CGAssignment: Setup

本代码框架是一个仿OpenGL的软光栅化渲染器，请同学们仔细阅读此文档部署框架代码。

环境要求

本代码框架仅在Windows 10和Ubuntu 16.04上测试编译通过，因此建议同学们在这两个系统环境中根据自己的喜爱偏好选一个。对于其他非Windows和Ubuntu的系统，请根据环境依赖自行尝试部署代码编译环境。（相比于Ubuntu系统，我们更倾向于推荐Windows系统）

对于Windows的用户，我们要求你的环境安装以下的软件：

- 安装Microsoft Visual Studio 2017或者Microsoft Visual Studio 2019（更低版本的VS未测试过）
- 安装CMake，要求CMake版本至少为3.5，用于项目构建

对于Ubuntu的用户，我们要求你的环境安装以下的软件：

- 安装SDL2，在终端输入以下的命令安装SDL2，用于部署代码的第三方库

```
sudo apt-get update
sudo apt-get install libsdl2-2.0
sudo apt-get install libsdl2-dev
```

- 安装CMake，同样至少版本为3.5，可在终端输入以下的命令安装：

```
sudo apt-get install cmake
```

- 安装Make，用于编译代码，生成可执行程序，可在终端输入以下的命令安装：

```
apt-get install make
```

- 安装C++编译环境，安装gcc和g++：

```
sudo apt-get install gcc-5
sudo apt-get install g++-5
```

以上软件如已安装，请直接忽略。对于Mac系统的同学，请类比Ubuntu尝试自行部署。

项目目录

本代码框架的目录结构如下所示：

```
├─build ----------------------> 二进制文件目录，存放项目工程文件、中间文件、编译结果
|  ├─Debug -------------------> vs在debug模式下的编译结果存放目录
|  ├─model -------------------> 模型数据文件目录，存放待渲染的模型文件
|  └─Release -----------------> vs在release模式下的编译结果存放目录
├─include -------------------> 项目依赖的第三方库头文件目录
|  ├─glm ---------------------> 数学库头文件目录
|  └─SDL2 --------------------> 窗口库头文件目录
├─libs ----------------------> 项目依赖的动态链接库文件目录
└─src -----------------------> 项目的源代码文件目录（存放项目的.h文件和.cpp文件）
```

在本项目代码的根目录，有两个文件需要注意：

- `CMakeLists.txt`：cmake的项目构建规则文件，如不清楚构建规则请勿做任何改动
- `LICENSE.txt`：本项目代码的开源协议MIT许可证，同学们可以直接忽略

