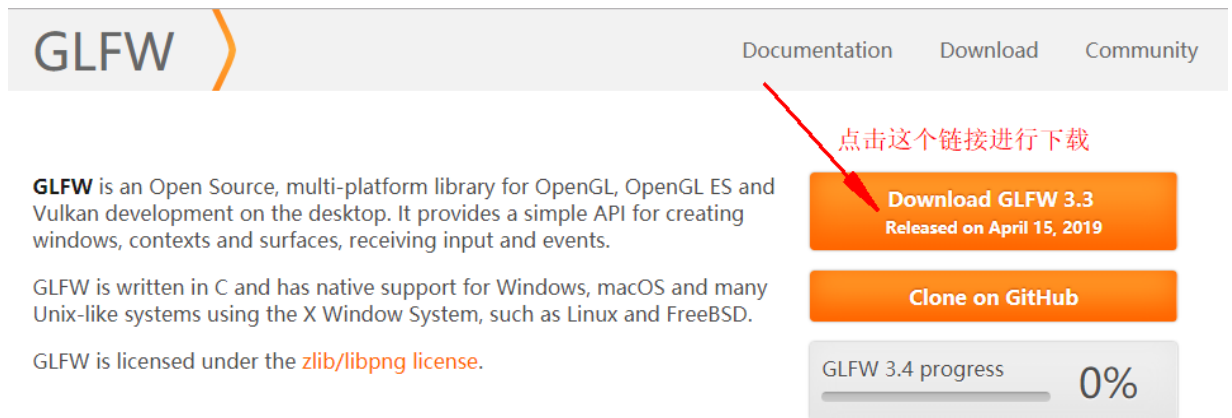


# 构建 GLFW

第一步：在官网上下载 GLFW（网址：<https://www.glfw.org/>）



注意：为了完整性我们将从编译源代码开始，所以我们需要下载**源代码包**。并且如果你要使用预编译的二进制版本的话，请下载**32 位的版本**而不是 64 位的，因为 64 位版本会出现很多奇怪的问题。

从源代码编译库可以保证生成的库是兼容你的操作系统和 CPU 的，而预编译的二进制文件可能会出现兼容问题（甚至有时候没提供支持你系统的文件）。提供源代码所产生的一个问题在于不是每个人都用相同的 IDE 开发程序，因而提供的工程/解决方案文件可能和一些人的 IDE 不兼容。所以人们只能从 .c/.cpp 和 .h/.hpp 文件来自己建立工程/解决方案，这是一项枯燥的工作。但因此也诞生了一个叫做 CMake 的工具。

第二步：下载 CMake（网址：<https://cmake.org/download/>）

 <span>About</span> <span>Resources</span> <span>Developer Resources</span> <span>Download</span>	
Platform	Files
Unix/Linux Source (has <code>\n</code> line feeds)	<a href="#">cmake-3.14.3.tar.gz</a>
	<a href="#">cmake-3.14.3.tar.Z</a>
Windows Source (has <code>\r\n</code> line feeds)	<a href="#">cmake-3.14.3.zip</a>

Binary distributions:

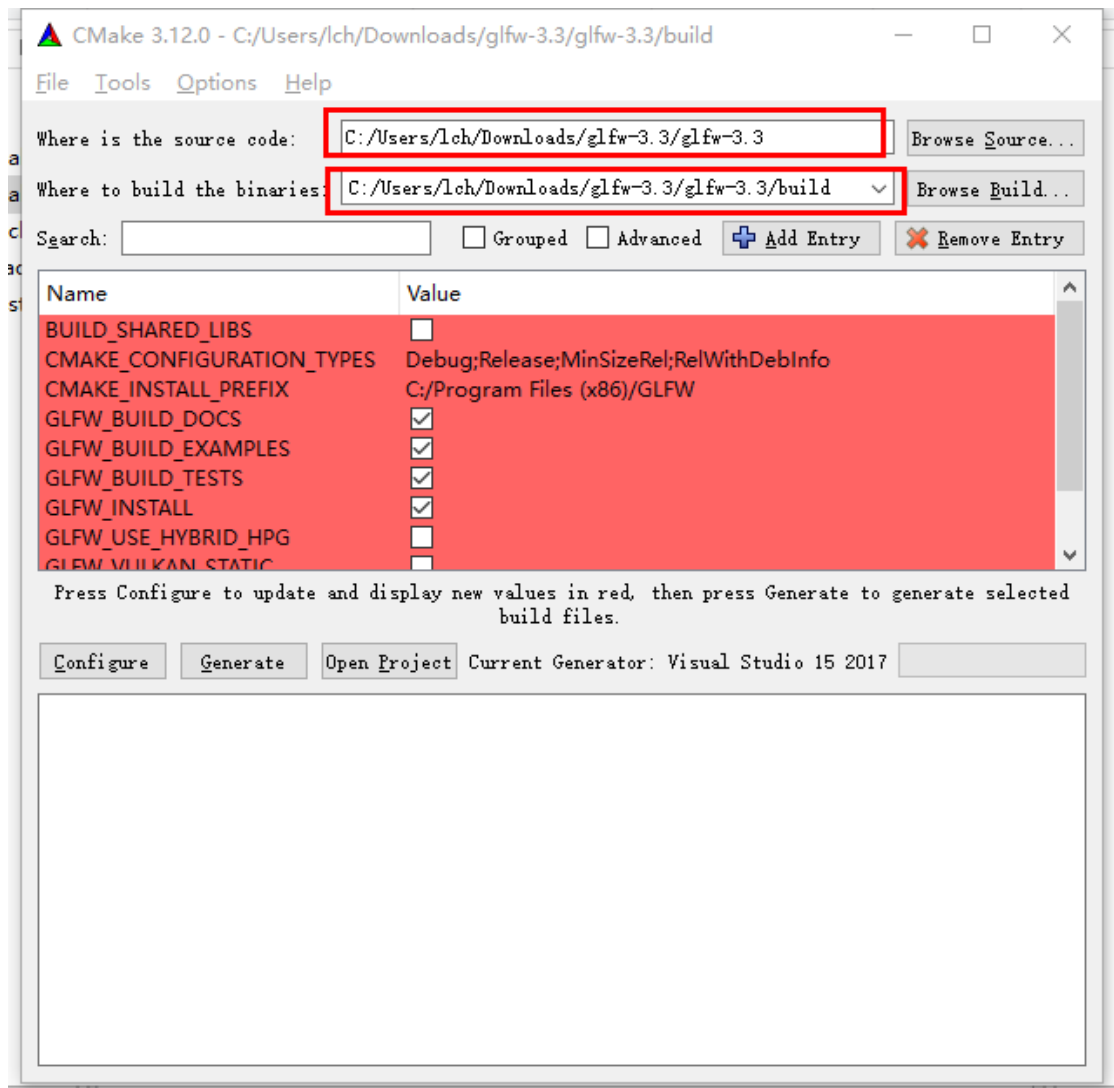
Platform	Files
Windows win64-x64 Installer: <b>Installer tool has changed. Uninstall CMake 3.4 or lower first!</b>	<a href="#">cmake-3.14.3-win64-x64.msi</a>
Windows win64-x64 ZIP	<a href="#">cmake-3.14.3-win64-x64.zip</a>
Windows win32-x86 Installer: <b>Installer tool has changed. Uninstall CMake 3.4 or lower first!</b>	<a href="#">cmake-3.14.3-win32-x86.msi</a>
Windows win32-x86 ZIP	<a href="#">cmake-3.14.3-win32-x86.zip</a>
Mac OS X 10.7 or later	<a href="#">cmake-3.14.3-Darwin-x86_64.dmg</a>
	<a href="#">cmake-3.14.3-Darwin-x86_64.tar.gz</a>
Linux x86_64	<a href="#">cmake-3.14.3-Linux-x86_64.sh</a>
	<a href="#">cmake-3.14.3-Linux-x86_64.tar.gz</a>

第三步：解压后，打开文件 bin 目录下的 `cmake-gui.exe`，启动 CMake。

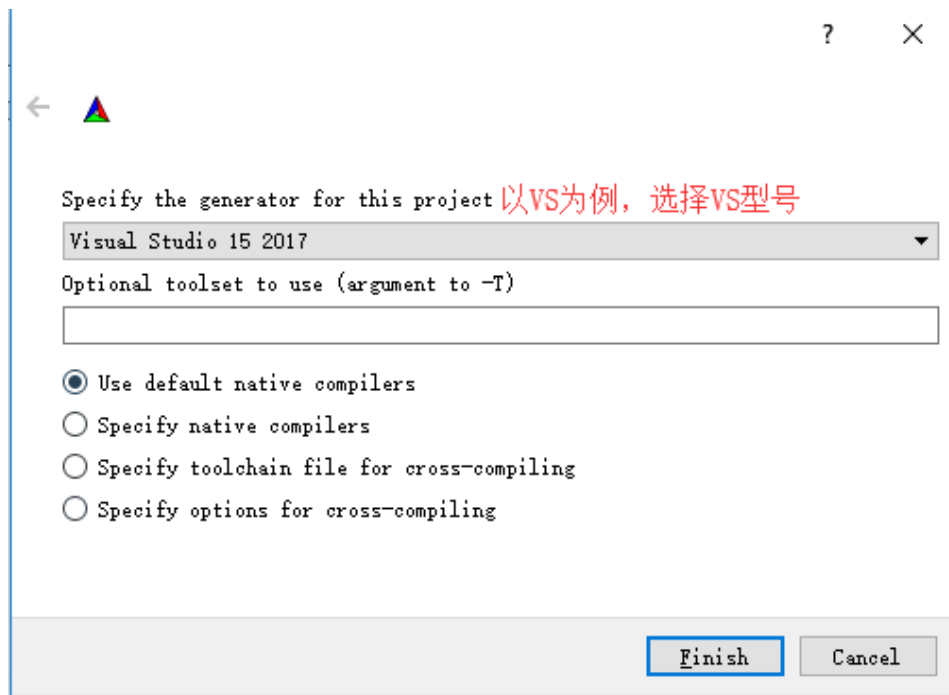
名称	修改日期	类型	大小
 <code>cmake.exe</code>	2018/5/17 10:24	应用程序	7,365 KB
 <code>cmake-gui.exe</code>	2018/5/17 10:24	应用程序	17,979 KB
 <code>cmcldeps.exe</code>	2018/5/17 10:23	应用程序	792 KB
 <code>cpack.exe</code>	2018/5/17 10:24	应用程序	7,071 KB
 <code>ctest.exe</code>	2018/5/17 10:24	应用程序	7,814 KB

