实验1.1 OpenGL图形编程环境配置

1. 实验目的
2. 掌握Xcode 12集成开发环境的安装和使用。
3. 掌握Homebrew软件包管理工具的安装和使用。
4. 掌握CMake跨平台构建工具的安装和使用。
5. 了解和掌握OpenGL的环境配置。
6. 掌握OpenGL工程项目的建立和基本设置。
7. 实验环境
8. 硬件环境：主流配置，最好配备独立显卡。
9. 软件环境
   1. 操作系统： macOS(11.0及以上)
   2. 集成开发环境：Xcode 12
   3. 构建系统：CMake
   4. 库安装工具：homebrew
   5. 开发库：OpenGL, GLFW, GLAD, GLM

后续的实验环境与此相同，以后不再赘述。

1. 理论背景
2. OpenGL简介

OpenGL（Open Graphics Library）是指定义了一个跨编程语言、跨平台的编程接口规格的专业的图形程序接口。它用于三维图像（二维的亦可），是一个功能强大，调用方便的底层图形库。OpenGL是行业领域中最为广泛接纳的 2D/3D 图形 API，其自诞生至今已催生了各种计算机平台及设备上的数千优秀应用程序。

1. GLFW简介

GLFW（Graphics Library Framework），它是一个小型 C 库，允许使用 OpenGL 上下文创建和管理窗口，从而也可以使用多个监视器和视频模式。它同时提供对键盘、鼠标和操纵杆输入的访问。

1. GLAD简介

由于OpenGL驱动版本众多，大多数函数的位置都无法在编译时确定下来，需要在运行时进行查询，使得开发过程较为繁琐。GLAD (Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator based on the official specs) 是一个开源的库，可以提供一种更为简单的方法加载所有相关的OpenGL函数。

1. GLM简介

GLM（OpenGL Mathematics）是专门为OpenGL量身定做的数学库，它是一个只有头文件的库，可以快速实现矩阵变换等各种图形学中常用的几何计算。

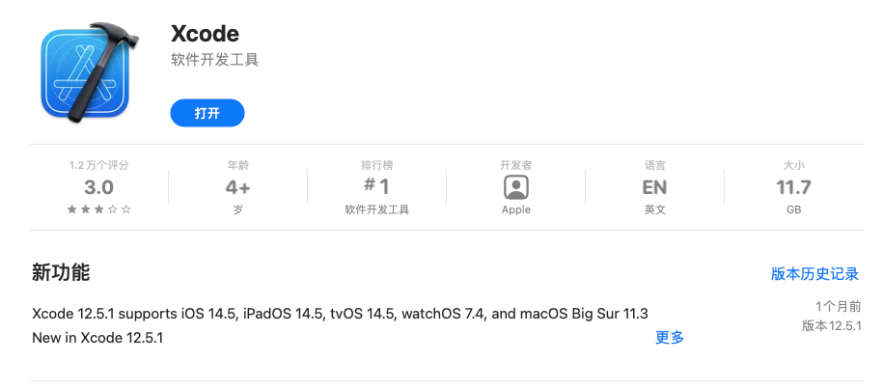
1. 三者的关系

OpenGL定义标准 → 显卡厂商编写驱动 → GLAD提供接口 → GLFW建立窗口

1. 实验内容
2. **环境配置**

（1）集成开发环境：Xcode12的安装

Xcode是苹果公司提供的集成开发环境，在macOS上的App Store就可以直接下载安装。Xcode自带很多由苹果维护的库和框架，包括本课程要使用的OpenGL。注意：目前Xcode仅支持macOS 11.0 及以上的系统安装使用。



（2）软件包管理工具：Homebrew的安装

Homebrew是macOS上一款非常方便好用的软件包管理工具，我们将用homebrew来安装cmake，glfw以及glm库。



a) 打开终端（推荐使用iTerm2，也可使用系统自带的终端），输入命令：/bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)" 即可自动安装。国内的源下载网速会更快一点，安装命令如下：/bin/zsh -c "$(curl -fsSL <https://gitee.com/cunkai/HomebrewCN/raw/master/Homebrew.sh>)"。

b) Homebrew常用命令：

|  |  |
| --- | --- |
| 搜索包 | brew search PACKAGE |
| 安装包 | brew install PACKAGE |
| 查看已安装的包 | brew list |
| 升级已安装的包 | brew upgrade [PACKAGE]  (无参数时将升级所有包) |

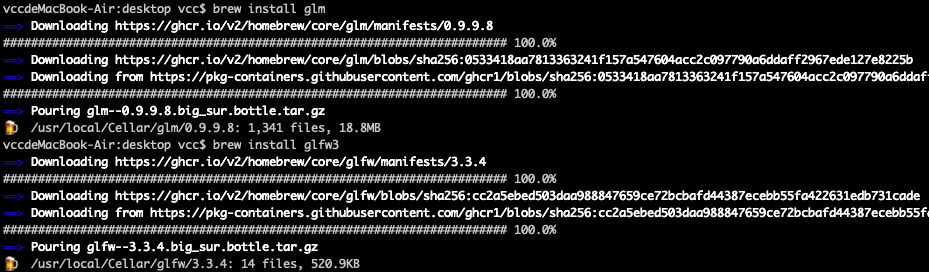
通过Homebrew下载的包会存放在 /usr/local/Celler/ 目录下。

