计算机图形学小白入门

————从0开始实现OpenGL

CMake工程配置介绍

授课：赵新政
资深三维工程师

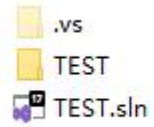
专注3D图形学技术
教育品牌

什么是Cmake?

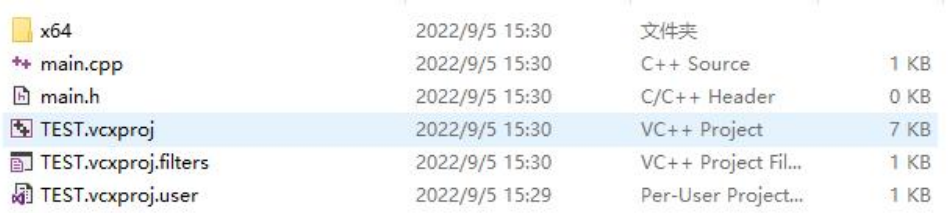
- CMake是一个跨平台的编译工具，可以用简单的语句来描述**所有平台**的编译链接过程

Visual Studio工程

- 使用Visual Studio创建的工程，属于windows下，并且只能够用相同的IDE打开（如下图）

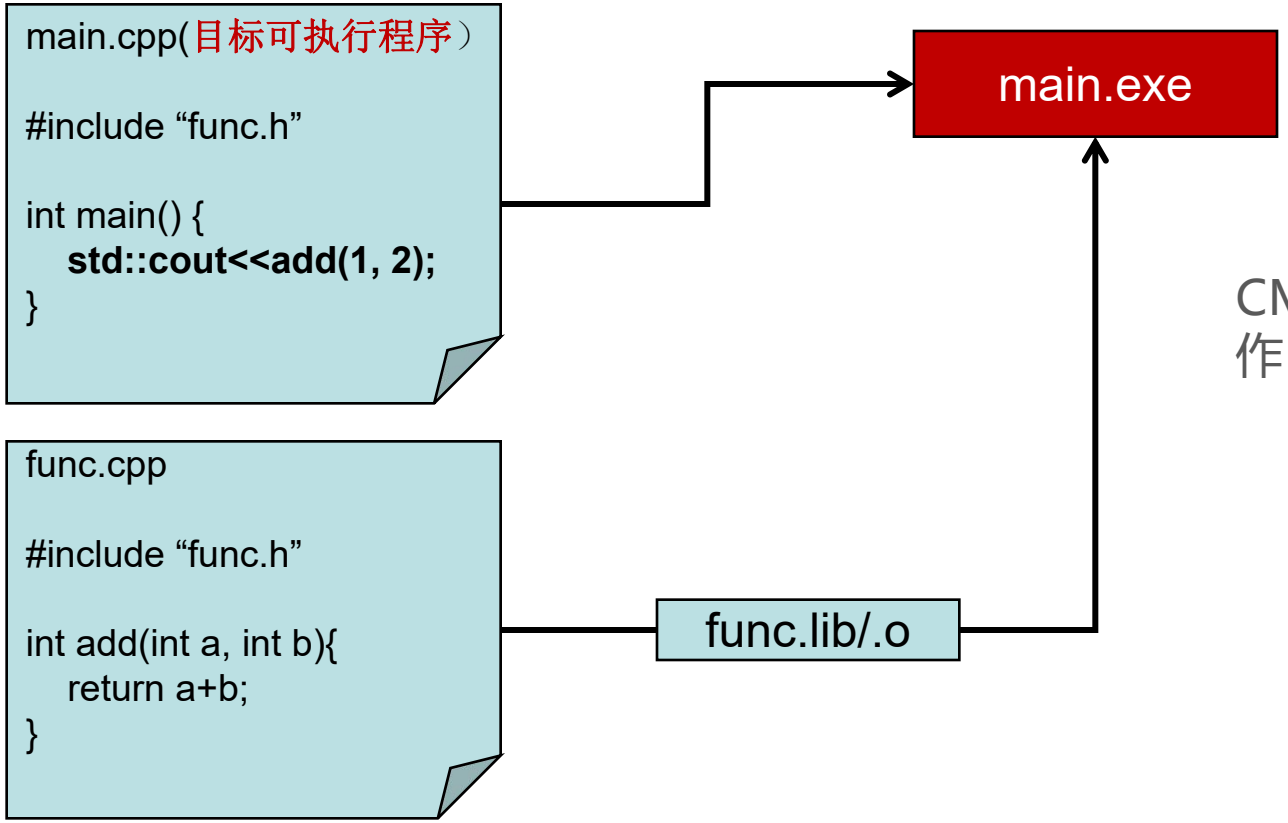


sln格式为解决方案
vcxproj为解决方案下的项目



编译与链接

- 编译：将当前的c++代码通过编译器，编译称为目标代码的过程
- 链接：将多段编译好的目标代码，互相进行链接，形成一个完整的可运行程序(exe)



