CMAKE\_CXX\_FLAGS\_RELEASE ${CMAKE\_CXX\_FLAGS\_RELEASE})

# 如果不使用 typeid 或 dynamic\_cast，我们可以禁用 RTTI 以节省二进制大小

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS ${CMAKE\_CXX\_FLAGS})

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS\_DEBUG ${CMAKE\_CXX\_FLAGS\_DEBUG})

string(REPLACE "/GR " "/GR- " CMAKE\_CXX\_FLAGS\_RELEASE ${CMAKE\_CXX\_FLAGS\_RELEASE})

target\_compile\_options(${PROJECT\_NAME} PRIVATE /fp:fast /GS /Gy)

if(CMAKE\_CXX\_COMPILER\_ID MATCHES "MSVC")

target\_compile\_options(${PROJECT\_NAME} PRIVATE /permissive- /Zc:\_\_cplusplus)

if(CMAKE\_BUILD\_TYPE MATCHES "Debug")

elseif(BUILD\_USING\_LTCG MATCHES ON)

target\_compile\_options(${PROJECT\_NAME} PRIVATE /GL /Gw)

target\_link\_options(${PROJECT\_NAME} PRIVATE /IGNORE:4075 /LTCG)

endif()

endif()

对于 Xbox 硬件上的系统和游戏操作系统，我们必须使用一组不同的链接库，并确保我们不会选取任何不受支持的库。我们还应启用适当的 API 分区以避免使用不受支持的 API，此示例可确保使用平台标头和库进行构建。

在 Xbox 硬件上运行的应用程序还需要提供所需的 Visual C++ 运行时 DLL，以及为调试生成的 ucrtbased.lib。

本示例中的 Xbox“主机”CMake 设置为生成 EXE，以便在 Xbox Series X|S 或 Xbox One 硬件上运行。由于无法将 Direct3D 用于主机应用，我们避免了平台中的主要 API 差异，并且可以合理地期望同一 EXE 在这两个平台上运行。这也由特定的 XboxOne 设置，并且未在 CMakeLists.txt 中设置 Scarlett include/lib 路径。

如果需要，可以专门为 Xbox Series X|S 硬件启用其他编译器 CPU 目标。这是通过将生成选项 OPTIMIZE\_FOR\_SCARLETT 设置为 ON 来完成的。生成的 EXE 将像以前一样在 Xbox Series X|S 上运行，但无法在 Xbox One 上运行。为了演示这一点，该示例使用 DirectXMath XMVerifyCPUSupport 函数来执行相关的 CPUID 检查。

# 其他信息

有关本示例中使用的所有编译器和链接器开关的详细信息，请参阅 **CMakeExample**。

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，因为它不使用 Gaming.\*.x64 MSBuild 平台。

若要启用此生成选项，请使用 CMakeSettings.json 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。

有关更多详细信息，请参阅 **BWOIExample**。

# 已知问题

如果使用 clang/LLVM 工具集，请确保使用的是包含 DirectXMath 3.14 的 Windows 10 SDK (19041)。在 DirectXMath 3.13 及更早版本中，XMVerifyCPUSupport 实现未为该工具集正确生成。有关详细信息，请参阅 <https://walbourn.github.io/directxmath-3.14/>。

# 版本历史记录

|  |  |
| --- | --- |
| **2020 年 5 月** | 初始版本。 |
| **2020 年 6 月** | 已针对 2020 年 6 月 GDK FAL 版本进行了更新。 |
| **2020 年 11 月** | 清理了 CMake 文件，添加了\_CONSOLE 定义。 |
| **2021 年 2 月** | 次要 CMake 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 CMake 预设文件。 |