MAC 下编写 opengl3+程序

mac os10.7 之后就开始支持 opengl 3.2 的语法，但 opengl3.3+在 10.9 才开始支持，当然具体支持的扩展是何显卡有关的，可以参考 Apple 官方说明

最近在 mac 和 windows 两边写 opengl 代码，但由于 mac 原本只支持 2.1，所以两边写不同的语法，让我很苦恼，而且在 mac 上无法学习一些较新的教程，在搜索了和踩了一些坑之后，终于实现在 mac 上编写 opengl 3+ 程序.

正文

安装 glfw

`OSX 对于 GL 2.1 以上只支持 Core Profile.`

`GLFW 支持创建 Core Profile Context.`

glfw 建议通过 brew 安装，在命令行界面:

执行以下命令:

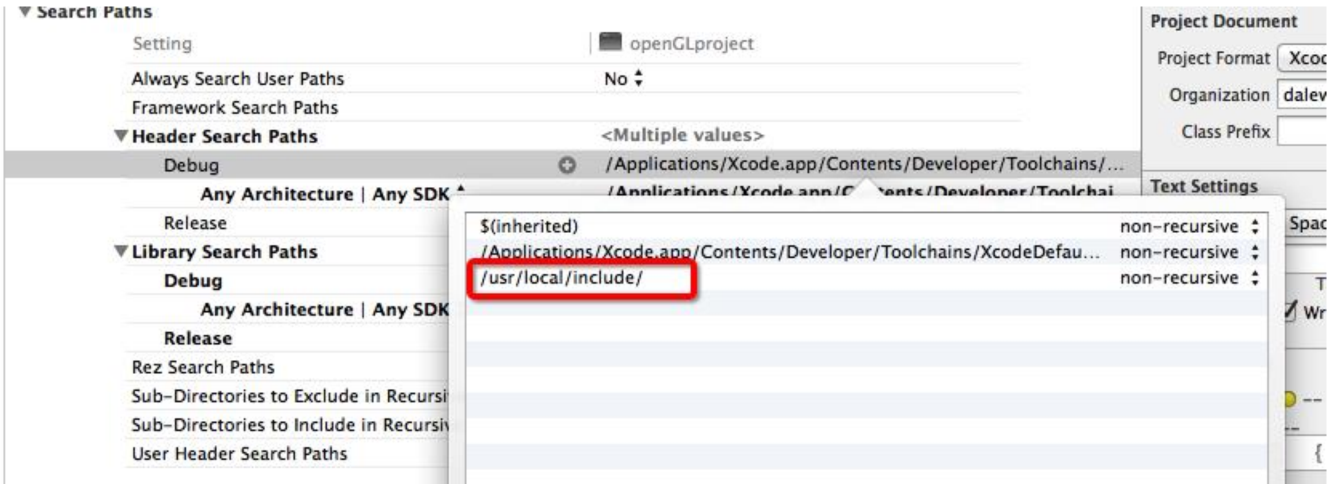
```
brew update
brew tap homebrew/versions
brew install glfw3
```