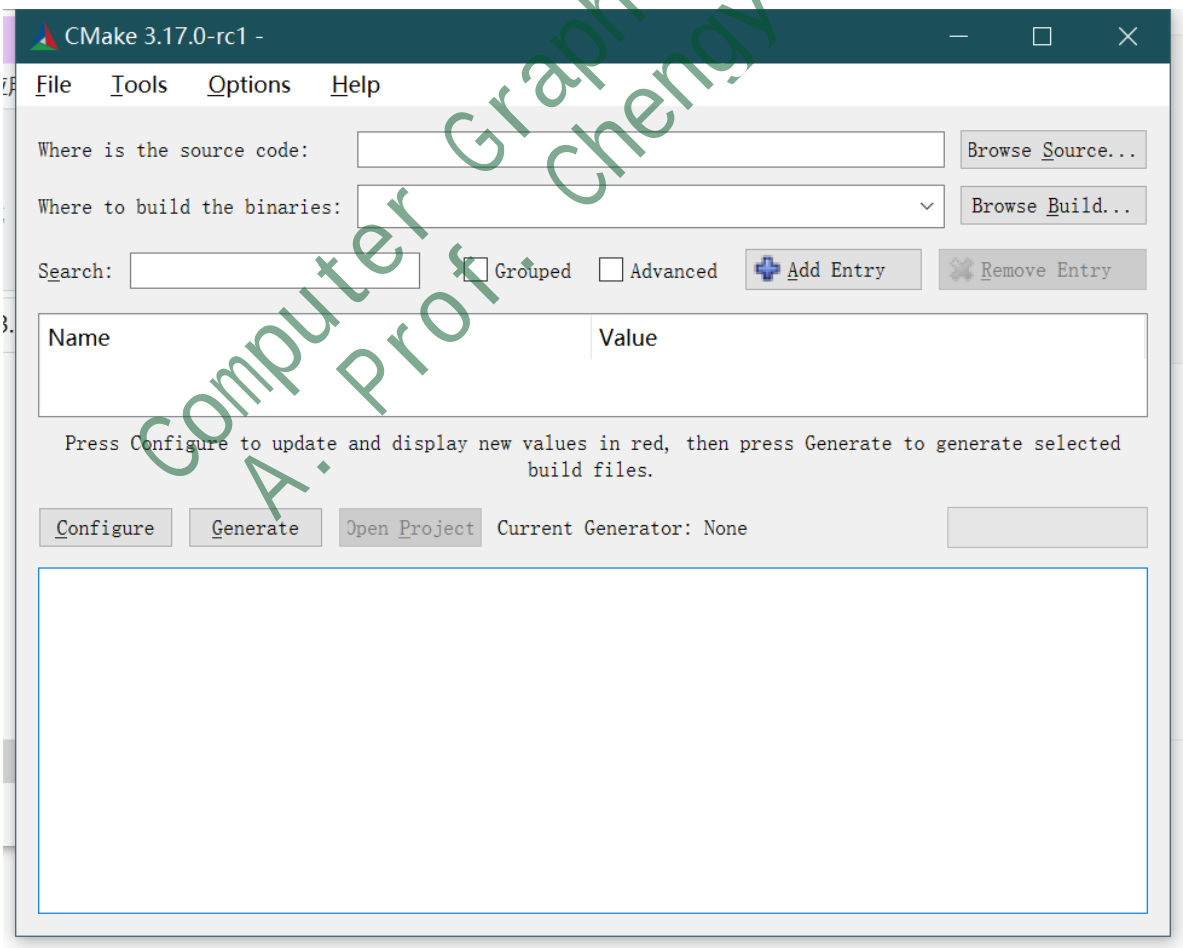
本项目代码的程序入口为`src/main.cpp`。

项目构建

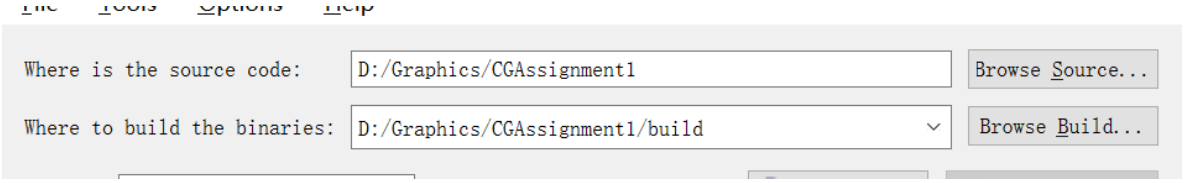
本代码框架采用cmake规则进行自动化构建相应的工程项目。以下以CGAssignment1为例。

对于Windows的用户，请按照如下的方式构建：

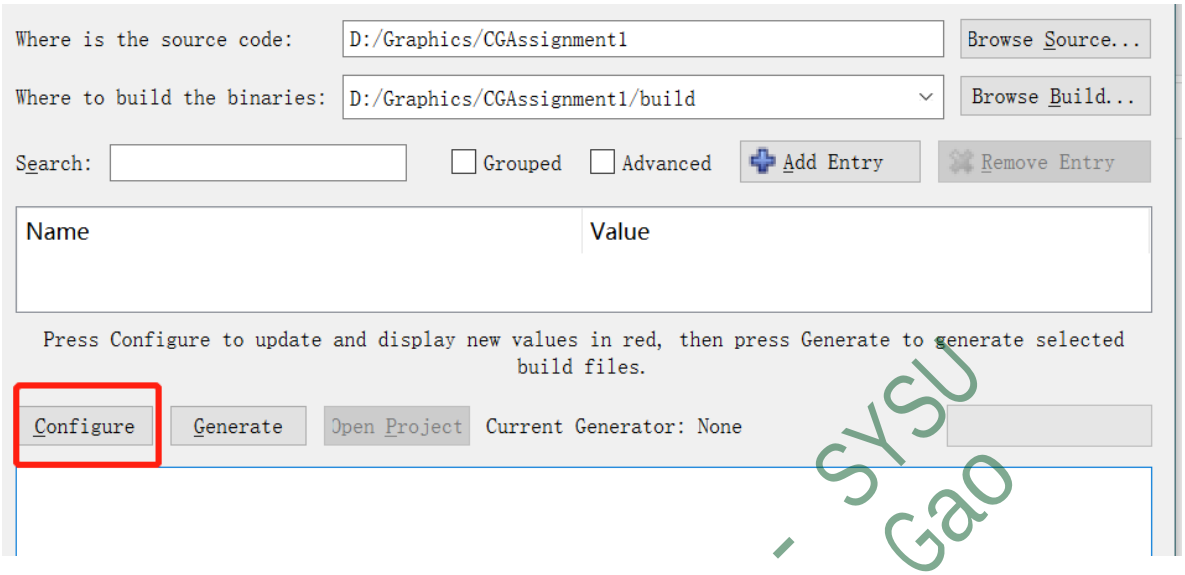
- 以构建VS 2017的工程项目为例，打开cmake-gui.exe软件，弹出以下的窗口：