（3）构建工具：CMake的安装

有了Homebrew，cmake的安装就非常容易了，只需在终端输入命令：brew install cmake。

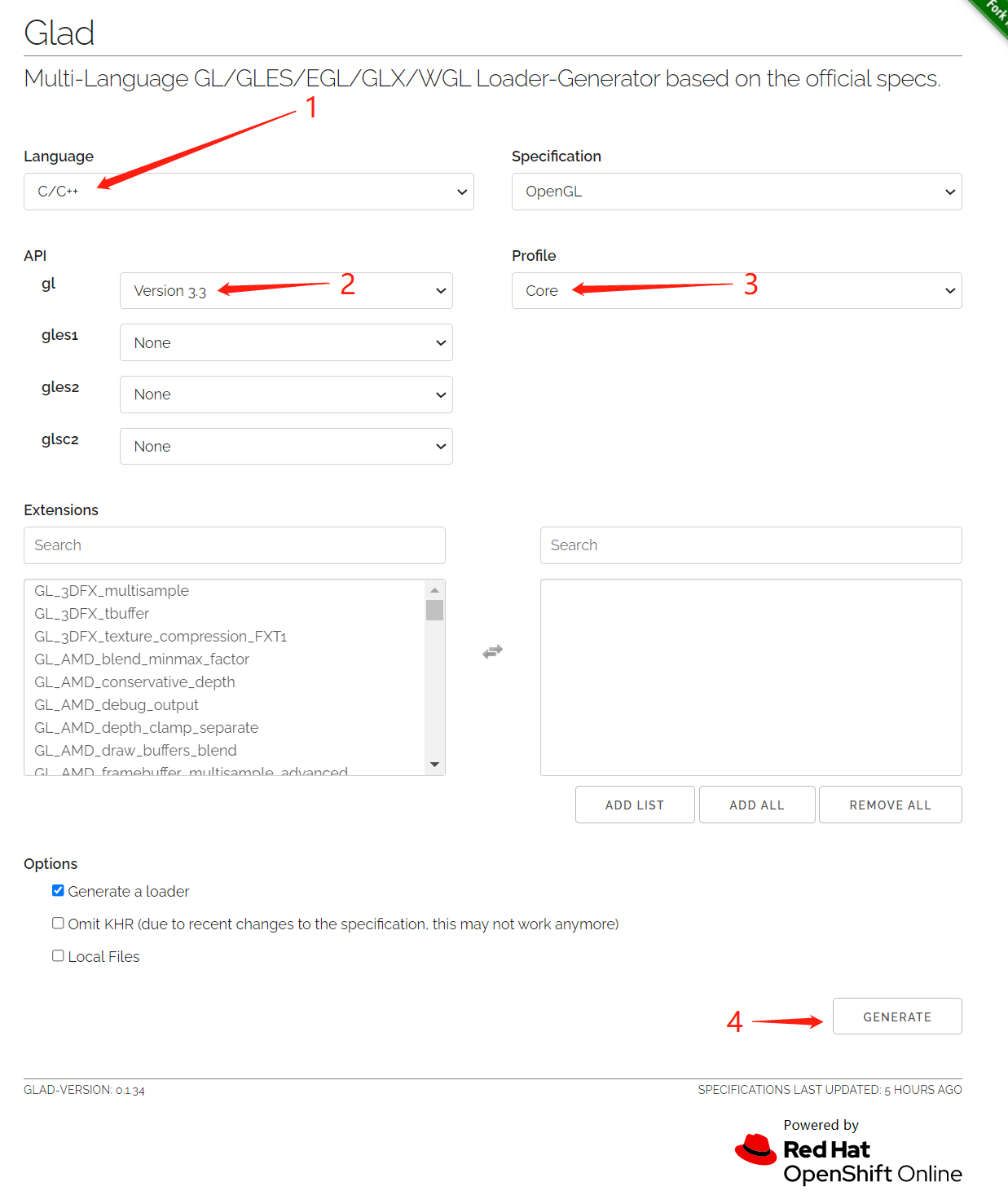
（5）依赖库：GLFW、GLAD、GLM的安装

GLFW和GLM可以很方便的通过homebrew安装，安装命令为：brew install glm glfw3。

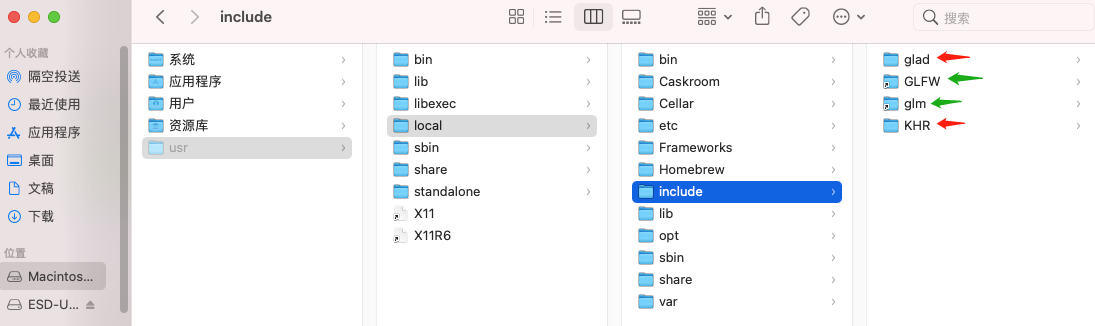


遗憾的是，我们无法采用同样简单的方法安装GLAD库，因为它在homebrew的源中不存在，所以我们只能手动来安装配置GLAD，但过程不会很复杂。

a) 访问GLAD网站下载源文件。网址为<https://glad.dav1d.de/>。进去之后，选择语言为C/C++, API为Version 3.3，核心模式，之后点击生成。