CMake工具就可以完成编译与链接的配置工作，让程序员可以跨平台编译链接程序

介绍CMakeLists.txt

- **CMakeLists.txt**是整个CMake工程的描述文件，如下所示：


```
#需求的最低cmake程序版本
cmake_minimum_required(VERSION 3.12)

#本工程的名字
project(Graphic)

#本工程支持的C++版本
set(CMAKE_CXX_STANDARD 17)

#本工程主程序文件及输出程序名称
add_executable(softRenderer "main.cpp")
```

Visual Studio打开CMake工程

1 事先准备好一个CMakeLists.txt及cpp文件

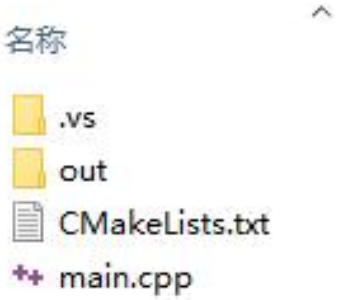
2 打开Visual Studio，文件->打开->CMake

3 选择对应的工程下的CMakeLists.txt打开

演示...

工程文件说明

-工程根目录



.vs文件夹：是visual studio建立的工程文件夹，不管它

out文件夹：是整个工程编译链接的结果输出文件夹

-out/build/x64-debug

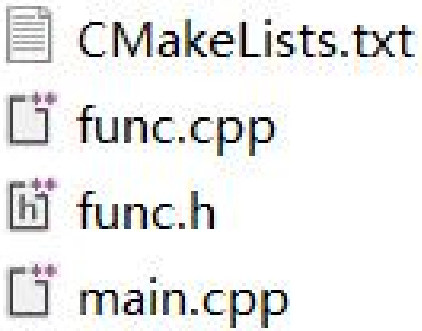
名称	修改日期	类型	大小
.cmake	2022/9/5 16:50	文件夹	
CMakeFiles	2022/9/5 16:50	文件夹	
Testing	2022/9/5 16:50	文件夹	
.ninja_deps	2022/9/5 16:54	NINJA_DEPS 文件	1 KB
.ninja_log	2022/9/5 16:54	NINJA_LOG 文件	1 KB
build.ninja	2022/9/5 16:50	NINJA 文件	33 KB
cmake_install.cmake	2022/9/5 16:50	CMake 源文件	2 KB
CMakeCache.txt	2022/9/5 16:50	文本文档	15 KB
softRenderer.exe	2022/9/5 16:54	应用程序	51 KB
softRenderer.ilk	2022/9/5 16:54	Incremental Link...	531 KB
softRenderer.pdb	2022/9/5 16:54	Program Debug...	1,180 KB
VSInheritEnvironments.txt	2022/9/5 16:50	文本文档	1 KB

这里是程序的运行目录，图示的softRender.exe就是最终的可执行程序

演示...

多CPP文件编译

- **需求**是当我们在工程里加入了新的cpp文件，主函数需要调用其中函数的时候，需要将其纳入编译与链接



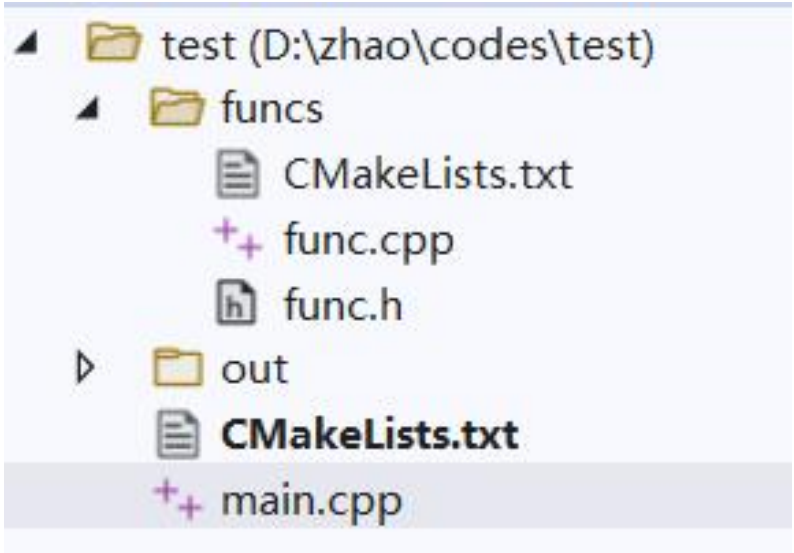

```
#需求的最低cmake程序版本
cmake_minimum_required(VERSION 3.12)

#本工程的名字
project(Graphic)

#本工程支持的C++版本
set(CMAKE_CXX_STANDARD 17)

#搜索所有的cpp，加入SRCS变量中
aux_source_directory(. SRCS)

#本工程所有cpp文件编译链接，生成exe
add_executable(softRenderer ${SRCS})
```

演示...

