

Windows下OpenGL安装与运行

1. 安装windows下c++编译工具：MinGW、cmake
2. 安装OpenGL库：glfw、glad
3. 编写cpp程序
4. 编译链接
5. 运行

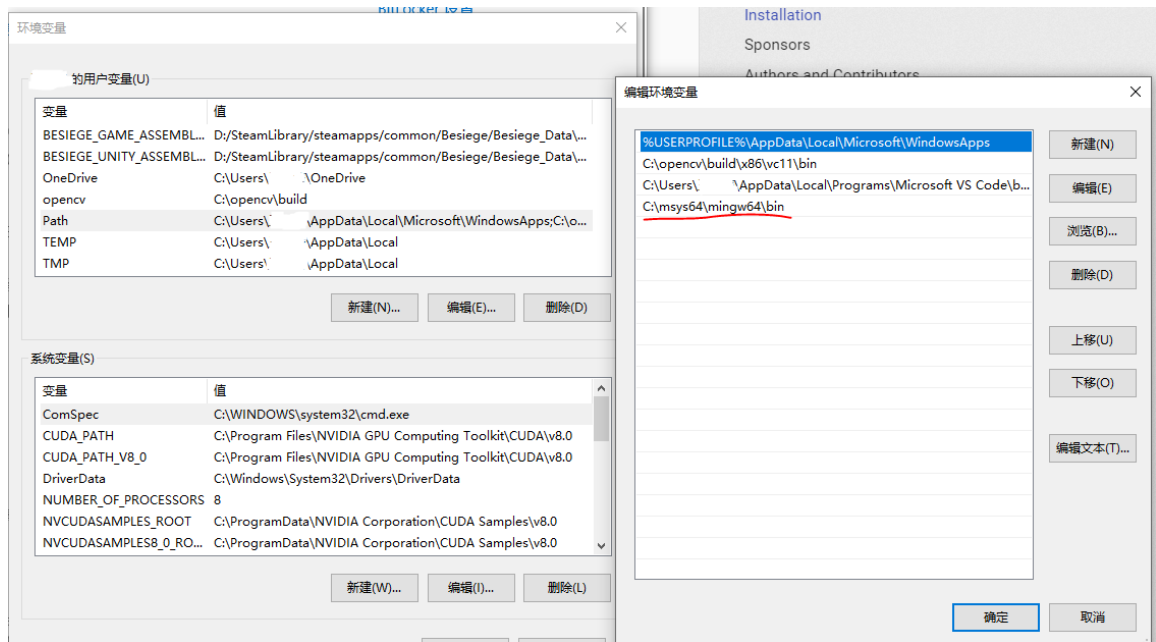
【参考】

- opengl安装编译：
<https://blog.csdn.net/linshuhe1/article/details/93976706>
- vscode下c++运行调试：
<https://code.visualstudio.com/docs/languages/cpp>

【代码链接】

1. 安装windows下c++编译工具：MinGW、cmake

- 安装MinGW：<https://www.msys2.org/>
按照步骤安装即可
注意修改环境变量



- 安装cmake：<https://cmake.org/download/>

Binary distributions:

Platform	Files
Windows x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.22.0-windows-x86_64.msi
Windows x64 ZIP	cmake-3.22.0-windows-x86_64.zip
Windows i386 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.22.0-windows-i386.msi
Windows i386 ZIP	cmake-3.22.0-windows-i386.zip
macOS 10.13 or later	cmake-3.22.0-macos-universal.dmg
	cmake-3.22.0-macos-universal.tar.gz
macOS 10.10 or later	cmake-3.22.0-macos10.10-universal.dmg
	cmake-3.22.0-macos10.10-universal.tar.gz
Linux x86_64	cmake-3.22.0-linux-x86_64.sh
	cmake-3.22.0-linux-x86_64.tar.gz

下载之后按照步骤安装即可

2. 安装OpenGL库：glfw、glad

- glfw库支持窗口创建、读取输入、处理事件等功能
- 安装glfw：<https://www.glfw.org/download.html>

Download

The current version is **3.3.5**, which was released on **October 28, 2021**. See the [release notes](#) for details.

