CMake构建说明

**CMake**

CMake是一个跨平台的安装（[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343?fromModule=lemma_inlink)）工具，可以用简单的语句来描述所有平台的安装(编译过程)。他能够输出各种各样的makefile或者project文件，能测试[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067?fromModule=lemma_inlink)所支持的C++特性,类似[UNIX](https://baike.baidu.com/item/UNIX/219943?fromModule=lemma_inlink)下的automake。只是 CMake 的[组态档](https://baike.baidu.com/item/%E7%BB%84%E6%80%81%E6%A1%A3/4812025?fromModule=lemma_inlink)取名为 CMakeLists.txt。**CMake 并不直接建构出最终的软件，而是产生标准的建构档（如 Unix 的 Makefile 或 [Windows](https://baike.baidu.com/item/Windows/165458?fromModule=lemma_inlink) [Visual C++](https://baike.baidu.com/item/Visual%20C%2B%2B/1811800?fromModule=lemma_inlink) 的 projects/workspaces），然后再依一般的建构方式使用。**这使得熟悉某个[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/298524?fromModule=lemma_inlink)（IDE）的开发者可以用标准的方式建构他的软件。[【百度百科】](https://baike.baidu.com/item/cmake/7138032?fr=aladdin)

**优缺点比较**

**优点**

1. CMakeLists.txt是文本文件，容易处理冲突，可以减少工程配置冲突的修改难度
2. CMake添加工程、修改工程属性、修改工程名字等操作非常方便，且能够自动解析工程依赖关系
3. 可以使用支持CMake构建的IDE，如VS2022，VS Code
4. 统一了QuikPlatform、第三方的引入，进行跨平台或者Qt框架升级时，容易迁移、改造整个构建系统

**缺点**

1. CMake脚本语言庞大且复杂，但基础入门还是比较容易，需要多找资料，学习成本高

**SimuNPS仓库改动**

1. 主仓库添加了cmake相关文件

|  |  |
| --- | --- |
| CMakeLists.txt | 顶级CMakeList.txt |
| CMakeSettings.json | VS2022支持的CMake构建配置，默认支持4种：Debug、Release、RelWithDebInfo、MinSizeRel |
| .vscode | 使用VSCode开发时需要的配置文件 |
| cmake/modules/FindKL.cmake | QuikPlatform平台库查找支持，引入平台库只需要链接KL::XXX，如 KL::CoreUi |
| cmake/modules/FindThirdParty.cmake | 第三方库查找支持，引入第三方库只需要链接ThirdParty::XXX，如ThirdParty::QScintilla |
| cmake/modules/SimuNPSHelper.cmake | 公共函数支持，拷贝动态库到Debug/Release目录，拷贝插件到plugins目录 |

1. 每个子仓库根目录添加了CMakeLists.txt，指定了分组名。
2. 插件和源码目录添加了CMakeLists.txt，用于添加对应的构建项目。

**准备工作**

1. 安装CMake，如果已经安装了VS2022，则无需安装，找到cmake.exe所在目录即可。
2. 主仓库以及各子模块均拉取最新代码，目前仅在main分支支持。

**构建方式**

**生成VS解决方案**

执行主仓库根目录的cmake-generator.bat，内容如下代码，即可在当前目录生成build文件夹，生成目标为VS2022解决方案，使用V100平台工具集（即编译套件使用Visual Studio 2010），如需直接生成VS2010解决方案则替换最后一行为“cmake .. -B%BUILDDIR% -G "Visual Studio 10 2010" -A win32”。

@echo off

set BUILDDIR="%cd%/build"

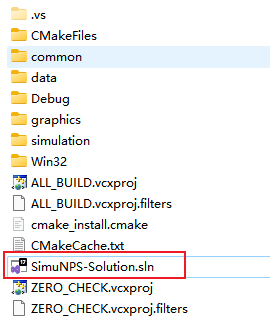
if not exist %BUILDDIR% md %BUILDDIR%

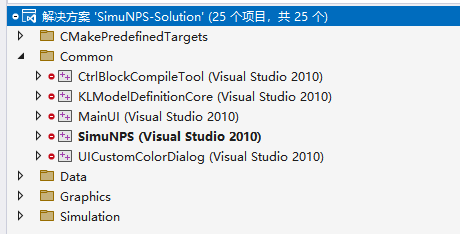
cd /D %BUILDDIR%

cmake .. -B%BUILDDIR% -G "Visual Studio 17 2022" -T V100 -A win32

打开build文件夹下面的SimuNPS-Solution.sln，设置SimuNPS为启动项目即可。

**注意：因为解决方案是根据CMakeLists.txt生成的，直接修改解决方案中项目的配置是无效的，只能修改项目对应的CMakeLists.txt，然后重新执行cmake-generator.bat。**





**使用VS Code**

使用VSCode开发需要安装以下插件

* 1. C/C++
  2. CMake
  3. CMake Tools

安装后，打开包含CMakeLists.txt的文件夹时，在状态栏会出现CMake相关的配置内容，说明如下：



**使用VS2022**

VS2022不仅支持sln解决方案，同样支持直接打开包含CMakeLists.txt的文件夹，VS2022打开主仓库文件夹后，选择生成所有即可编译，选择SimuNPS.exe作为调试目标即可调试。

默认已经编写好CMakeSettings.json配置文件，直接使用即可，包含x86平台的四种配置。

**其它**

**四种配置说明**

默认的四种配置说明如下，目前发版因需要定位问题，因此主要使用Debug和RelWithDebInfo两种配置。

|  |  |  |
| --- | --- | --- |
| Debug | 禁用优化，包括调试信息 | 生成到Debug目录 |
| Release | 进行优化，提高速度，排除调试信息 | 生成到Release目录 |
| RelWithDebInfo | 进行优化，提高速度，包括调试信息 | 生成到Release目录 |
| MinSizeRel | 优化最小二进制大小，排除调试信息 | 生成到Release目录 |

**注意事项和推荐使用**

1. 使用source\_group对源码文件分组，这样生成的VS解决方案包含筛选文件夹，如分类为头文件、源文件、资源文件、UI文件。
2. 各个插件或者动态库的源码文件夹，可以调整目录结构，使生成的解决方案种筛选器和文件夹一一对应。
3. 引入第三方库时，修改FindThirdParty.cmake，而不是直接在CMakeLists.txt引入头文件和库。
4. 运行和调试时，需要将Qt的bin目录加入环境变量，否则无法启动
5. CMake的版本不能高于3.24