安装成功后，头文件和库文件会分别在/usr/local/include 和/usr/local/lib

在 xcode 中使用

build setting 的配置

在头文件路径中增加 /usr/local/include



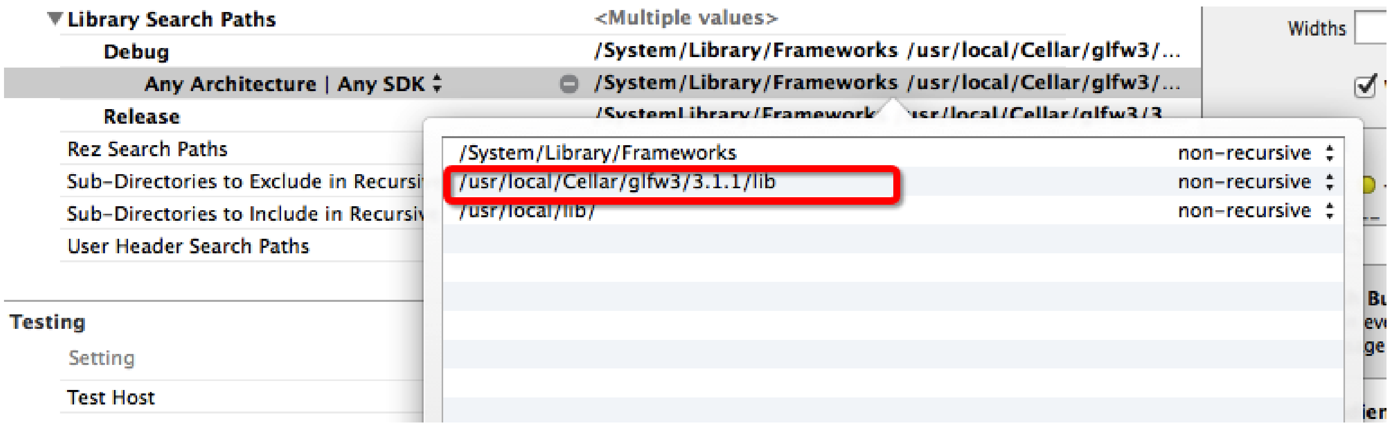
▼ Search Paths

Setting	openGLproject
Always Search User Paths	No
Framework Search Paths	
▼ Header Search Paths	<Multiple values>
Debug	/Applications/Xcode.app/Contents/Developer/Toolchains/...
Any Architecture | Any SDK	/Applications/Xcode.app/Contents/Developer/Toolchai
Release	
▼ Library Search Paths	
Debug	
Any Architecture | Any SDK	
Release	
Rez Search Paths	
Sub-Directories to Exclude in Recursi	
Sub-Directories to Include in Recursi	
User Header Search Paths	

$(inherited)	non-recursive
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefau...	non-recursive
/usr/local/include/	non-recursive

Project Document
Project Format Xcoc
Organization dalev
Class Prefix
Text Settings

在库路径中增加 /usr/local/Cellar/glfw3/3.1.1/lib



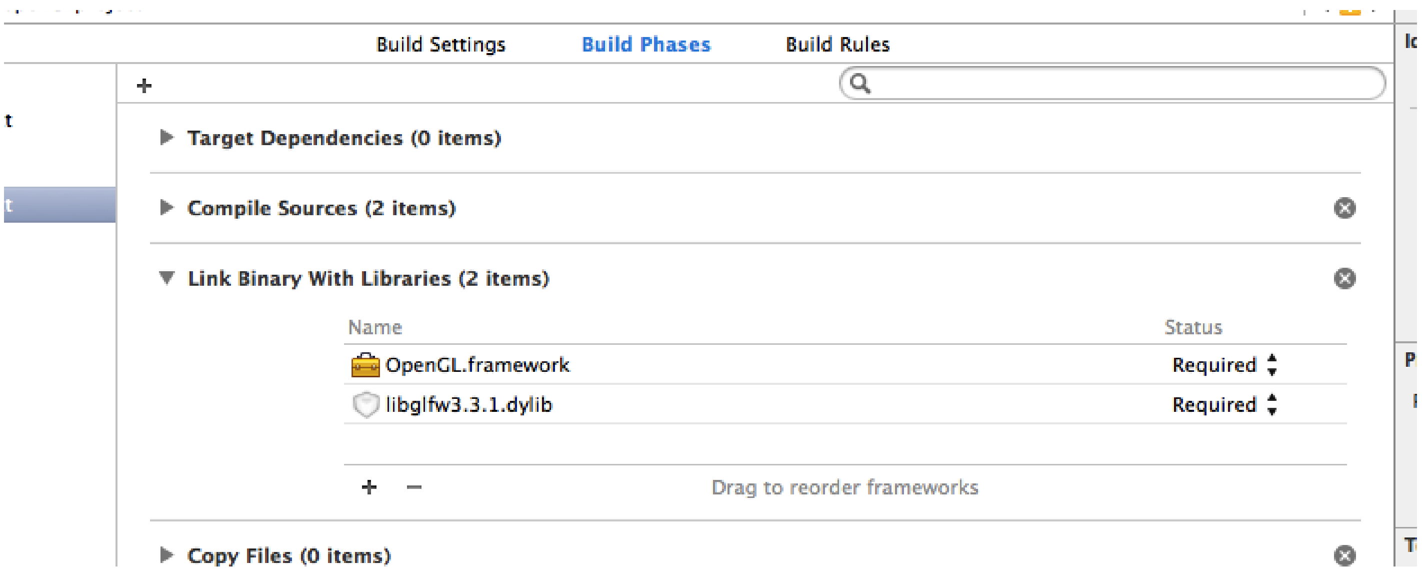
▼ Library Search Paths	<Multiple values>
Debug	/System/Library/Frameworks /usr/local/Cellar/glfw3/...
Any Architecture | Any SDK	/System/Library/Frameworks /usr/local/Cellar/glfw3/...
Release	/System/Library/Frameworks /usr/local/Cellar/glfw3/3
Rez Search Paths	
Sub-Directories to Exclude in Recursi	
Sub-Directories to Include in Recursi	
User Header Search Paths	