第四步：CMake 需要一个源代码目录和一个存放编译结果的目标文件目录。

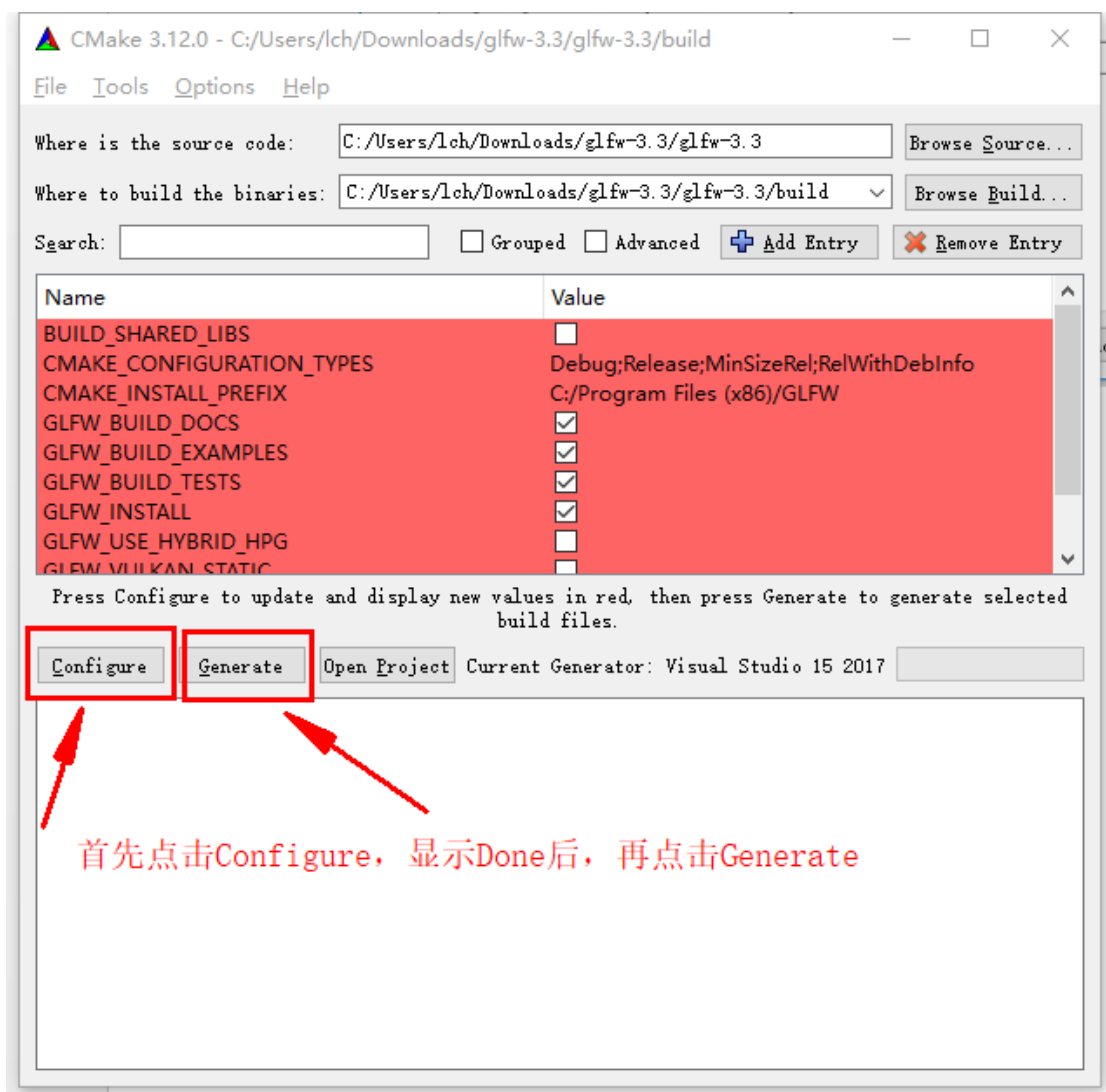
源代码目录我们选择 GLFW 的源代码的根目录，然后我们新建一个 `build` 文件夹，选中作为目标目录。