Source package

This package contains the complete source code with CMake build files, [documentation](#), examples and test programs. It is the recommended download for all platforms and offers the most control.

All development is done on GitHub. The `master` branch is our integration branch for the next feature release while the `3.3-stable` branch only adds bug fixes for patch releases.

[Source package](#)[GitHub repository](#)

Windows pre-compiled binaries

These packages contain the GLFW header files, [documentation](#) and release mode static libraries, DLLs and import libraries for Visual C++ 2010-2019 and the 2022 preview, MinGW-w64 and plain MinGW.

Binaries for Visual C++ 2010 and plain MinGW are only available in the 32-bit package.

[64-bit Windows binaries](#)[32-bit Windows binaries](#)

macOS pre-compiled binaries

This package contains the GLFW header files, [documentation](#) and release mode static and dynamic libraries for macOS 10.8 and later. Both Intel, ARM and Universal binaries are included.

[64-bit macOS binaries](#)

- glad的API包括：窗口操作、窗口初始化、窗口大小、位置调整等；回调函数；响应刷新消息、键盘消息、鼠标消息、定时器函数等；创建复杂三维体；菜单函数；程序运行函数等
- 安装glad：<https://glad.dav1d.de/>

Glad

Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator based on the official specs.

Language: C/C++

Specification: OpenGL

API: gl (Version 4.6), gles1 (None), gles2 (None), glsc2 (None)

Profile: Core

Extensions: Search, GL_3DFX_multisample, 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

Options: ☒ Generate a loader

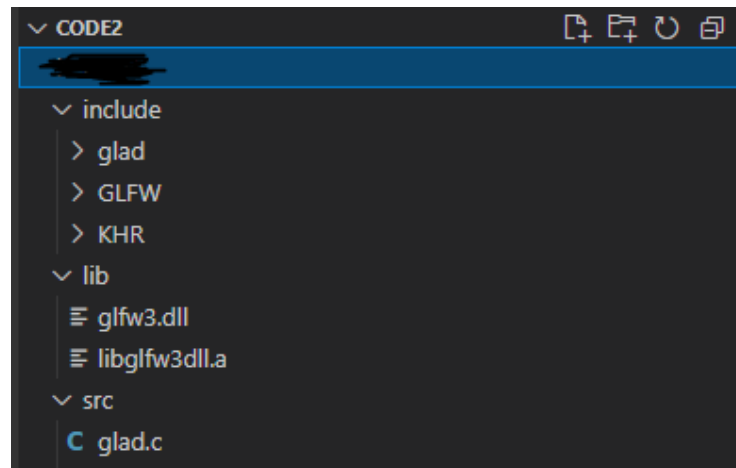
Glad

Generated files. These files are not permanent!

Name	Last modified	Size
include	2021-11-28 11:42:50	-
src	2021-11-28 11:42:50	-
glad.zip	2021-11-28 11:42:50	335.6 kB

Permalink:
<https://glad.dav1d.de/#language=c&specification=gl&api=gl%3D4.6&api=gles1%3Dnone&api=gles2%3Dnone&api=glsc2%3Dnone&profile=>

- 创建工程文件夹，命名为“code2”，在code2下创建lib、src、include
- 复制glad文件：
 - 将glad\include\下的glad、KHR文件复制到code2\include下
 - 将glad\src\glad.c复制到code2\src
- 复制glfw文件：
 1. 将glfw下的include\GLFW复制到code2\include下
 2. 将glfw下的lib-mingw-w64\glfw3.dll、lib-mingw-w64\libglfw3dll.a复制到code2\lib下
- 得到工程目录：