/System/Library/Frameworks	non-recursive
/usr/local/Cellar/glfw3/3.1.1/lib	non-recursive
/usr/local/lib/	non-recursive

Testing
Setting
Test Host

Widths

build pharse 中使用 glfw3 动态链接库



Build Settings Build Phases Build Rules

▶ Target Dependencies (0 items)

▶ Compile Sources (2 items)

▼ Link Binary With Libraries (2 items)

Name	Status
OpenGL.framework	Required
libglfw3.3.1.dylib	Required

Drag to reorder frameworks

▶ Copy Files (0 items)

opengl 样例

如果仅仅按 glfw 官网教程框架是无法编写 opengl3+的程序的，我们必须在创建 glfw 窗口之前加上以下 4 行：

```
  glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
  glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
  glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
  glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 2);
```

而且不要使用 glut 框架

注意，**一旦加了这 4 行，以前的固定管线函数就无效了，像 glbegin,glend 就无法使用了。**

下面给一个示例程序:

```
#include <GLFW/glfw3.h>
#include <OpenGL/gl3.h>
#include <stdlib.h>
#include "initShader.h"
static void error_callback(int error, const char* description)
{
    fputs(description, stderr);
}
static void key_callback(GLFWwindow* window, int key, int scancode, int ac
tion, int mods)
{
    if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
        glfwSetWindowShouldClose(window, GL_TRUE);
}


static const GLfloat g_vertex_buffer_data[] = {
    -1.0f, -1.0f, 0.0f,
    1.0f, -1.0f, 0.0f,
    0.0f, 1.0f, 0.0f,
};

#define printGlErr(func) err = glGetError(); if (err) fprintf(stderr, func
 " error: %u at line %d\n", err, __LINE__);
GLuint err =GL_INVALID_OPERATION;

void init()
{

    GLuint vaoId = 0;
    glGenVertexArrays(1, &vaoId);
    glBindVertexArray(vaoId);
    // Generate 1 buffer, put the resulting identifier in vertexbuffer
    GLuint vertexbuffer;
    glGenBuffers(1, &vertexbuffer);
    printGlErr("glGenBuffers()");
    // The following commands will talk about our 'vertexbuffer' buffer
    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);

    // Give our vertices to OpenGL.
    glBufferData(GL_ARRAY_BUFFER, sizeof(g_vertex_buffer_data), g_vertex_
buffer_data, GL_STATIC_DRAW);

    GLuint program = InitShader("/Users/dale/Documents/gameDevelop/opengl
_learn/openGL/openGLproject/openGLproject/vertex.glsl","/Users/dale/Docu
ments/gameDevelop/opengl_learn/openGL/openGLproject/openGLproject/fragme
nt.glsl");
    GLuint loc = glGetAttribLocation(program, "vPosition");
    glEnableVertexAttribArray(loc);

    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
    glVertexAttribPointer(
                          loc,              // attribute 0. No particular re
ason for 0, but must match the layout in the shader.
                          3,                // size
                          GL_FLOAT,         // type
                          GL_FALSE,         // normalized?
                          0,                // stride
                          (void*)0          // array buffer offset
                          );
    printGlErr("glVertexAttribPointer()");

}




void display()
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glDrawArrays(GL_TRIANGLES, 0, 3);
    glFlush();
}

int main(void)
{

    GLFWwindow* window;
    glfwSetErrorCallback(error_callback);
    if (!glfwInit())
        exit(EXIT_FAILURE);
     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);


    window = glfwCreateWindow(640, 480, "Simple example", NULL, NULL);
    if (!window)
    {
        glfwTerminate();
        exit(EXIT_FAILURE);
    }
    glfwMakeContextCurrent(window);
    glfwSwapInterval(1);
    glfwSetKeyCallback(window, key_callback);
    init();
    while (!glfwWindowShouldClose(window))
    {
        float ratio;
        int width, height;
        glfwGetFramebufferSize(window, &width, &height);
        ratio = width / (float) height;
        glViewport(0, 0, width, height);
        glClear(GL_COLOR_BUFFER_BIT);

        display();
        glfwSwapBuffers(window);
        glfwPollEvents();
    }
    glfwDestroyWindow(window);
    glfwTerminate();
    exit(EXIT_SUCCESS);
}
```