CMake介绍

多文件夹编译

- **需求**是当代码分布在不同的文件夹下的时候，需要将其他文件夹下的cpp打包为lib库，从而纳入链接范畴



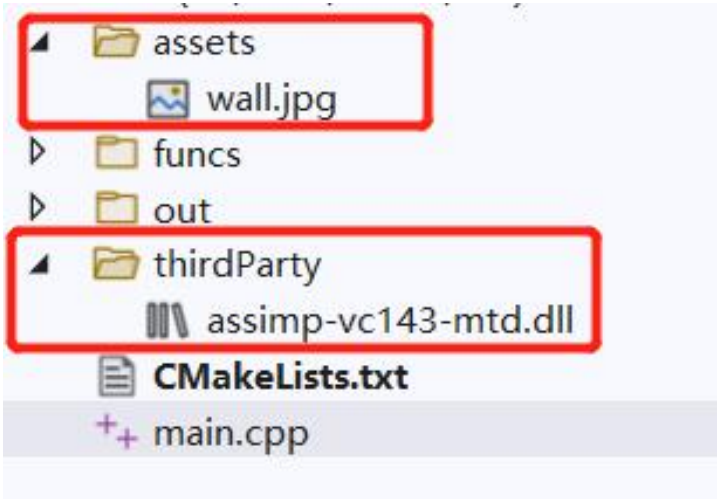

```
#需求的最低cmake程序版本
cmake_minimum_required(VERSION 3.12)

#本工程的名字
project(Graphic)

#本工程支持的C++版本
set(CMAKE_CXX_STANDARD 17)

#将funcs文件夹纳入到编译系统
add_subdirectory(funcs)

#搜索所有的cpp，加入SRCS变量中
aux_source_directory(. SRCS)

#本工程所有cpp文件编译链接，生成exe
add_executable(softRenderer ${SRCS})

#将funcs.lib链接入softRender
target_link_libraries(softRenderer funcs)
```

```
#递归将本文件夹下所有cpp放到FUNCS中
file(GLOB_RECURSE FUNCS ./  *.cpp)

#将FUNCS中所有cpp编译为funcs这个lib库
add_library(funcs ${FUNCS} )
```

演示...

资源文件拷贝

- **需求**是当我们工程中出现资源文件（图片、模型、音频、视频、**动态链接库**等），都需要拷贝到编译链接完成的exe所在目录下面，才能够被程序正确读取，所以需要拷贝功能



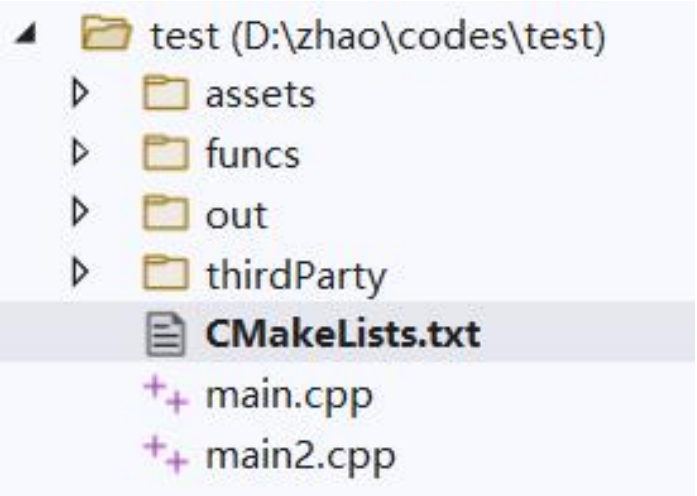

```
.....

#把需要拷贝的资源路径都放到ASSETS里
file(GLOB ASSETS "./assets" "thirdParty/assimp-vc143-mtd.dll")

#把ASSETS指代的目录集合的内容，都拷贝到可执行文件目录下
file(COPY ${ASSETS} DESTINATION ${CMAKE_BINARY_DIR})

.....
```

- **注意**：每次修改了CMakeLists.txt，**或者**增加了新的assets资源，必须在CMakeLists.txt界面上重新点击Ctrl+s保存一次，才能够促使CMake重新做一次缓存更新以及资源拷贝

演示...

多编译目标

- **需求**是当我们在一个工程里，有多个main程序的时候，希望可以每次选择一个执行




```
.....................

#本工程主程序文件及输出程序名称
add_executable(main "main.cpp")
add_executable(main2 "main2.cpp")

#将funcs.lib链接入softRender
target_link_libraries(main funcs)
```

演示...

总结

- CMakeLists.txt是CMake工程管理的核心
- 基础CMakeLists.txt的书写方式
- 多Cpp文件编译链接方法
- 多文件夹代码编译链接方法
- 资源文件&动态链接库拷贝运行环境目录方法
- 多运行目标编写方法