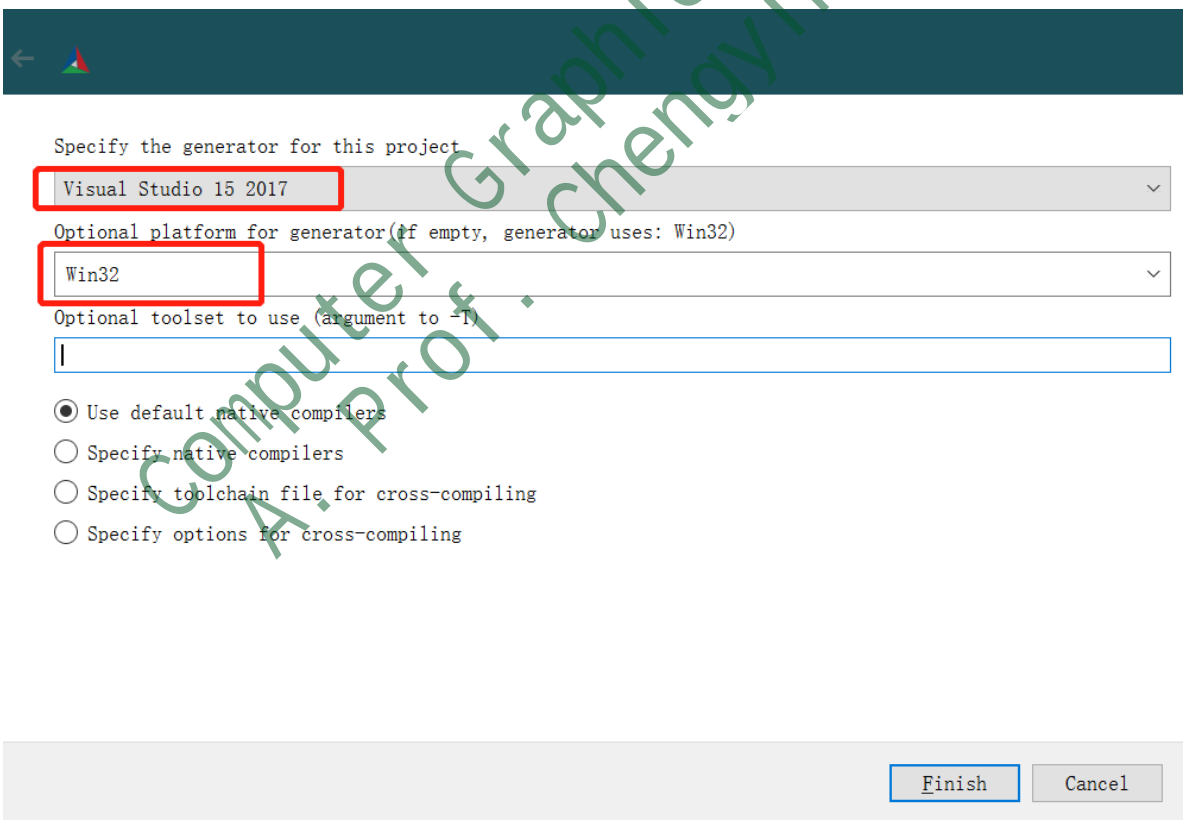
- 点击Where is the source code右边的Browse Source...按钮，找到CGAssignemtn1的根目录并确定；同样地，点击Where to build the binaries右边的Browse Build...按钮，找到CGAssignemtn1/build的目录并确定：



