CMake GDK 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 Visual Studio 生成器使用 Microsoft 游戏开发工具包生成可执行文件的示例。



*此示例演示如何使用 CMake 生成使用 Microsoft GDK 生成的 Gaming.\*.x64 平台 VC++ 项目文件。**有关通过 Ninja 生成器使用 CMake 的替代方法，请参阅* ***CMakeExample****。*

# 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 gxdk\_toolchain.cmake / gxdk\_xs\_toolchain.cmake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。

在组合框中选择要生成的平台（如果使用支持 [CMake 预设集成](https://devblogs.microsoft.com/cppblog/cmake-presets-integration-in-visual-studio-and-visual-studio-code/)的 VS 2019 （16.10） 或更高版本，将按如下所示进行填充）：

Graphical user interface, text, application

Description automatically generated

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)。

* 如果使用 Visual Studio 2022，请编辑 CMakePresets.json 以从以下位置更改此行：

"generator":"Visual Studio 16 2019",

to:

"generator":"Visual Studio 17 2022",

# 生成示例（命令行）

还可以使用 *VS x64 本机开发人员命令提示符*从命令行生成和构建：

cd CMakeGDKExample

cmake . -B out -DXdkEditionTarget=220300 -DCMAKE\_TOOLCHAIN\_FILE=.\gxdk\_toolchain.cmake

cmake --build out

还提供了 CMake 预设（需要 CMake 3.19 或更高版本）：

cmake --list-presets

cmake --preset=x64-XboxOne

cmake --build out\build\x64-XboxOne

# 使用示例 (Visual Studio)

从 Visual Studio 的另一个实例打开生成的 SLN/VCXPROJ：

CMakeGDKExample\out\build\x64-XboxOne\CMakeGDKExample.sln

如果使用的是 CMake 3.17 或更早版本，请先使用配置服务器检查 “*CMakeGDKExample* 项目的“部署”复选框。

然后使用 F5 进行部署/运行。

*如果从原始 CMakeLists.txt 上下文中按 F5，则它将无法启动，因为松散的布局未放置在 ‘bin’ 目录中。对于 Gaming.Xbox.\*.x64 配置，它还尝试在开发电脑上而不是远程主机上运行。*

# 使用示例（命令行）

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 CMakeGDKExample\out\build\x64-XboxOne\bin\Gaming.Xbox.XboxOne.x64

对于桌面，松散布局位于 bin\Gaming.Desktop.x64\Debug 中

### 推送部署

若要执行推送部署 “松散”布局：

xbapp deploy Layout\Image\Loose

### 从电脑运行

如果要从电脑运行“松散”布局：

xbapp launch Layout\Image\Loose\CMakeGDKExample.exe

### 打包部署

编辑 Layout\Image\Loose\Microsoft.Config 以添加 TargetDeviceFamily 组件 (“PC”、“Scarlett” 或 ”XboxOne”):

<ExecutableList>

<Executable Name="CMakeGDKExample.exe"

**TargetDeviceFamily="XboxOne"**

Id="Game" />

</ExecutableList>

如果要创建包：

makepkg genmap /f chunks.xml /d Layout\Image\Loose

makepkg pack /f chunks.xml /lt /d Layout\Image\Loose /pd .

然后将生成的包安装到主机（确切的 .xvc 文件名将会有所不同）

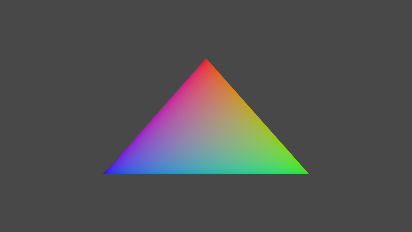
xbapp install CMakeGDKExample\_1.0.0.0\_neutral\_\_8wekyb3d8bbwe\_x.xvc

对于桌面打包：

makepkg genmap /f chunks.xml /d bin\Gaming.Desktop.x64\Debug

makepkg pack /pc /f chunks.xml /lt /d bin\Gaming.Desktop.x64\Debug /pd .

