**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](https://developer.apple.com/opengl/capabilities/index.html" \t "_blank)   
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

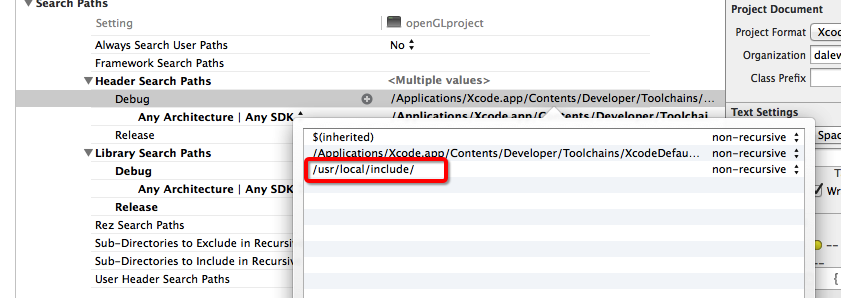
OSX对于GL 2.1以上只支持Core Profile.

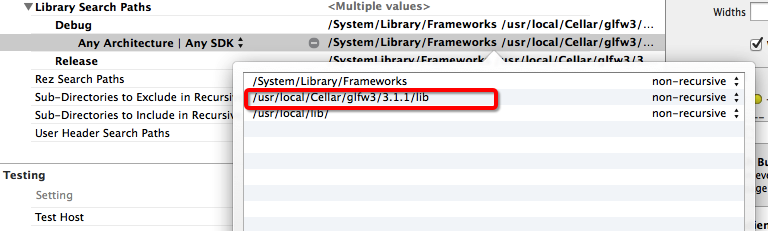
GLFW支持创建Core Profile Context.

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

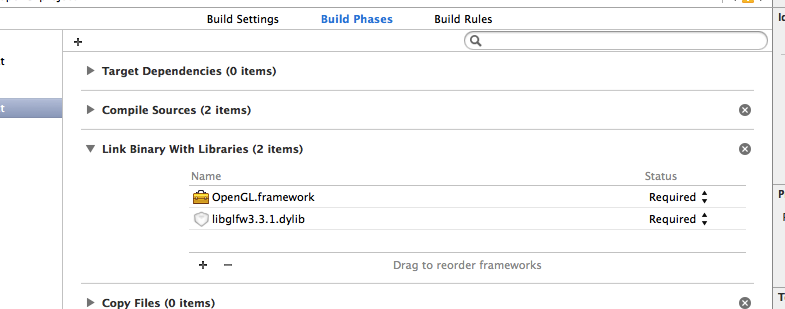
在xcode中使用

build setting的配置

在头文件路径中增加 /usr/local/include   


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


build pharse中使用glfw3动态链接库



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 action, 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/Documents/gameDevelop/opengl\_learn/openGL/openGLproject/openGLproject/fragment.glsl");

GLuint loc = glGetAttribLocation(program, "vPosition");

glEnableVertexAttribArray(loc);

glBindBuffer(GL\_ARRAY\_BUFFER, vertexbuffer);

glVertexAttribPointer(

loc, // attribute 0. No particular reason 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

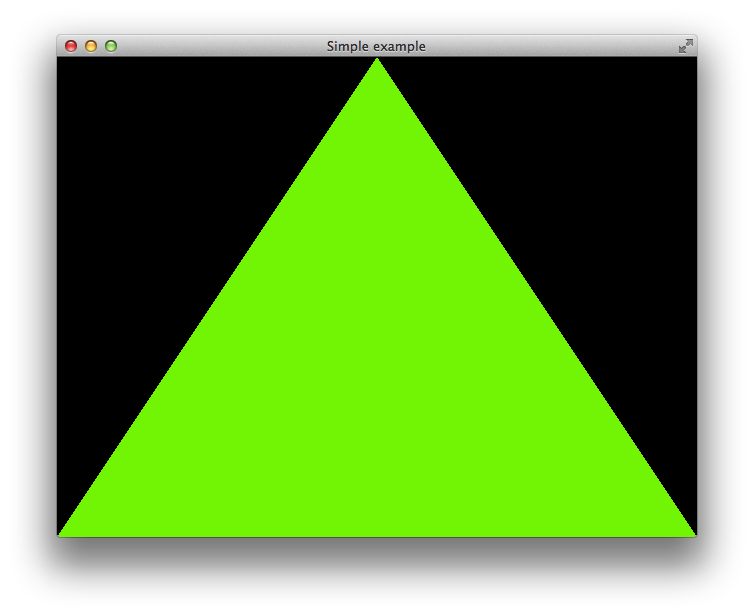
out vec4 glcolor;

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：）