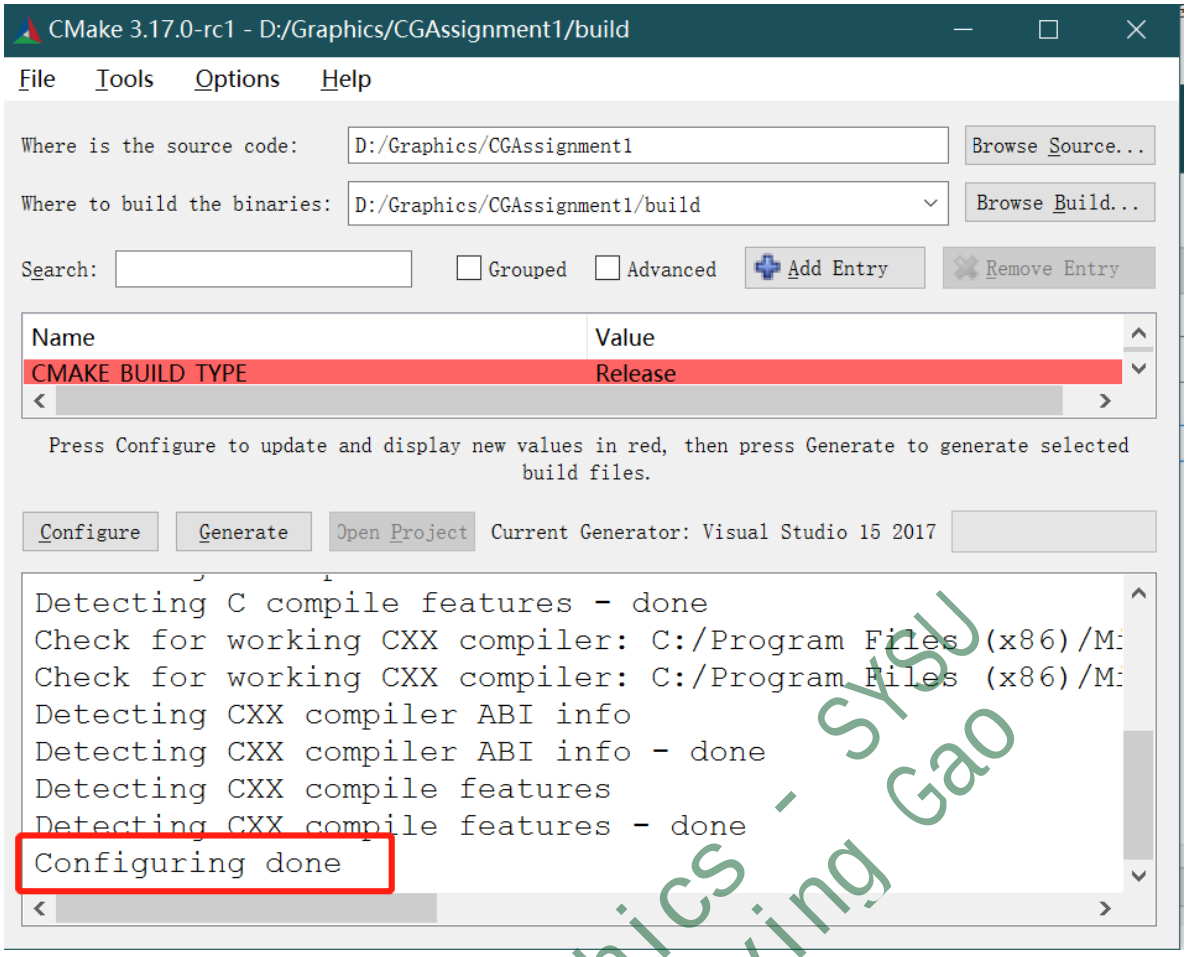
- 然后点击下面的Configure按钮



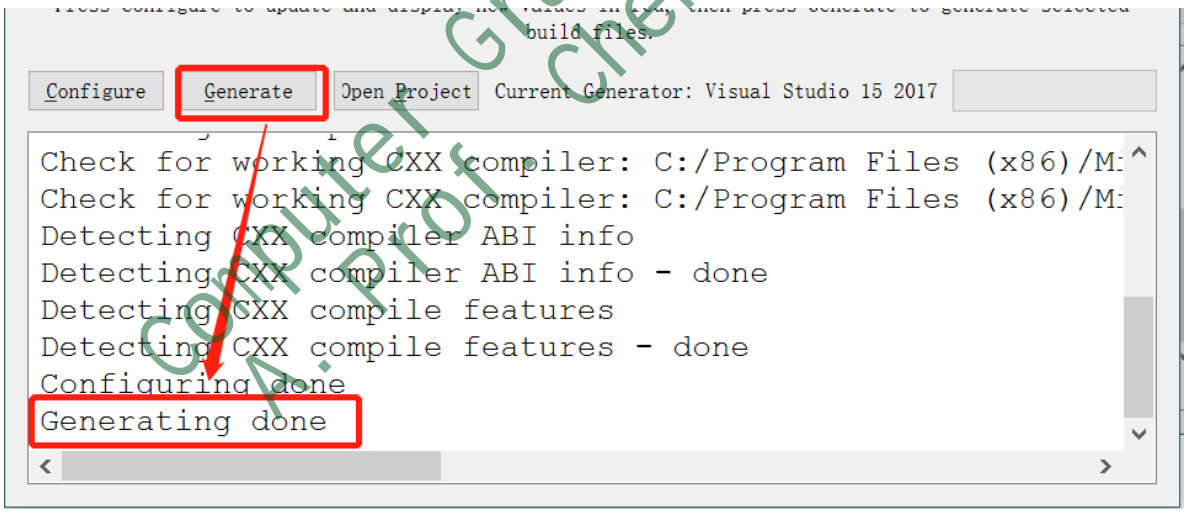
- 此时会弹出一个窗口，先选择VS 2017或VS 2019的项目类型，然后选择Win32的项目构建平台（**请注意必须要Win32的，因为依赖库的lib是x86的**），最后点击Finish即可：