未给出的是 initshader 函数代码，这个网上很多实现，就不加在这里显的代码很长了

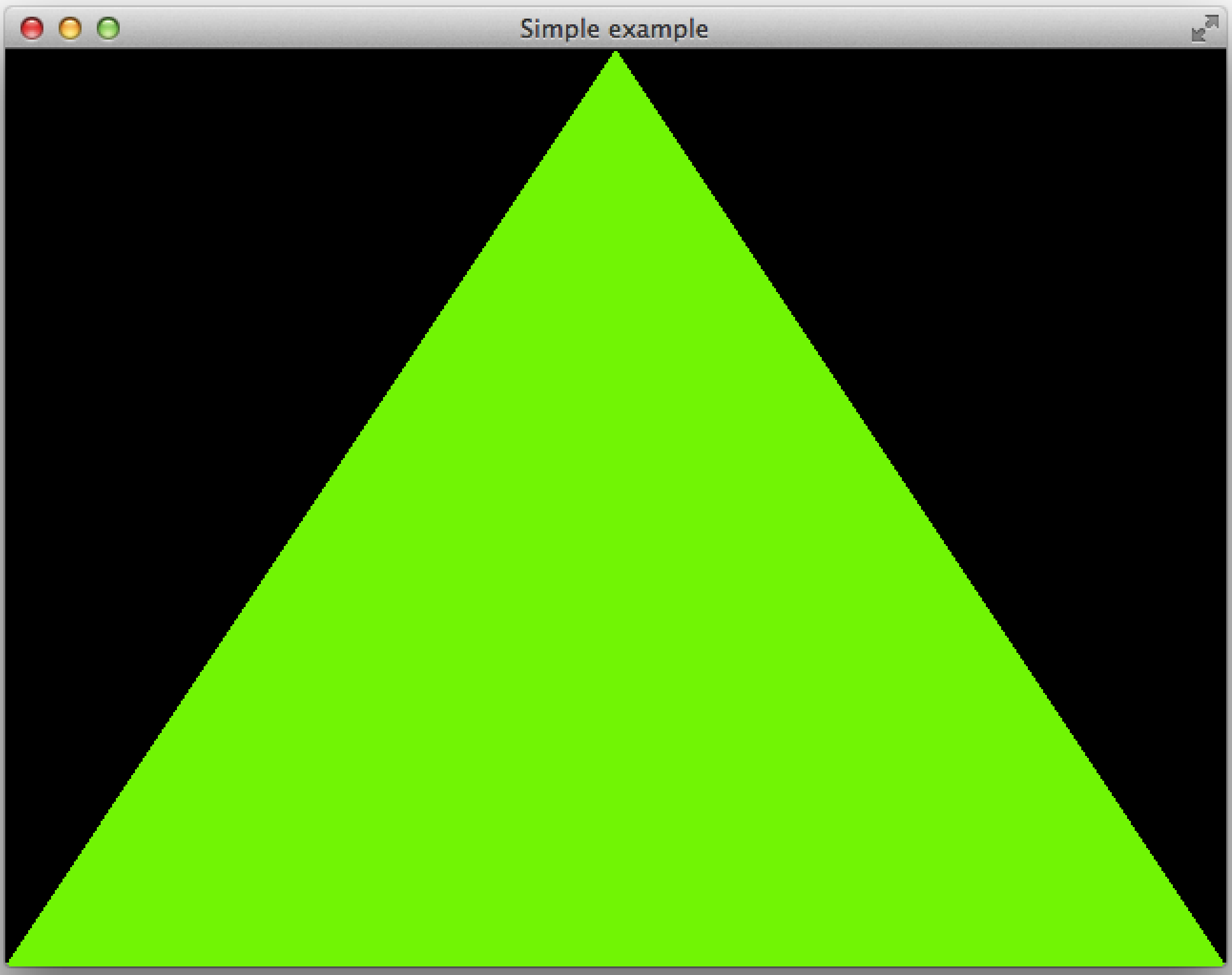
然后是 vertex shader；

```
#version 330 core
layout(location =0 )in vec3 vPosition;
void main()
{
    gl_Position = vec4(vPosition,1.0);
}
```

fragment shader：

```
#version 330 core
out vec4 glcolor;
void main()
{
    glcolor = vec4(0.0, 1.0, 0.0, 1.0);
}
```

编译运行就看到一个绿色的三角形:



祝大家好运：）