b) 下载生成的zip文件夹并解压，将子文件夹include中的内容移动到电脑/usr/local/include文件夹中。



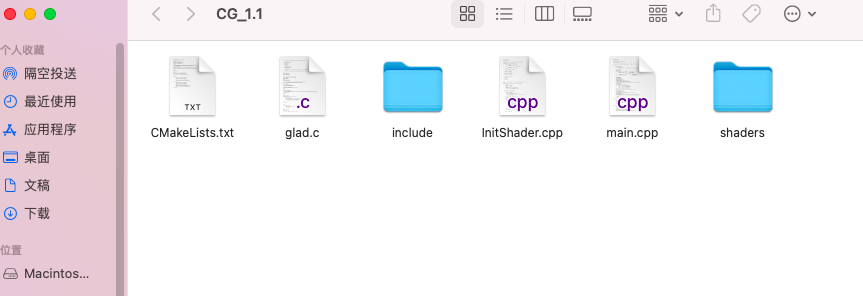
红色箭头所指的为需要新移动的文件夹，绿色箭头所指的是在上一步安装过程中安装的文件夹。

至此，我们需要的所有的安装环境都已经搭建好了，接下来将开始代码的编译与运行。

1. **运行代码**

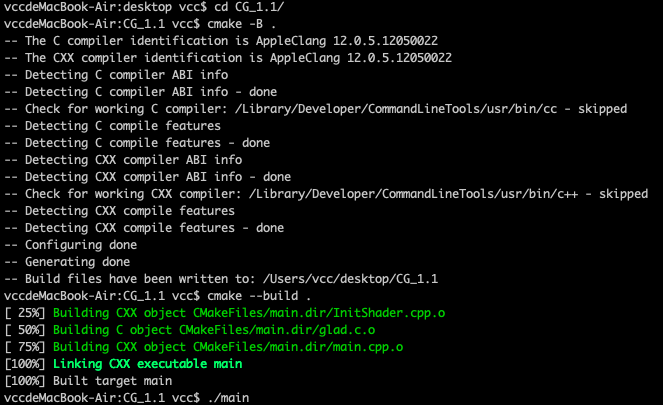
（1）方法1：通过Cmake构建运行实验1.1

新建项目文件夹，解压提供的参考代码，把内容拷贝到项目文件夹中。

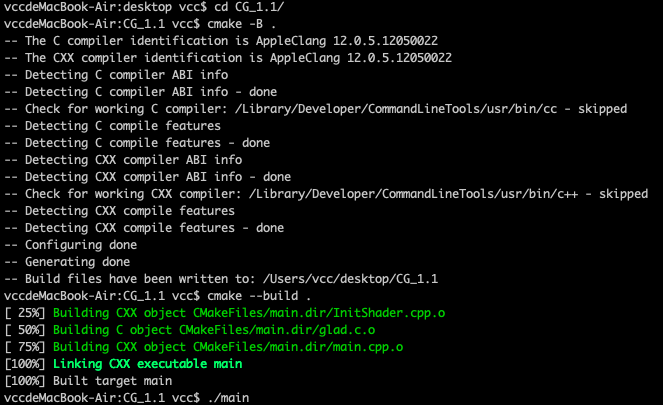


打开终端，进入项目文件夹，使用cmake构建、编译项目（注意参数“.”表示当前所在路径）：

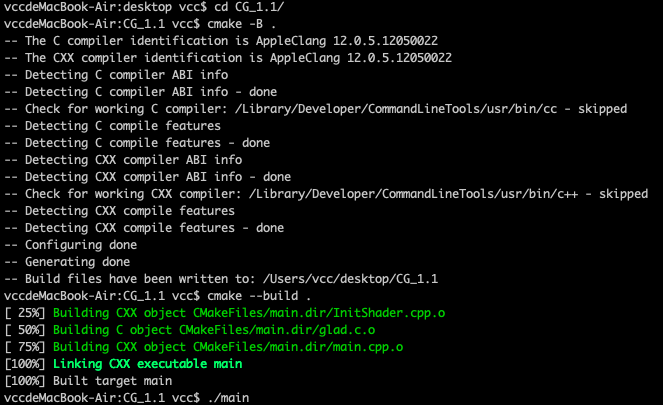
运行cmake -B . 。



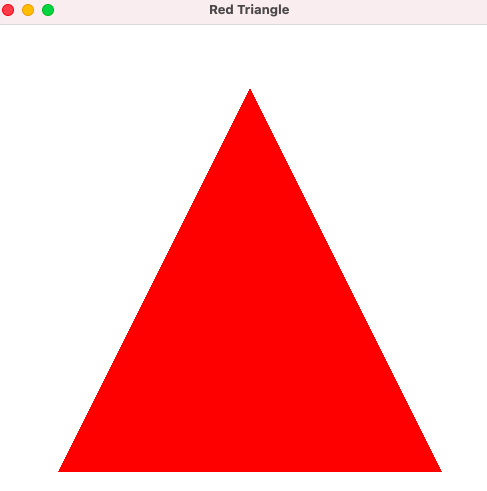