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。



*若要打包 Xbox Series X|S 和/或 Xbox 版本，请更改为平台和配置的正确目录。*

# 实现详细信息

**CMakeExample** 使用 “Ninja” 生成器，因此不使用 Microsoft GDK MSBuild 规则。此版本改用 “Visual Studio 16 2019 Win64” 生成器，它使用 Microsoft GDK MSBuild 指令。

CMake 生成使用作为命令行传递的工具链文件：

|  |  |
| --- | --- |
| Gaming.Desktop.x64 | -DCMAKE\_TOOLCHAIN\_FILE="grdk\_toolchain.cmake" |
| Gaming.Xbox.XboxOne.x64 | -DCMAKE\_TOOLCHAIN\_FILE="gxdk\_toolchain.cmake" |
| Gaming.Xbox.Scarlett.x64 | -DCMAKE\_TOOLCHAIN\_FILE="gxdk\_xs\_toolchain.cmake" |

这三者还使用自定义 MSBuild 属性文件 gdk\_build.props。

将 Gaming.\*.x64 MSBuild 规则与 GDK 配合使用可处理 MicrosoftGame.Config 本地化、将 CRT 文件放入布局等。

CMake 无法将 FXCCompile MSBuild 目标用于着色器，因此 CMakeLists.txt 将 DXC 作为自定义目标运行。工具链负责查找正确的着色器编译版本，这就是 gxdk\_toolchain.cmake 和 gxdk\_xs\_toolchain.cmake 需要 **XdkTargetEdition** 变量的原因。

若要通过生成的 CMake 支持*无安装生成* （BWOI），需要 (a) 显式设置 **GDK\_DXCTool** 以指向要生成的平台的正确 DXC.EXE; (b) 使用 **BWOIExample** 示例中详述的 Directory.Build.props 解决方案，因为 CMake 生成的 vcxproj 使用 Microsoft GDK 的 MSBuild 规则。在*生成* CMake 以及构建生成的 SLN/VCXPROJ 时，需要存在 Directory.Build.props文件，并且需要正确设置环境。

使用 BWOI 从命令行生成时，可以通过添加 -DGDK\_DXCTool=<path>（其中<path>采用 <path to GDK>\<edition number>\GXDK\bin\<XboxOne or Scarlett>\dxc.exe 的形式来指定 **GDK\_DXCTool**。例如：

-DGDK\_DXCTool="d:\xtrctd.sdks\BWOIExample\Microsoft GDK\210600\GXDK\bin\XboxOne\dxc.exe".

## 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 **CMakeSettings.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"

}

],

如果在不使用 Visual Studio 集成时直接使用 CMake 和 VS 生成器，也可以通过 **set\_property** 指定此项。

set\_property(TARGET ${PROJECT\_NAME} PROPERTY VS\_GLOBAL\_ClearDevCommandPromptEnvVars "false")

set\_property(TARGET ${PROJECT\_NAME} PROPERTY VS\_GLOBAL\_VCToolsVersion "14.20.27508")

# 版本历史记录

|  |  |
| --- | --- |
| **2020 年 4 月** | 此版本 CMake 示例的初始版本。 |
| **2020 年 6 月** | 已针对 2006年 GDK FAL 版本进行了更新。 |
| **2020 年 8 月** | 添加显式设置的 VS 启动项目。  更新了并行详细信息。 |
| **2020 年 11 月** | 扩展库需要显式引用 .lib 文件。  将 xmem.lib 和 xg\_\*.lib 添加到 Gaming.Xbox.\*.x64 的链接。 |
| **2021 年 2 月** | 修复了游戏.Desktop.x64 部署。 |
| **2021 年 4 月** | 添加 appnotify.lib 以解决 Gaming.Desktop.x64 的链接问题。  Add LargeLogo.png。 |
| **2021 年 6 月** | 修复了在生成的 vcxproj 中显式设置版本号以进行并行方案的错误。  常规代码清理。 |
| **2021 年 8 月** | 工具链文件的改进。  有关 BWOI 方案的其他说明。 |
| **2021 年 10 月** | 有关 BWOI 方案的进一步说明。 |
| **2022 年 1 月** | 添加了 VS 2022 支持。  CMake 清理并添加了预设文件。 |