**第五步：**在设置完源代码目录和目标目录之后，点击 Configure(设置)按钮，我们接下来需要选择工程的生成器，由于我们使用的是 Visual Studio 2017，我们选择 Visual Studio 15 选项（因为 Visual Studio 2017 的内部版本号是 15）。



**第六步：**设置完成后，再次点击 Configure(设置)按钮保存设置。保存之后，点击 Generate(生成)按钮，生成的工程文件会在你之前创建的 build 文件夹中。



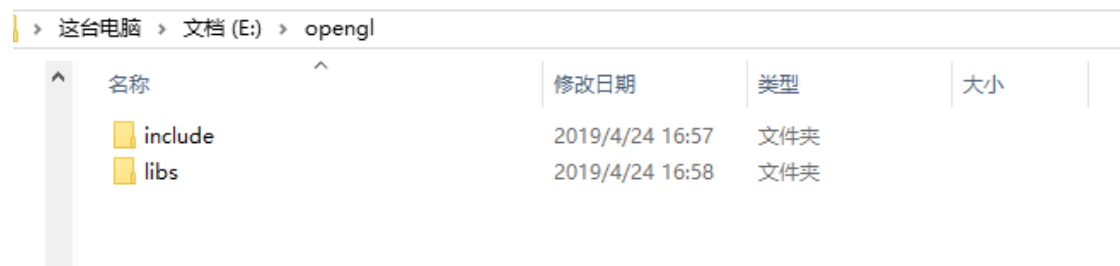
**第七步：**在 build 文件夹里可以找到 GLFW.sln 文件，用 Visual Studio 2017 打开。因为 CMake 已经配置好了项目，所以我们直接点击 Build Solution(生成解决方案)按钮，然后编译的库 glfw3.lib（注意我们用的是第 3 版）就会出现在 src/Debug 文件夹内。

glfw-3.2.1 > glfw-3.2.1 > build > src > Debug			
名称	修改日期	类型	大小
 glfw3.lib	2018/8/22 16:59	对象文件库	576 KB

**第八步：**库生成完毕之后，我们需要让 IDE 知道库和头文件的位置。**推荐的方式是：**建立一个新的目录包含所有的第三方库文件和头文件，并且在你的 IDE 或编译器中指定这些文件夹。我个人会使用一个单独的文件夹，里面包含 Libs 和 Include 文件夹，在

这里存放 OpenGL 工程用到的所有第三方库和头文件。这样我的所有第三方库都在同一个位置(并且可以共享至多台电脑)。然而这要求你每次新建一个工程时都需要告诉 IDE/编译器在哪能找到这些目录。

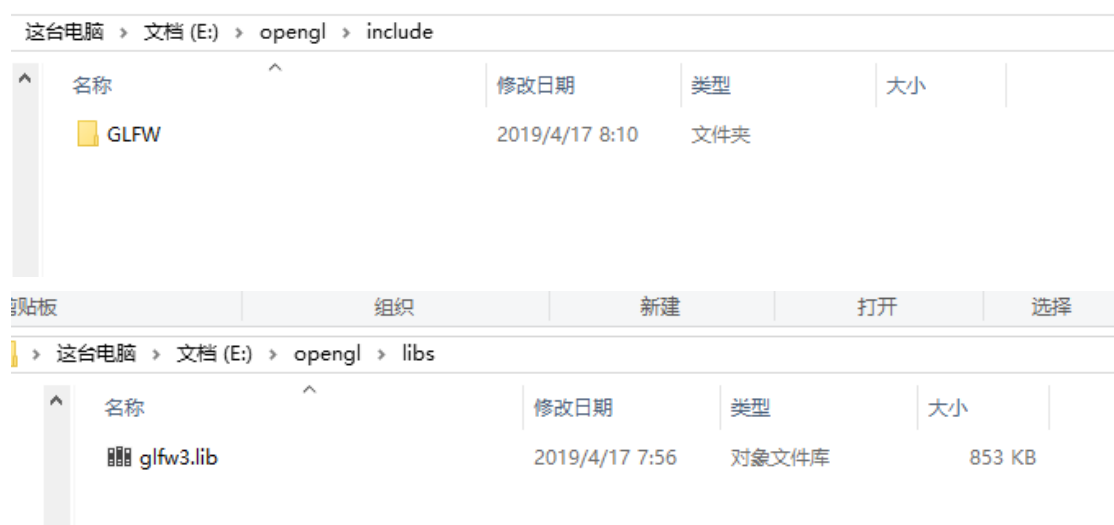
因此可以新建一个单独的文件夹 opengl, 里面再新建一个 libs 和 include 文件夹。在这里存放 OpenGL 工程用到的所有第三方库和头文件, 新建项目的时候直接在编译器中链接这个目录即可。



将刚刚编译好的 glfw3.lib 文件放在 libs 文件夹下, 将之前下载的 glfw 文件中 include 文件夹下的 GLFW 文件复制粘贴到新建的 opengl 文件夹下的 include 文件夹中。



最后 opengl 文件夹中应该是这样:

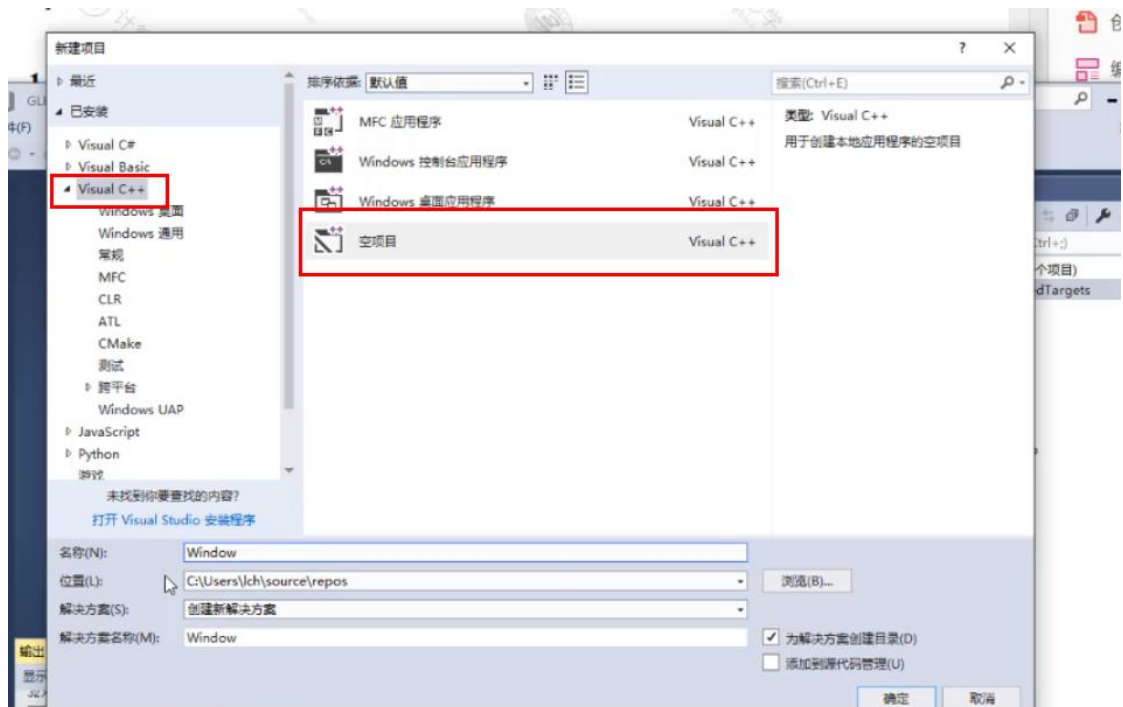


完成上面步骤后，我们就可以使用 GLFW 创建我们的第一个 OpenGL 工程了！

## 我们的第一个工程

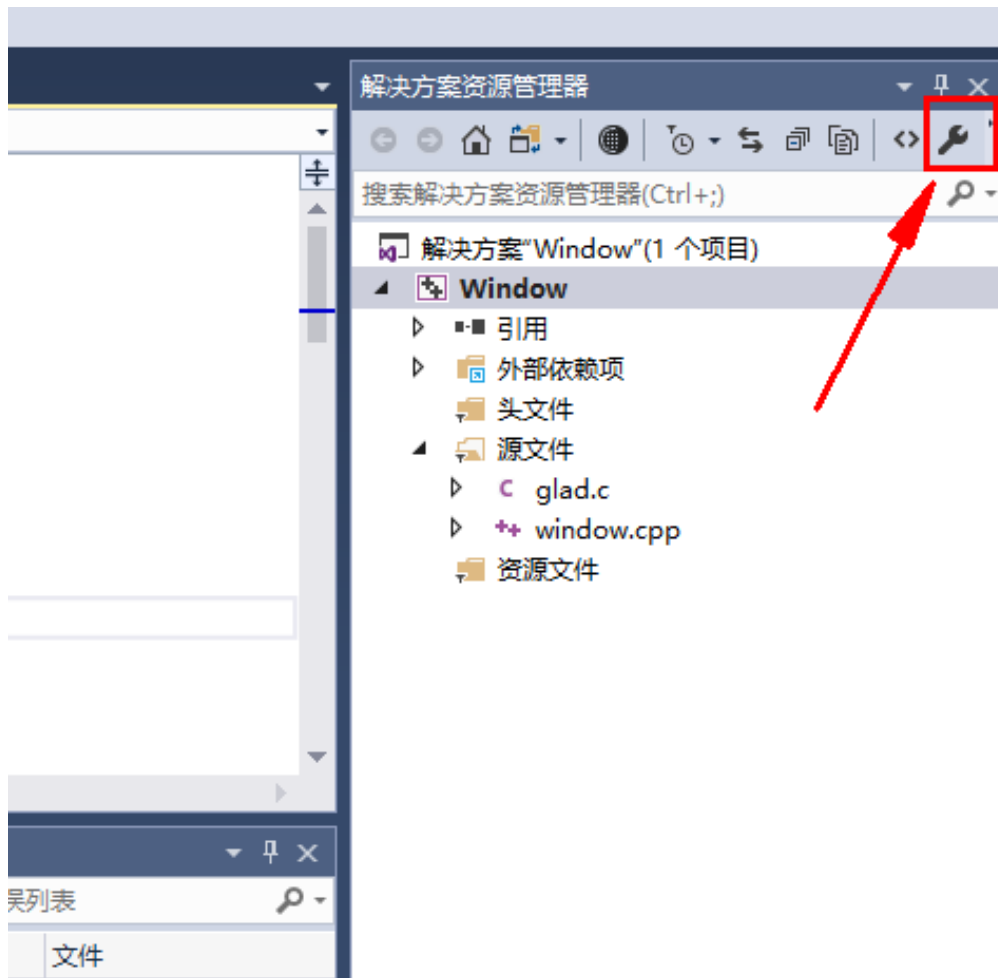
### 第一步：新建一个 Visual C++ 的空项目

首先，打开 Visual Studio，创建一个新的项目。如果 VS 提供了多个选项，选择 Visual C++，然后选择 Empty Project（空项目）（别忘了给你的项目起一个合适的名字）。现在我们终于有一个空的工作空间了，开始创建我们第一个 OpenGL 程序吧！