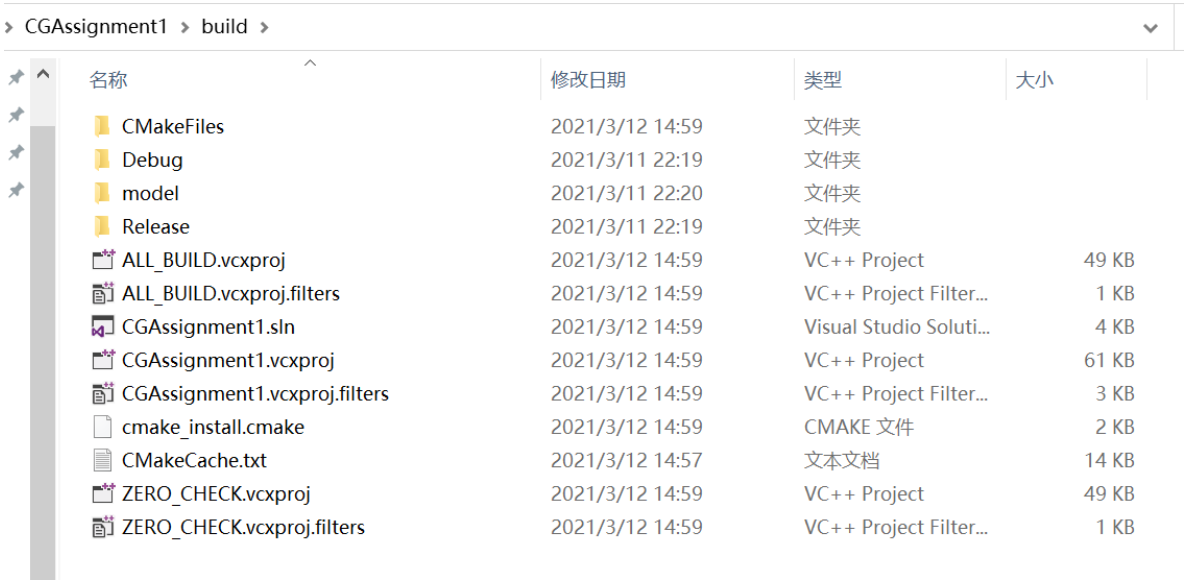
此时cmake会开始构建项目的工程文件，输出框会输出相应的构建信息，出现Configuring done表示配置成功：



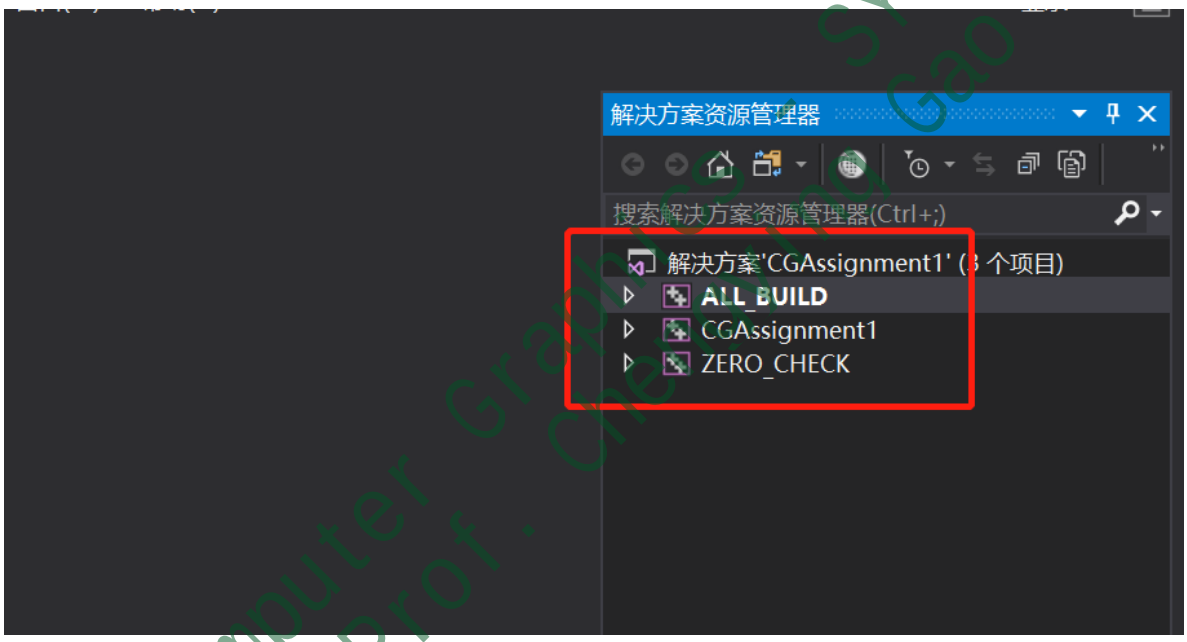
- 然后再点击Configure按钮右边的Generate按钮进行项目生成，输出框出现Generating done表示构建成功：