运行cmake --build . 。



运行生成的可执行文件：./main 。

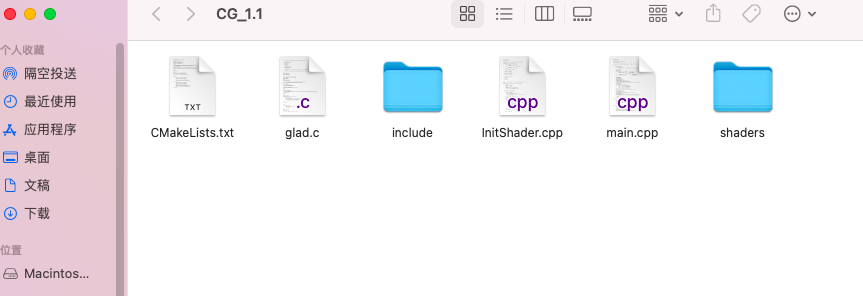


顺利的话会出现如下所示的三角形。

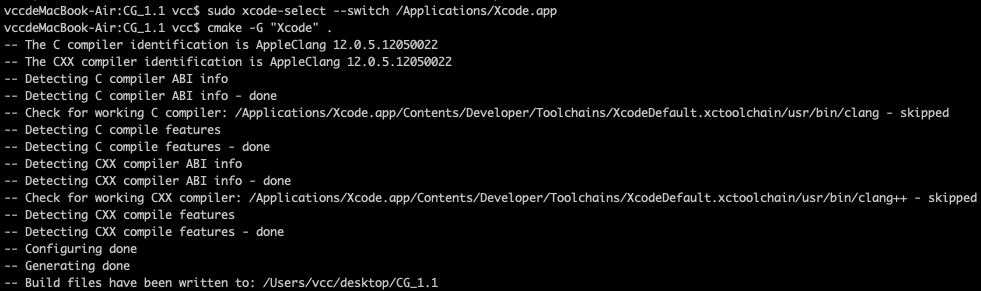


（2）方法2：通过CMake创建Xcode工程，在Xcode中编译运行实验1.1

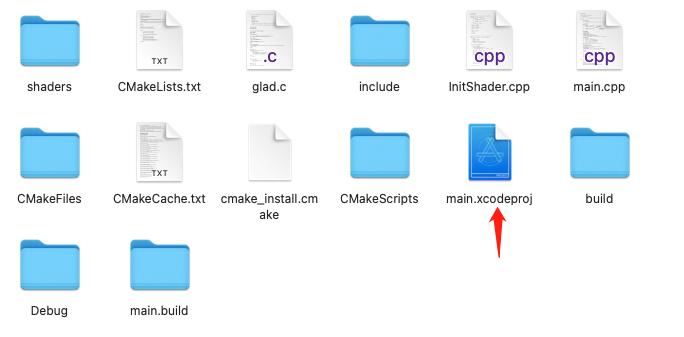
需新建项目文件夹，解压提供的参考代码，把内容拷贝到项目文件夹中。



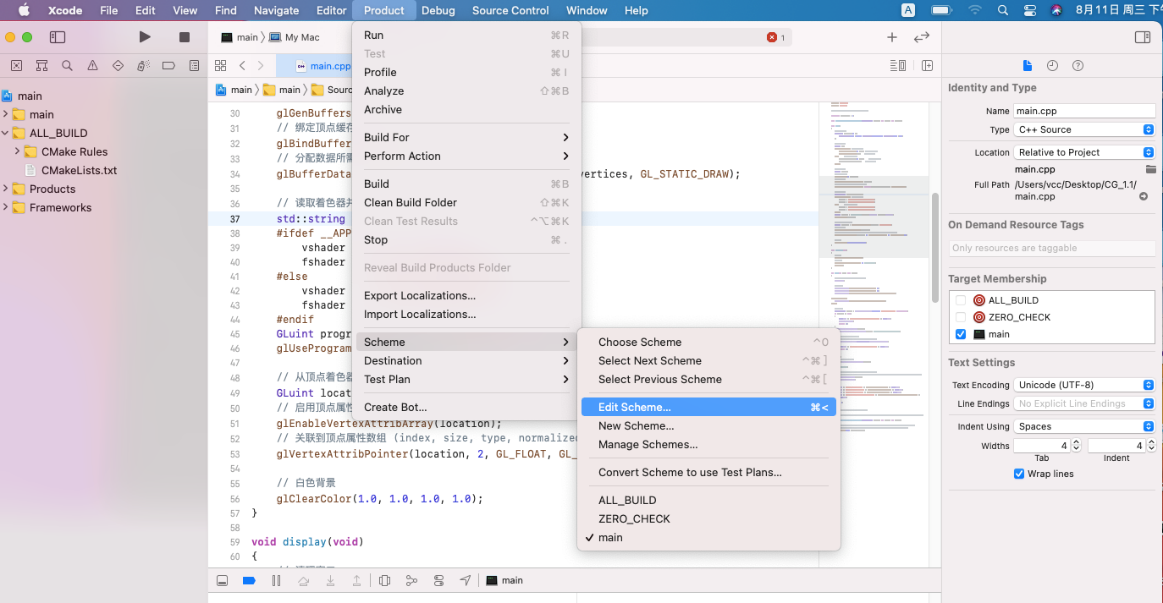
打开终端，进入项目文件夹，使用cmake构建Xcode项目（注意参数“.”表示当前所在路径），命令如下：cmake -G “Xcode” . 。如果出现未找到编译器类型的错误（No CMAKE\_C\_COMPILER could be found），可以先执行以下命令进行切换：sudo xcode-select --switch /Applications/Xcode.app/ 。

