







# 框架配置指南

## 引言

这是一个点云和网格处理的框架，融合了 Libigl 和 CGAL 两个主流的几何运算库。这里只提供了源码 src，没有写 cmake，需要自己手动配置 Libigl 和 CGAL，然后复制源码到项目里。

首先源码主要四个部分：

-  main.cpp
-  PointCloud.cpp
-  PointCloud.h
-  TriMesh.cpp
-  TriMesh.h
-  Type.h

- (1) Main.cpp：包含了可视化的代码。采用的是 Libigl 的可视化框架，原因是 Libigl 的可视化用的是 ImGui，CGAL 是 Qt，个人认为 ImGui 要比 Qt 好用亿点。
- (2) Type.h：包含了 CGAL 中的数据类型定义 (typedef)
- (3) PointCloud.h/.cpp：一个点云类，这里点云的数据结构采用的是 CGAL 的数据结构，主要原因是 Libigl 的点云算法还是比较少，主要集中在网格处理，CGAL 虽然复杂，但是确实强大，而且官网文档齐全，虽然是英文。
- (4) TriMesh.h/.cpp：一个网格类，这里网格的数据结构采用的是 Libigl 的数据结构，也就是 Eigen::Matrix<T, N, 1>，Libigl 的最大优点就是比 CGAL 要简单，数据结构很像 Matlab，CGAL 的网格采用半边数据结构，我个人是不太爱用半边，很麻烦。Libigl 里面封装了很多方法可以计算邻域等，可以直接

调用。另外我也写了一个 TriMesh::SetMesh 方法可以进行 Libigl Mesh 和 CGAL Mesh 的转化。

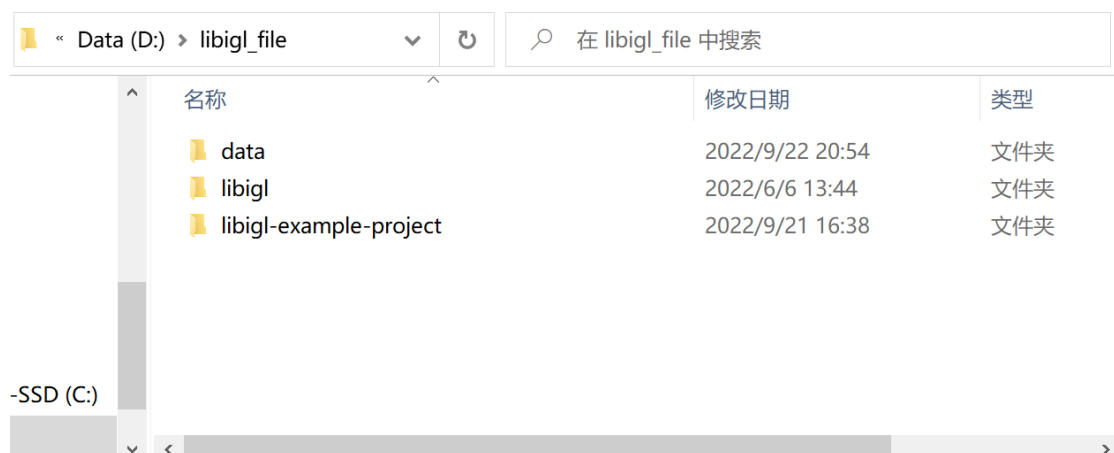
## Libigl 配置

Libigl 的配置和教程文档可以参考官网 <https://libigl.github.io/>。

这里提一下，个人认为还是采用空项目的方式安装 Libigl 的方式比较方便(官网也有写)，就是把 libigl 和 libigl-example-project 放到一起

### libigl Example Project

We provide a [blank project example](#) showing how to use libigl and cmake. Feel free and encouraged to copy or fork this project as a way of starting a new personal project using libigl.