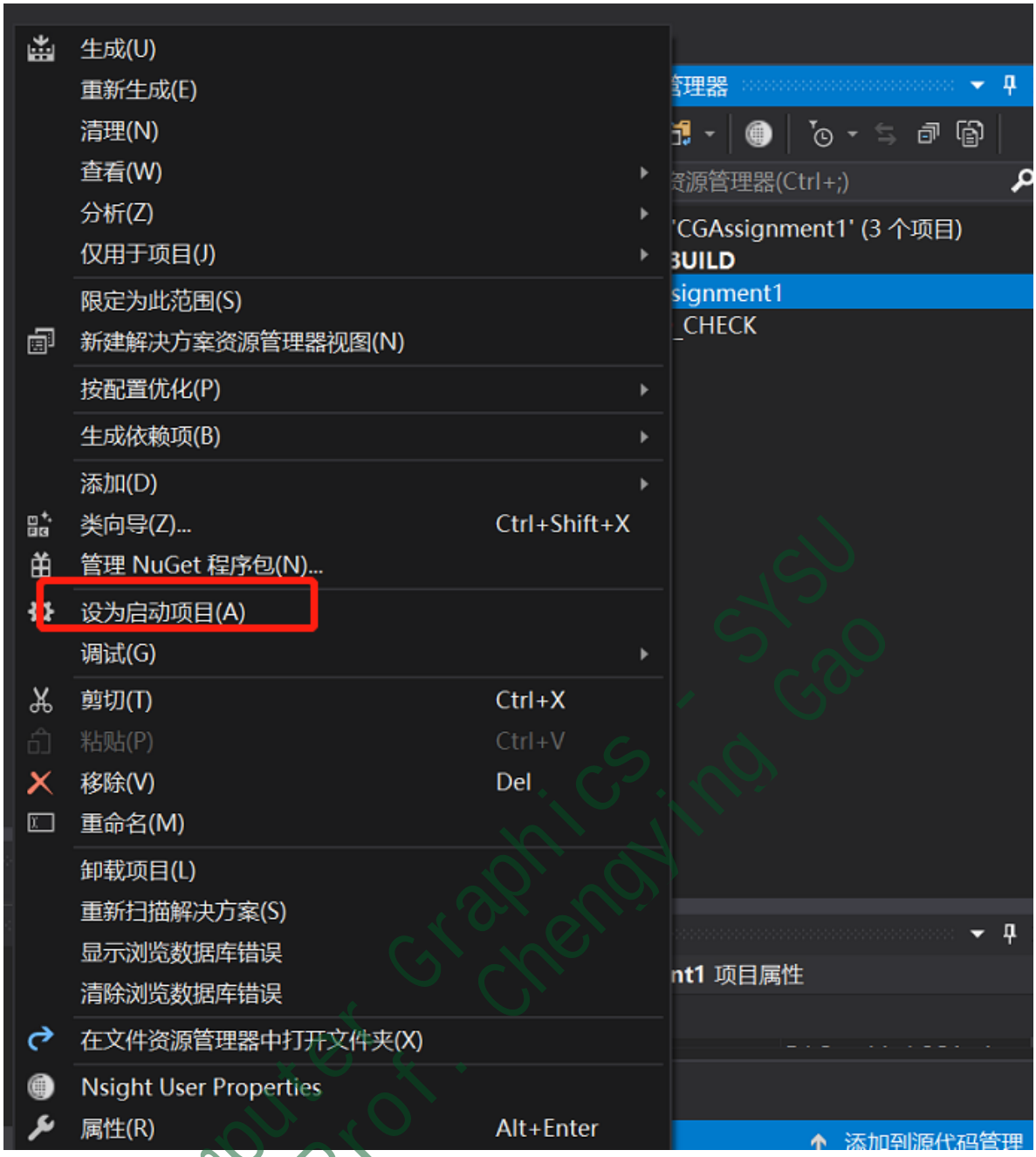
构建成功之后，在build目录下有如下所示的VS工程文件：



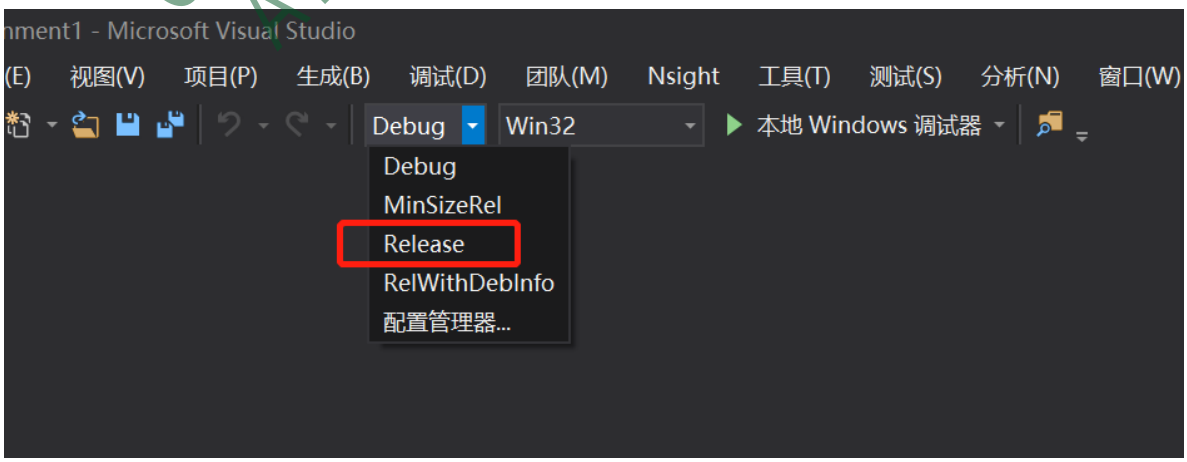
- 用VS2017打开CGAssignment1.sln文件，会启动三个项目，如下所示，分别是ALL_BUILD、CGAssignment1和ZERO_CHECK，我们只需关注CGAssignment1项目。