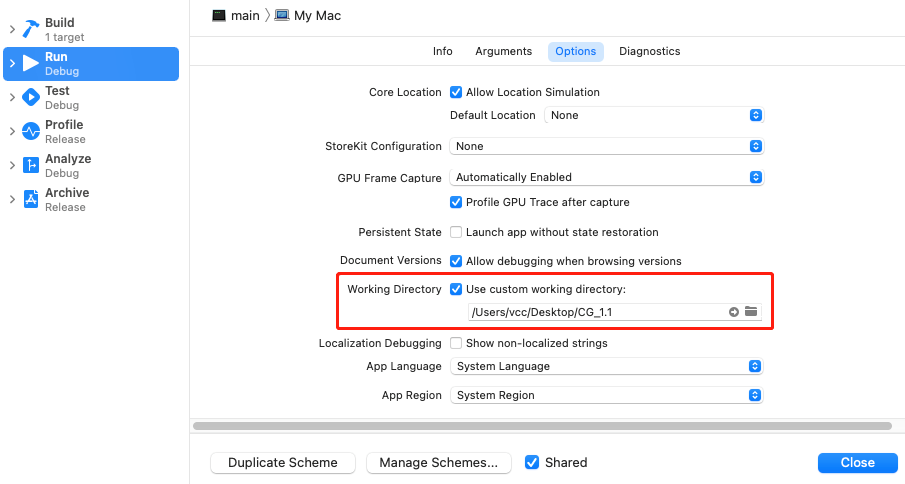


命令将会生成一个cmake工程文件main.xcodeproj。

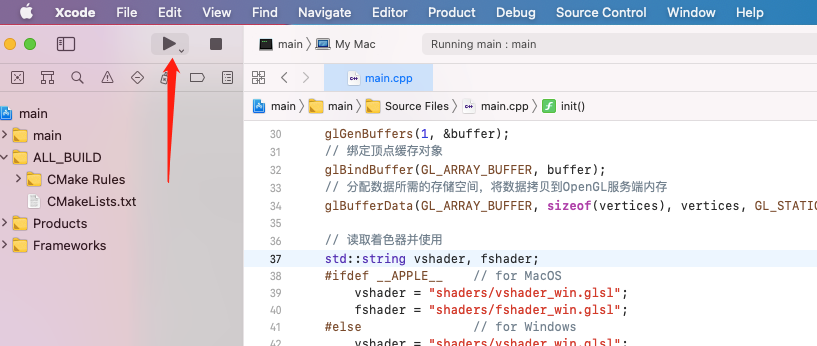


打开main.xcodeproj，点击Edit Scheme修改项目运行路径，指定为当前路径，即该项目文件夹的路径。





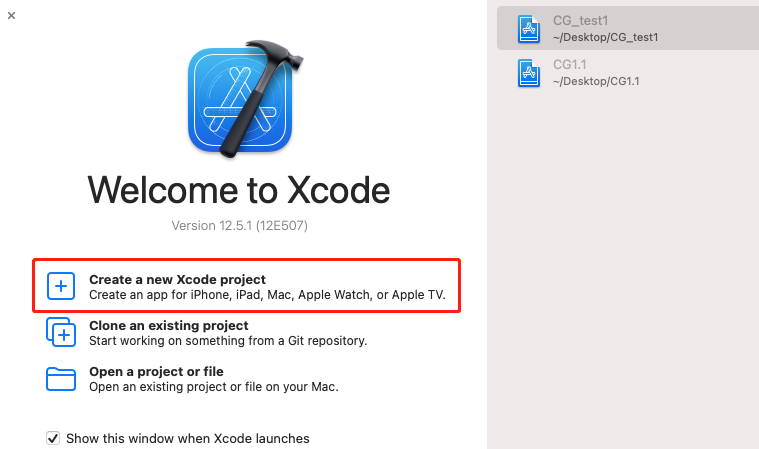
配置完成，点击左上角的播放键（快捷键Command+R），即可编译运行。



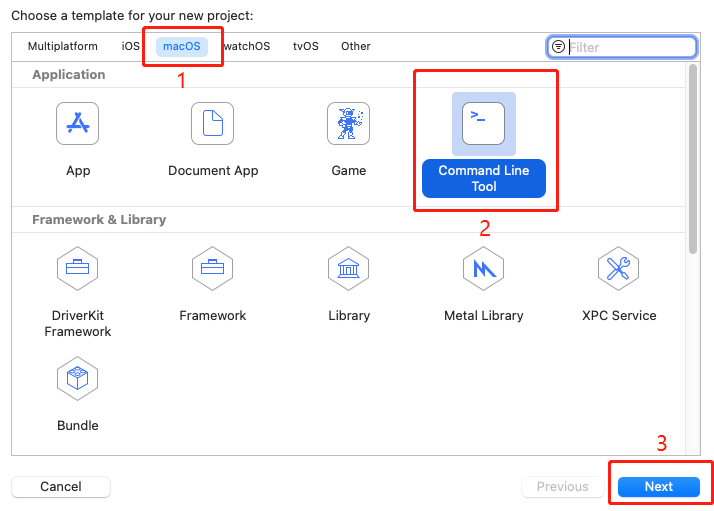