忽略图里的 data 文件夹。

然后直接 cmake 这个 libigl-example-project 就可以得到一个空项目连接了 Libigl，使用起来很方便。

注意，需要把 libigl-example-project 的 cmakelists 修改一下，添加 ImGui 组件进去

```
cmake_minimum_required(VERSION 3.16)
project(example)

list(PREPEND CMAKE_MODULE_PATH ${CMAKE_CURRENT_SOURCE_DIR}/cmake)

# Libigl
option(LIBIGL_Glfw "Build target igl::glfw" ON)
option(LIBIGL_Imgui "Build target igl::imgui" ON)
include(libigl)

# Add your project files
file(GLOB SRC_FILES *.cpp)
add_executable(${PROJECT_NAME} ${SRC_FILES})
target_link_libraries(${PROJECT_NAME} PUBLIC igl::glfw)
target_link_libraries(${PROJECT_NAME} PUBLIC igl::imgui)
```

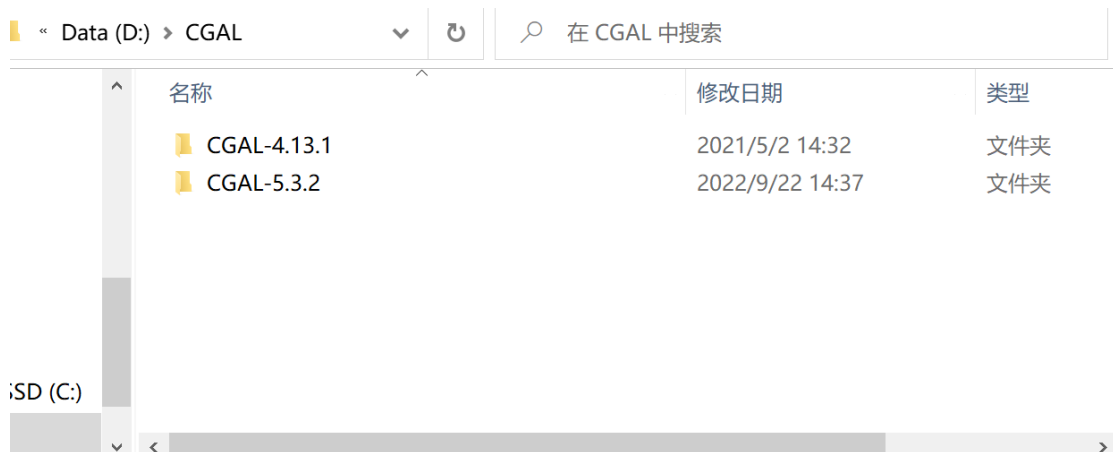
这样编译完之后就可以得到包含 Libigl 和 Imgui 的空项目了。

## CGAL 配置

CGAL 作为最强大的几何库，官网文档太齐全了，怎么配置直接看官网，主要就是需要 Boost 和 GMP，也可以参考

[https://blog.csdn.net/liyy001/article/details/123005145?ops\\_request\\_misc=%257B%2522request%255Fid%2522%253A%2522166390147216800186515909%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\\_id=166390147216800186515909&biz\\_id=0&utm\\_medium=distribute.pc\\_search\\_result.none-task-blog-2~all~top\\_click~default-3-123005145-null-null.142^v50^control,201^v3^add\\_ask&utm\\_term=CGAL&spm=1018.2226.3001.4187](https://blog.csdn.net/liyy001/article/details/123005145?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522166390147216800186515909%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=166390147216800186515909&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_click~default-3-123005145-null-null.142^v50^control,201^v3^add_ask&utm_term=CGAL&spm=1018.2226.3001.4187)

注意，这里不要配置 Qt，Qt 这东西是真的麻烦，我们直接用 Imgui 轻松多了，不要配置 Qt，不要配置 Qt。



这是我安装好的 CGAL 文件夹，一个新版本一个老版本。

注意，本框架用的 CGAL 新版本(5.3 之后)，可以自己下个老版本以备不时之需。

接下来需要说的是配置完 CGAL 之后，怎么在自己的项目(libigl-example-project)里面链接 CGAL

CGAL 是一个头文件库，只要 include 头文件就行，跟 Libigl 差不多，但是需要在项目中同时连接 boost 和 GMP (include 和 lib)