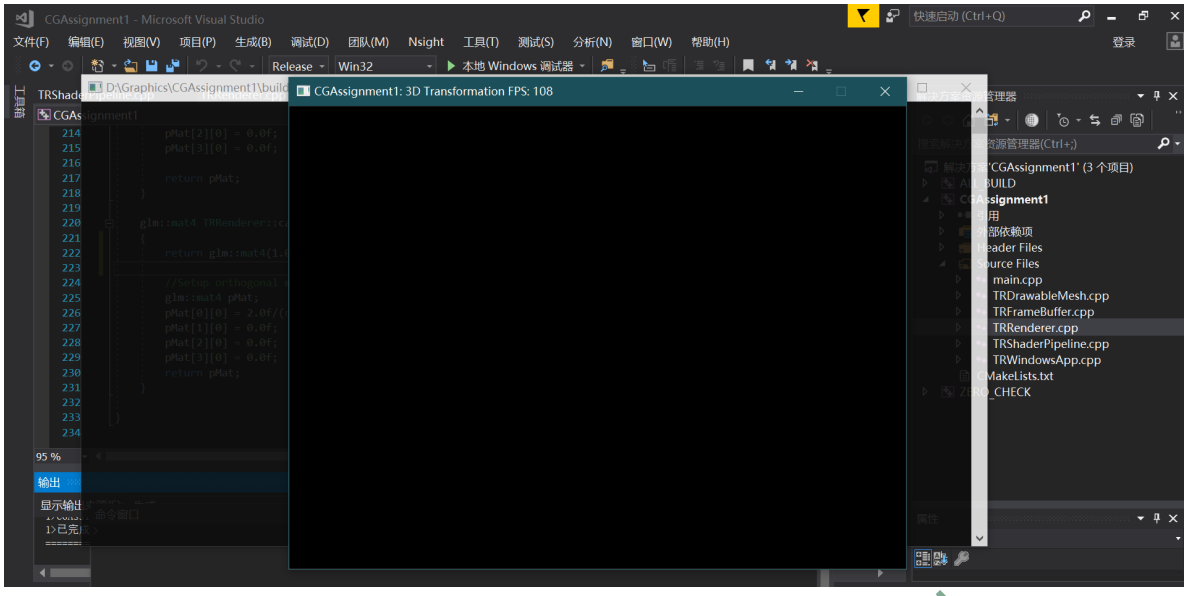
- 第一次打开默认启动的项目是ALL_BUILD，因此需要**将CGAssignment1设置为启动项目**，鼠标箭头移到CGAssignment1上并单击鼠标右键，在弹出的选项中点击设为启动项目即可。



- 由于本框架代码运算量大，为了有更好的程序运行体验，**请尽可能在Release模式下编译该项目**，在需要Debug的时候再切换到Debug模式。



- 编译成功并运行后将弹出一个窗口，表示构建成功。可通过点击窗口左上角的x按钮或者按键盘的ESC键退出窗口，终止程序运行。



对于Windows系统构建项目的同学，请注意：

（1）务必尽量用Release模式编译程序，可以有一个比较流畅的编程体验

（2）编译构建平台必须是Win32的，请勿改成x64

对于Ubuntu的用户，在终端输入相应的命令构建：

- 首先进入到CGAssignment1的目录下：

```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1$ pwd
/home/yangwc/Desktop/CGAssignment1
```

- 然后进入到CGAssignment1/build目录下：

```
cd build
```

```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1$ cd build
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1/build$ pwd
/home/yangwc/Desktop/CGAssignment1/build
```

- 紧接着在终端输入以下的命令进行构建：

```
cmake ..
```

```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1/build$ cmake ..
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 5.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
SDL2 found
-- Configuring done
-- Generating done
-- Build files have been written to: /home/yangwc/Desktop/CGAssignment1/build
```

请注意终端的输出信息查看是否构建成功（即上图的`Configuring done`和`Generating done`），构建成功之后，在build目录有以下的文件（Debug和Release目录针对于Windows用户，这里直接忽略）：

```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1/build$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile  model
```