(3) 方法3：通过Xcode创建工程运行实验1.1

方法3相对来说配置最为复杂，但是在实际开发的过程中可以随时debug。

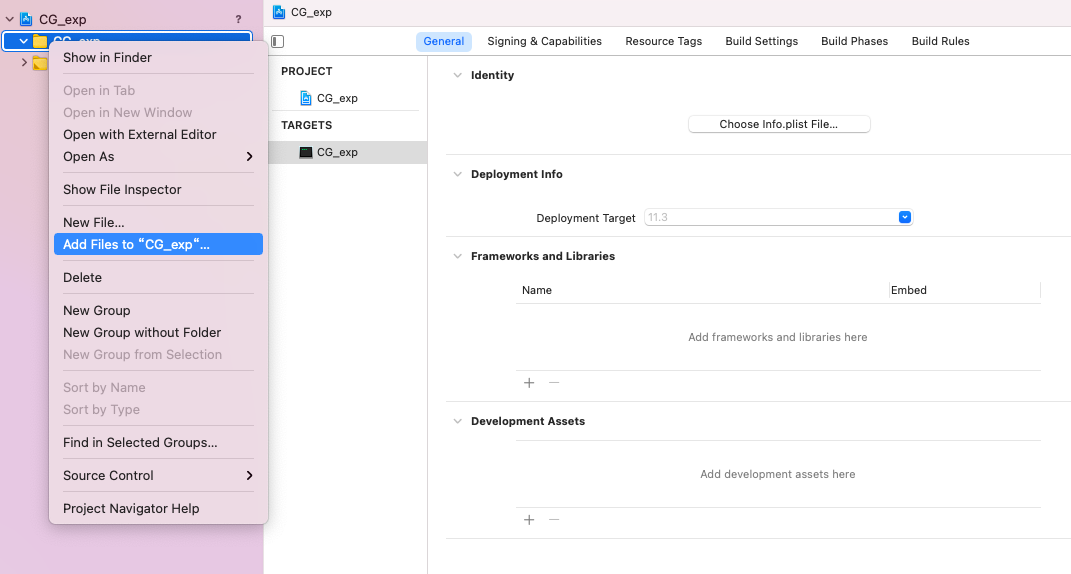
打开Xcode，创建一个新的Xcode project。

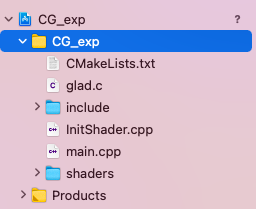


选择Command Line Tool.

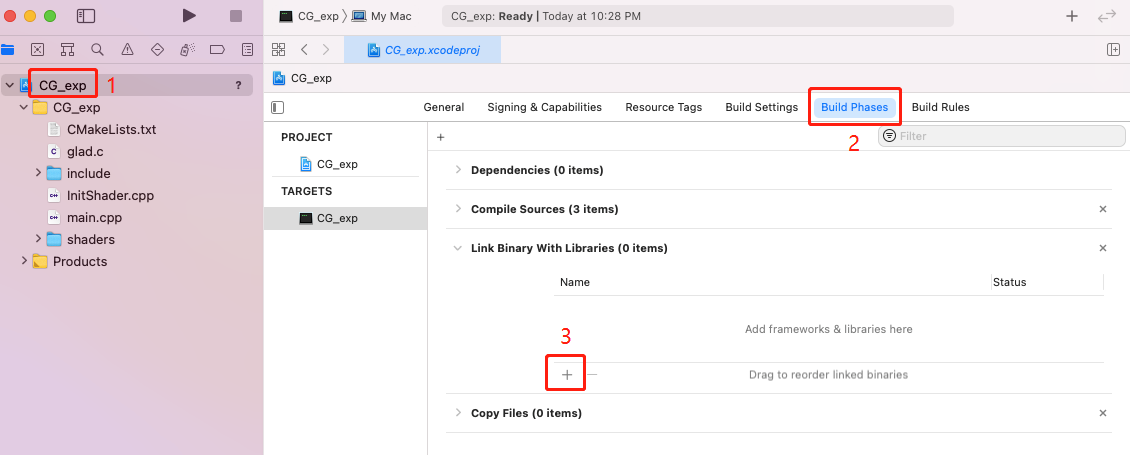


给工程命名并选择存放目录，Xcode会为新建的工程项目创建一个main.cpp文件，是一个Helloworld，删除即可。接下来解压提供的参考代码，放入工程目录中。