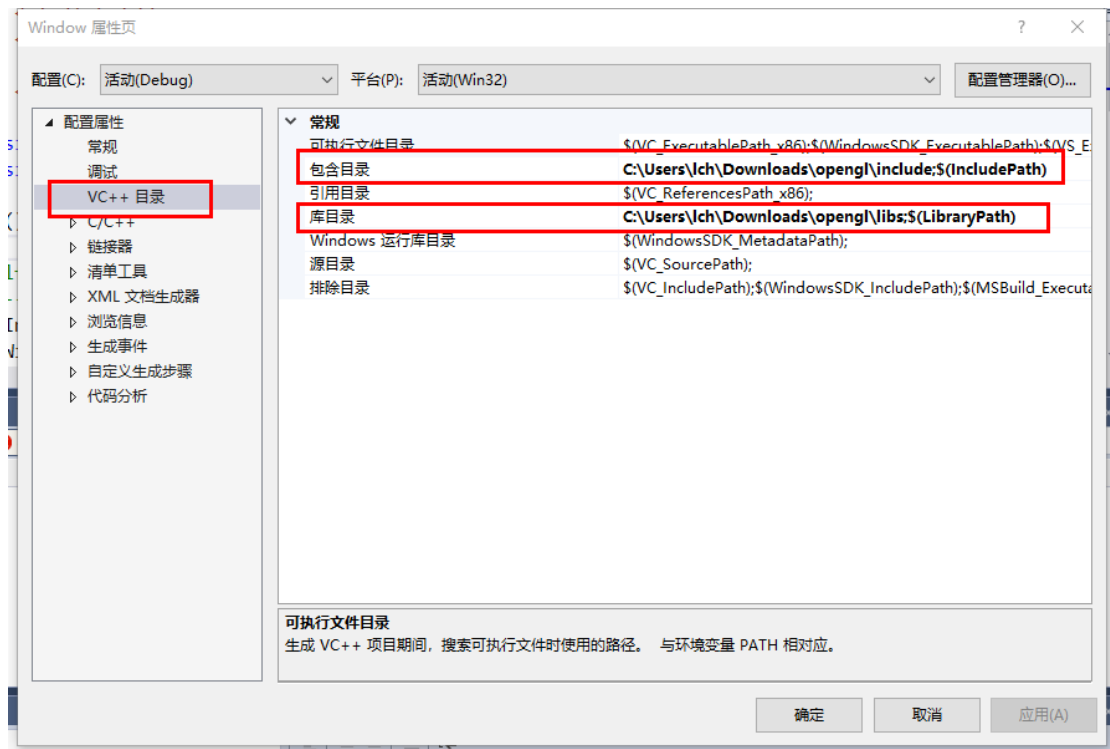
### 第二步：将 GLFW 库链接进工程

(1) 选中 Window 项目，然后点击右上角的工具按钮，进入该工程的属性页。



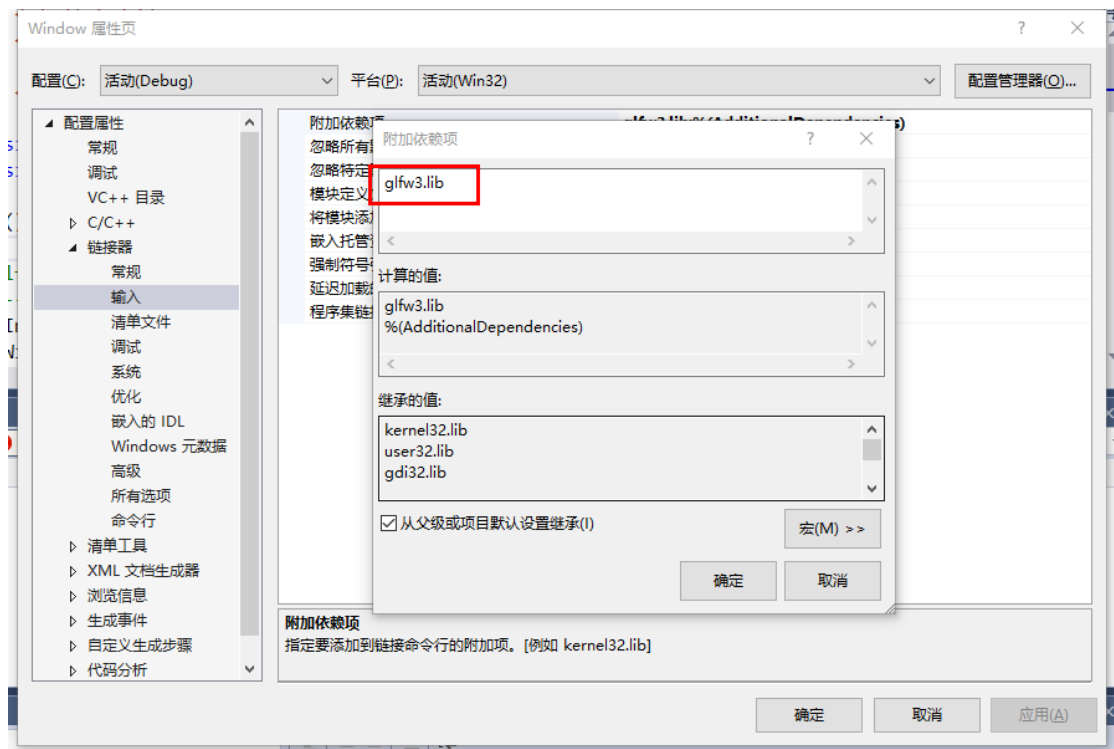
(2) 然后选择 VC++ 目录选项卡 (如下图), 将之前我们创建的 opengl 文件夹下的 include 和 libs 目录都链接进工程中来。