其中最重要的就是`Makefile`文件，有了该文件，我们就可以直接使用如下的命令对项目代码进行编译：

- 使用Make命令对项目代码进行编译，生成可执行文件：

```
make
```

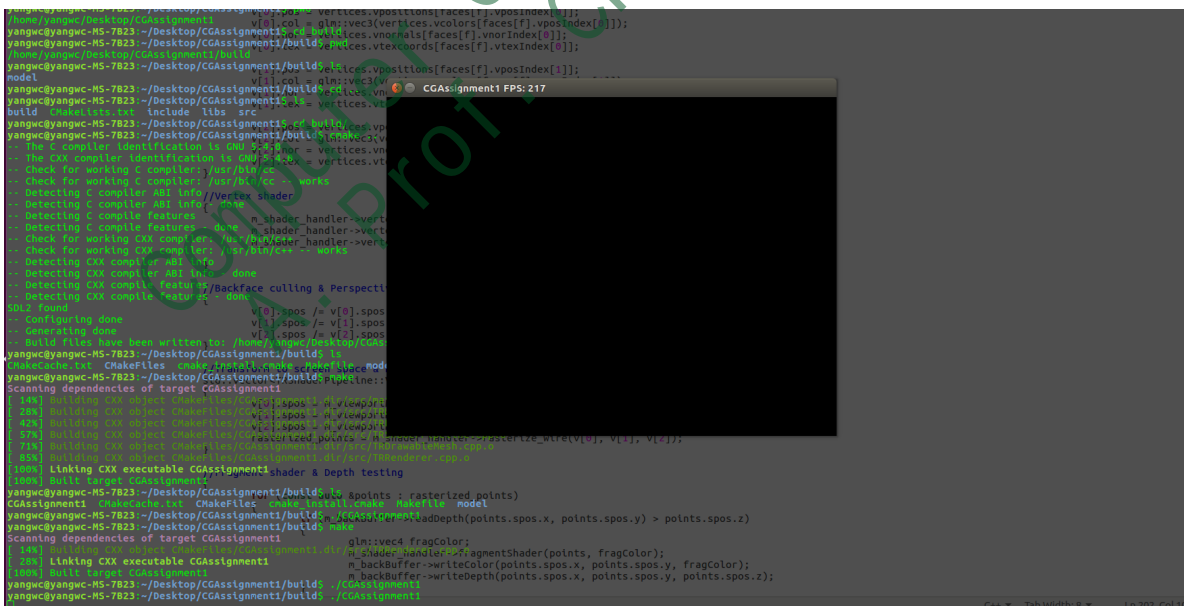
```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1/build$ make
Scanning dependencies of target CGAssignment1
[ 14%] Building CXX object CMakeFiles/CGAssignment1.dir/src/main.cpp.o
[ 28%] Building CXX object CMakeFiles/CGAssignment1.dir/src/TRWindowsApp.cpp.o
[ 42%] Building CXX object CMakeFiles/CGAssignment1.dir/src/TRFrameBuffer.cpp.o
[ 57%] Building CXX object CMakeFiles/CGAssignment1.dir/src/TRShaderPipeline.cpp.o
[ 71%] Building CXX object CMakeFiles/CGAssignment1.dir/src/TRDrawableMesh.cpp.o
[ 85%] Building CXX object CMakeFiles/CGAssignment1.dir/src/TRRenderer.cpp.o
[100%] Linking CXX executable CGAssignment1
[100%] Built target CGAssignment1
```

如果编译过程出现错误，则会在终端输出相应的错误提醒，并终止编译过程，表示构建失败，请注意同学们根据给出的编译错误对源代码进行纠正（理论上提供的代码不会有错误）。编译成功则会在当前目录下生成可执行文件CGAssignment1：

```
yangwc@yangwc-MS-7B23:~/Desktop/CGAssignment1/build$ ls
CGAssignment1  CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile  model
```

- 在终端键入以下命令则可以运行生成的可执行文件：

```
./CGAssignment1
```



运行成功会弹出一个窗口，表示项目成功构建。可通过点击窗口左上角的x按钮或者按键盘的ESC键退出窗口，终止程序运行。对于Ubuntu系统构建项目的同学，请注意：

（1）如有新增新的`.h`文件或`.cpp`文件的需要，请在`/src`目录下添加相应的代码文件，并重新执行cmake构建，cmake会自动扫描该目录下所有的C++代码文件并重新生成Makefile，然后再执行make即可。

（2）若仅修改代码文件，则无需再重新执行cmake构建，只需执行make重新编译即可。

项目开发

项目构建成功之后，同学们就可以根据我们提供的作业说明文档开始编写相应的代码。**同学们在提交代码时，请注意清除build目录下的所有工程文件、中间文件、二进制文件，确保只提交源代码文本即可。**在构建过程遇到问题，请同学们及时向助教反馈！

License