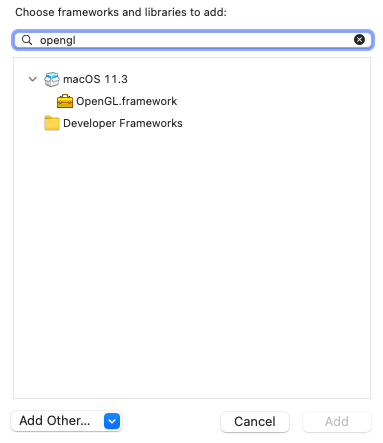




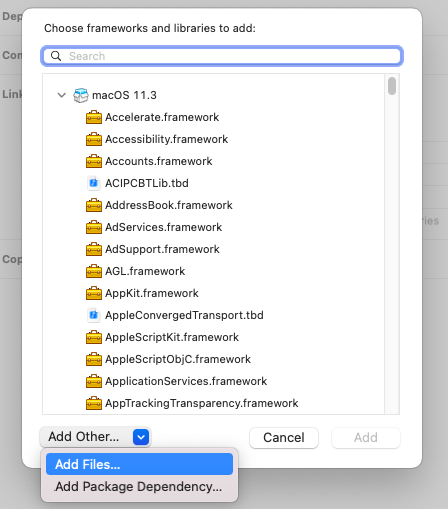
接下来添加库连接，进入工程设置界面，选择build phases选项卡，进入Link Binary Libraries，点击加号。