具体方法可以参考

[https://blog.csdn.net/summer\\_dew/article/details/107811371](https://blog.csdn.net/summer_dew/article/details/107811371)

## 在VS中使用CGAL库

### 引用boost

boost的dll和lib文件名中包含 **gd** 的为debug版本，这里以使用release版本为例

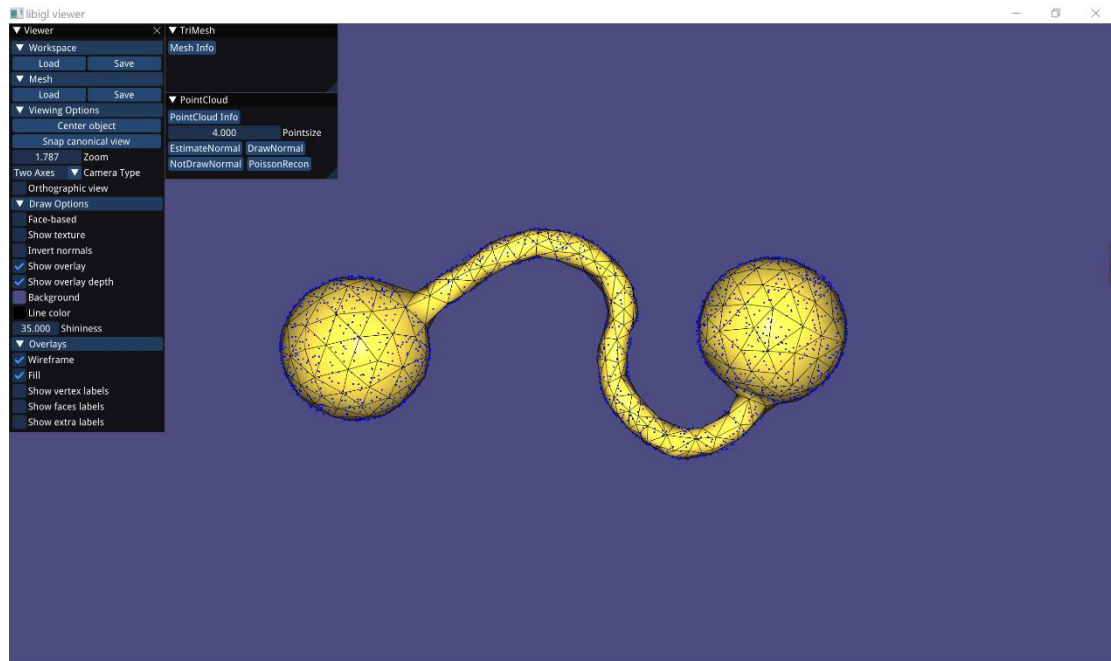
1. include目录: **D:\local\boost\_1\_71\_0**
2. lib文件目录: **D:\local\boost\_1\_71\_0\lib64-msvc-14.1**
3. lib文件名列表 (release版本)

(或者直接 **D:\local\boost\_1\_71\_0\lib64-msvc-14.1\\*.lib** )

```
1 | boost_container-vc141-mt-gd-x64-1_71.lib
2 | boost_date_time-vc141-mt-gd-x64-1_71.lib
3 | libboost_exception-vc141-mt-gd-x64-1_71.lib
```

## 使用

配置完 Libigl 和 CGAL 之后就复制 src 里面的源码编译就可以了。



这个图是 Poisson 重建的结果。

自己使用的时候就在 main 里面加入 imgui 的控件，然后 TriMesh 和 PointCloud 封装相应的算法就 Ok 了，个人觉得用起来还是很方便的。