在下面的两栏添加目录:

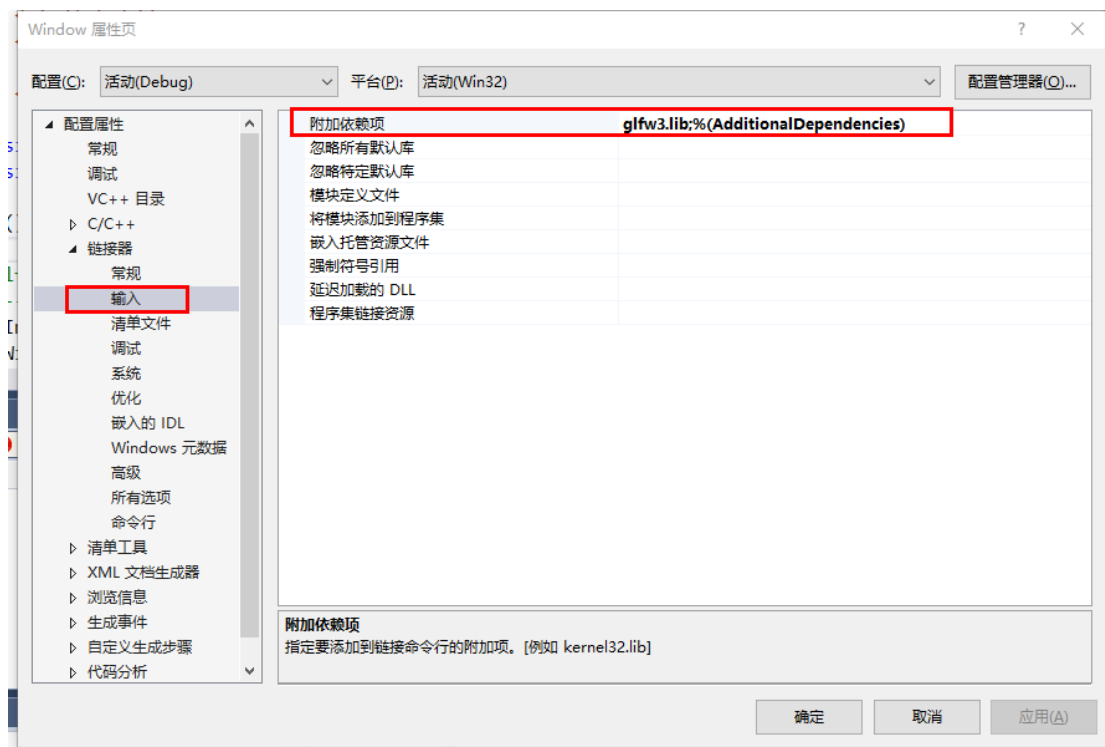




所以只要你将 GLFW 的 Include 文件夹加进路径中, 你就可以使用<GLFW/.>来引用头文件。库目录也是同样。

(3) 最后需要在 Linker(链接器)选项卡里的 Input(输入)选项卡里添加 glfw3.lib 这个文件:





要链接一个库我们必须告诉链接器它的文件名。库名字是 **glfw3.lib**，我们把它加到附加依赖项字段中，这样 GLFW 在编译的时候就会被链接进来了。

## 配置 GLAD 库

**第一步：**对 GLAD 进行在线配置 (<https://glad.dav1d.de/> )

将语言(Language)设置为 **C/C++**，在 API 选项中，选择 **3.3** 以上的 OpenGL(gl) 版本（我们的教程中将使用 3.3 版本，但更新的版本也能正常工作）。之后将模式(Profile)设置为 **Core**，并且保证**生成加载器**(Generate a loader)的选项是选中的。现在可以先（暂时）忽略拓展(Extensions)中的内容。都选择完之后，点击**生成**(Generate)按钮来生成库文件。

Language	C/C++	Specification	OpenGL
API	gl Version 3.3	Profile	Core
gles1	None		
gles2	None		
glsc2	None		

GL\_3DFX\_tbuffer  
GL\_3DFX\_texture\_compression\_FXT1  
GL\_AMD\_blend\_minmax\_factor  
GL\_AMD\_conservative\_depth  
GL\_AMD\_debug\_output  
GL\_AMD\_depth\_clamp\_separate  
GL\_AMD\_draw\_buffers\_blend

ADD LIST

ADD ALL

REMOVE ALL

Options

☒ Generate a loader

☐ Omit KHR (due to recent changes to the specification, this may not work anymore)

☐ Local Files

GENERATE

点击这个压缩包进行下载

Glad

Generated files. These files are not permanent!

Name ^	Last modified	Size
<div>include</div>	2019-04-29 12:04:46.095070	-
<div>src</div>	2019-04-29 12:04:46.095070	-
<div>glad.zip</div>	2019-04-29 12:04:46.159068	179.5 kB

Permalink:

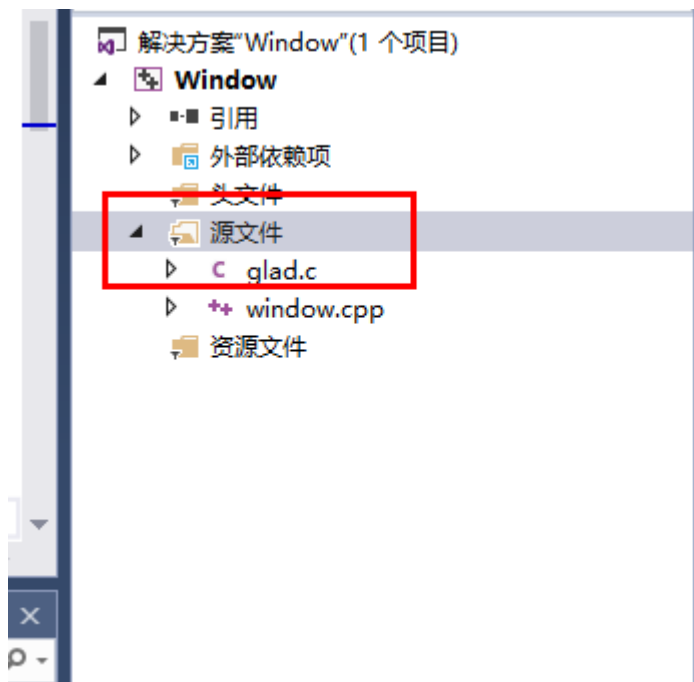
http://glad.dav1d.de/#profile=core&specification=gl&api=gl%3D3.3&api=gles1%3Dnone&api=gles2%3Dnone&api=glsc2%3Dnone&language

**第二步:** 将需要的文件移动到之前创建的 opengl 文件夹下。压缩包下载解压后，有 include 和 src 这两个文件。

(1) 将 include 文件夹下的 glad 和 KHR 文件夹复制粘贴到 opengl 文件夹下的 include 文件夹中。



(2) 将 src 文件夹下的 glad.c, 通过在 VS 项目中 源文件 → 添加 → 新建项 来将该文件引入工程中。



将以下代码添加到你的 .cpp 文件中:

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
```

```

#include <iostream>

const unsigned int SCR_WIDTH = 800;
const unsigned int SCR_HEIGHT = 600;

int main()
{
    // glfw: initialize and configure
    // -----
    glfwInit();
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

    GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "OpenGL", NULL,
    NULL);
    if (window == NULL)
    {
        std::cout << "Failed to create GLFW window" << std::endl;
        glfwTerminate();
        return -1;
    }
    glfwMakeContextCurrent(window);

    // glad: load all OpenGL function pointers
    // -----
    if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
    {
        std::cout << "Failed to initialize GLAD" << std::endl;
        return -1;
    }

    // render loop
    // -----
    while (!glfwWindowShouldClose(window))
    {

        glfwSwapBuffers(window);
        glfwPollEvents();
    }
}

```

```
}

glfwTerminate();

return 0;

}
```

点击运行后，生成一个 opengl 窗口，就表示配置成功。