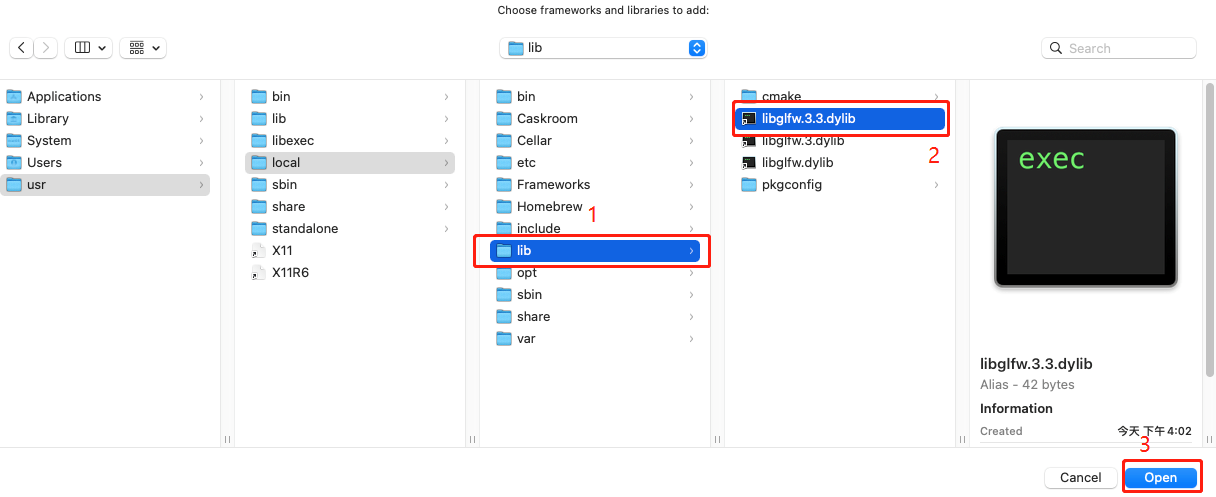
搜索OpenGL并添加。



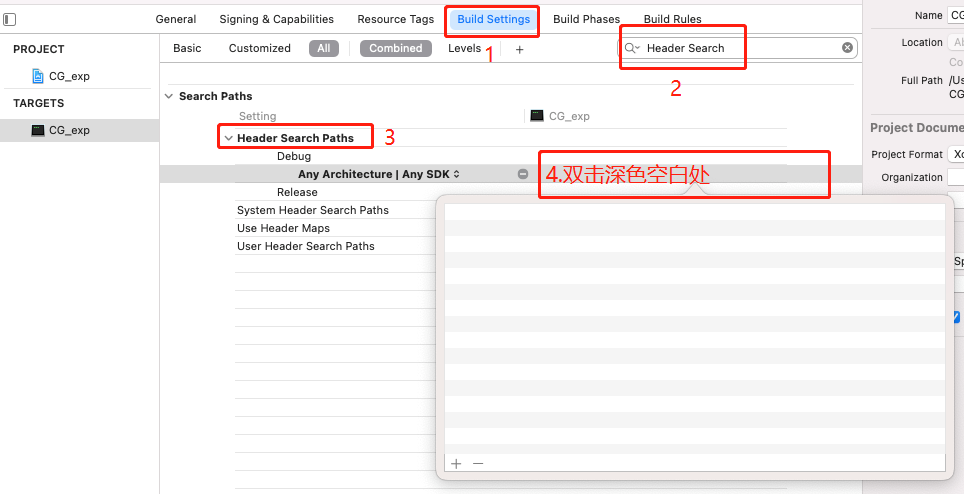
添加glfw动态链接库，再次点击上个界面的加号，选择添加文件。



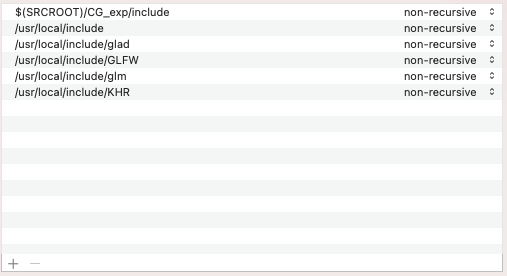
按“command”+“shift”+“G”快速进入‘/usr/local’文件夹，将lib文件夹中的libglfw.3.3.dylib 添加进项目。



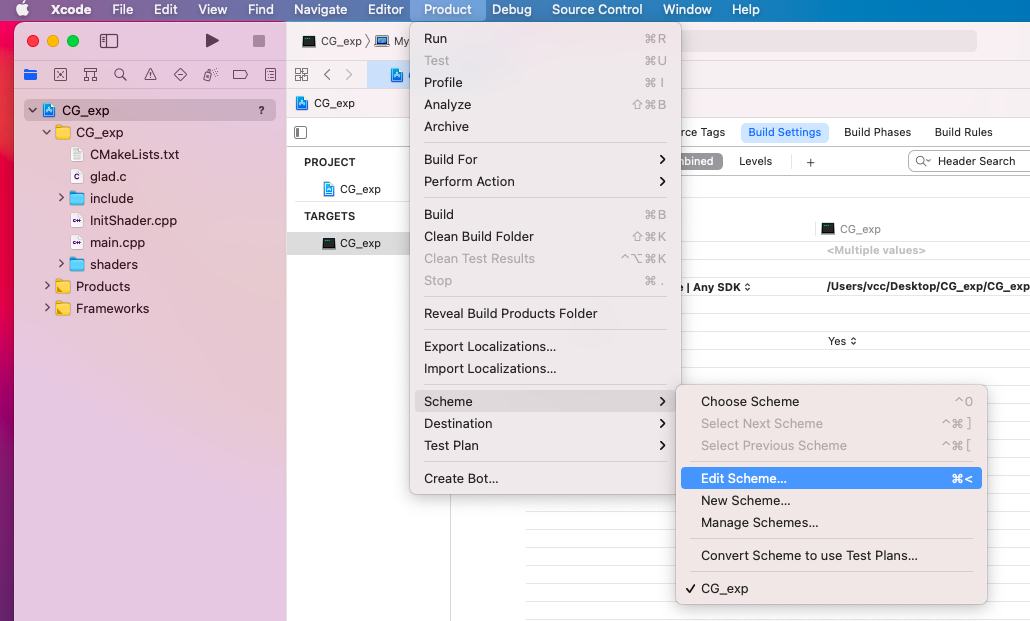
添加头文件搜索路径，进入前文提到的工程设置界面，选择Build Settings选项卡，通过搜索栏找到Header Search Paths .

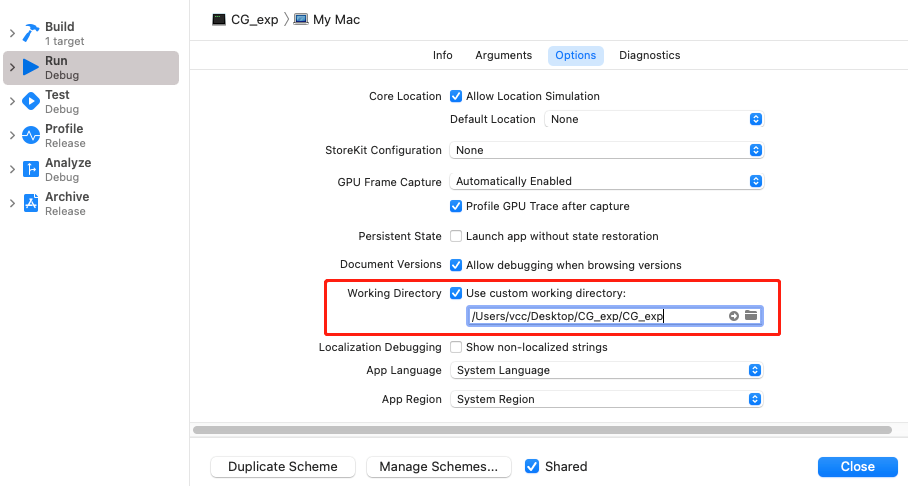


添加头文件搜索路径 $(SRCROOT)/{自己的工程名称}/include；/usr/local/include; /usr/local/include/glad; /usr/local/include/GLFW; ; /usr/local/include/glm; /usr/local/include/KHR.



点击Product-Scheme-Edit Scheme，设置工作路径到工程文件夹下。





配置完成，点击编译运行。