3. 编写cpp程序

- 在code2\src文件下创建main.cpp：

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

#include <stdio.h>

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

void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode);
void framebuffer_size_callback(GLFWwindow* window, int width, int height);

int main()
{
    //glfw初始化
    glfwInit();
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 4);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

    //glfw创建窗口
    GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "LearnOpenGL", NULL, NULL);
    if (window == NULL)
    {
        printf("创建窗口失败");
        //终止
        glfwTerminate();
        return -1;
    }
    //设置当前OpenGL上下文
    glfwMakeContextCurrent(window);

    //设置回调，当窗口大小调整后将调用该回调函数
    glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
    //设置回调，当发生按键操作时将调用该回调函数
    glfwSetKeyCallback(window, key_callback);
```

```

// glad初始化
if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
{
    printf("加载失败");
    return -1;
}

// 使用循环达到循环渲染效果
while (!glfwWindowShouldClose(window))
{
    //检查事件
    glfwPollEvents();

    //渲染指令
    glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    //交换缓冲
    glfwSwapBuffers(window);
}

//终止渲染 关闭并清理glfw本地资源
glfwTerminate();
return 0;
}

void key_callback(GLFWwindow* window, int key, int scancode, int action, int mode)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, GL_TRUE);
}
void framebuffer_size_callback(GLFWwindow* window, int width, int height)
{
    glViewport(0, 0, width, height);
}

```

4. 编译链接

- 在code2文件夹下新建CMakeLists.txt：

```

cmake_minimum_required(VERSION 3.0)

project(code2)
link_directories(${PROJECT_SOURCE_DIR}/lib)
message(STATUS "CMAKE_CXX_FLAGS: " ${CMAKE_CXX_FLAGS})
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -g -std=c++17")
message(STATUS "CMAKE_CXX_FLAGS: " ${CMAKE_CXX_FLAGS})

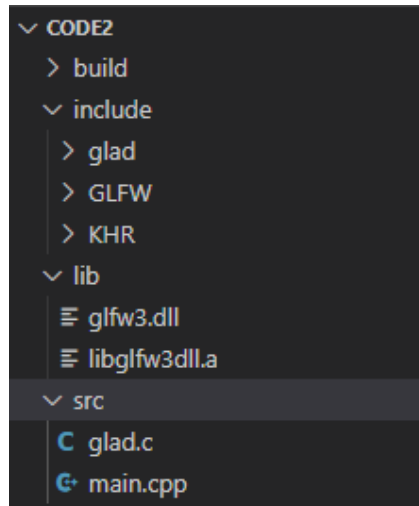
set(SOURCE_FILES src/main.cpp src/glad.c)
add_executable(main ${SOURCE_FILES})

include_directories(${PROJECT_SOURCE_DIR}/include)

```

```
target_link_libraries(main glfw3)
```

- 在code2文件夹下新建build文件夹，得到目录结构：



- 打开命令行，在build目录下运行：

```
> cmake -G"MinGW Makefiles" ..
```

输出如下：

```
(base) E:\opengl\code2\build>cmake -G"MinGW Makefiles" ..
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/msys64/mingw64/bin/gcc.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/msys64/mingw64/bin/g++.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- CMAKE_CXX_FLAGS:
-- CMAKE_CXX_FLAGS:  -g -std=c++17
-- Configuring done
-- Generating done
-- Build files have been written to: E:/opengl/code2/build
```

- 继续运行：

```
> cmake --build .
```

输出如下：

```
(base) E:\opengl\code2\build>cmake --build .  
Consolidate compiler generated dependencies of target main  
[ 33%] Building CXX object CMakeFiles/main.dir/src/main.cpp.obj  
[ 66%] Linking CXX executable main.exe  
[100%] Built target main
```

5. 运行

- 在code2文件夹下得到“main.exe”文件，再把lib下的glfw3.dll拷贝到code2目录下，双击main.exe，运行结果如下：

