**【OpenGL学习】glut和freeglut下载及配置**

4 篇文章0 订阅

订阅专栏

**目录**

**目录**

[1、glut下载](https://blog.csdn.net/hhy321/article/details/121065642#t0)

[2、freeglut下载](https://blog.csdn.net/hhy321/article/details/121065642#t1)

[3、代码示例](https://blog.csdn.net/hhy321/article/details/121065642#t2)

[3.1 打印显卡驱动信息](https://blog.csdn.net/hhy321/article/details/121065642#t3)

[3.2 绘制一个三角形](https://blog.csdn.net/hhy321/article/details/121065642#t4)

[3.3 使用VBO绘制一个三角形](https://blog.csdn.net/hhy321/article/details/121065642#t5)

[3.4 使用VAO，VBO绘制一个矩形](https://blog.csdn.net/hhy321/article/details/121065642#t6)

[4、调试OpenGL](https://blog.csdn.net/hhy321/article/details/121065642#t7)

[后续](https://blog.csdn.net/hhy321/article/details/121065642#t8)

**1、glut下载**

[OpenGL](https://so.csdn.net/so/search?q=OpenGL&spm=1001.2101.3001.7020)工具库 OpenGL Utility Toolkit，包含大约30多个函数，函数名前缀为glut。

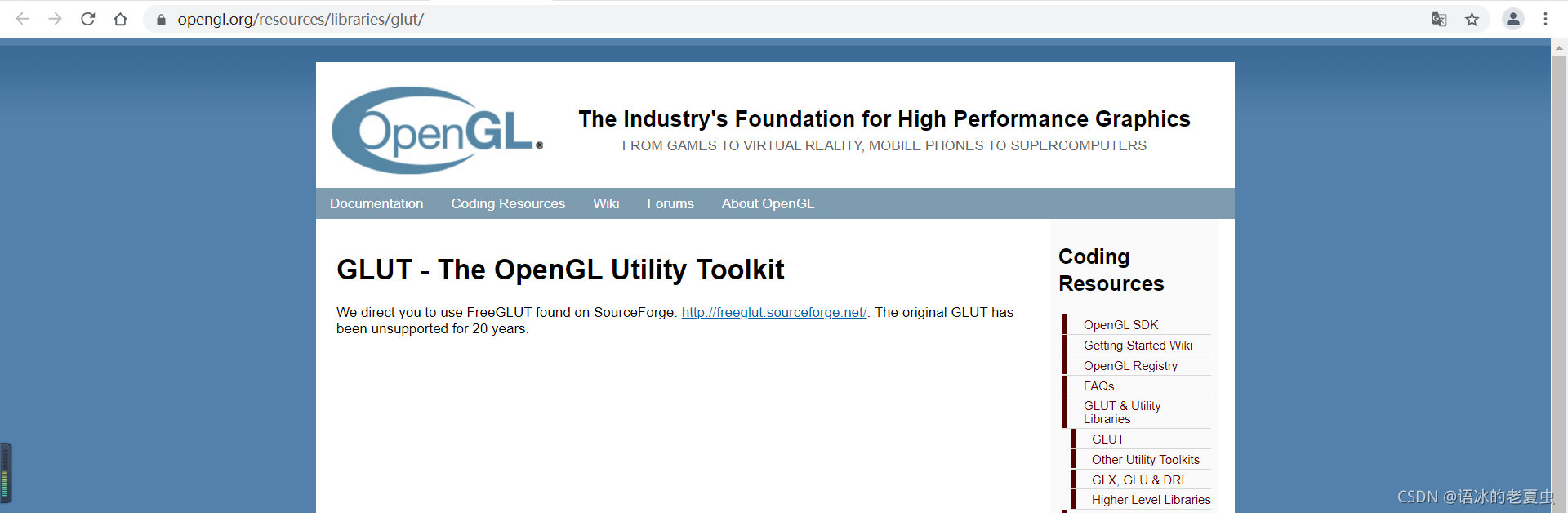
glut是不依赖于窗口平台的OpenGL工具包，由Mark KLilgrad在SGI编写（现在在Nvidia），目的是隐藏不同窗口平台[API](https://so.csdn.net/so/search?q=API&spm=1001.2101.3001.7020)的复杂度。 函数以glut开头，它们作为aux库功能更强的替代品，提供更为复杂的绘制功能，此函数由glut.dll来负责解释执行。由于glut中的窗口管理函数是不依赖于运行环境的，因此OpenGL中的工具库可以在X-Window, Windows NT, OS/2等系统下运行，特别适合于开发不需要复杂界面的OpenGL示例程序。对于有经验的程序员来说，一般先用glut理顺3D图形代码，然后再集成为完整的应用程序。

这部分函数主要包括

* **窗口操作函数：**窗口初始化、窗口大小、窗口位置等函数glutInit() glutInitDisplayMode() glutInitWindowSize() glutInitWindowPosition()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，GlutDisplayFunc() glutPostRedisplay() glutReshapeFunc() glutTimerFunc() glutKeyboardFunc() glutMouseFunc()
* **创建复杂的三维物体：**这些和aux库的函数功能相同。创建网状体和实心体。如glutSolidSphere()、glutWireSphere()等
* **菜单函数**：创建添加菜单的函数GlutCreateMenu()、glutSetMenu()、glutAddMenuEntry()、glutAddSubMenu() 和glutAttachMenu()
* **程序运行函数：**glutMainLoop()

GLUT（OpenGL Utility Toolkit）是一个跨平台工具库，只能在win32下使用，不提供64位的开发环境，而且在1998年之后一直没有更新了。

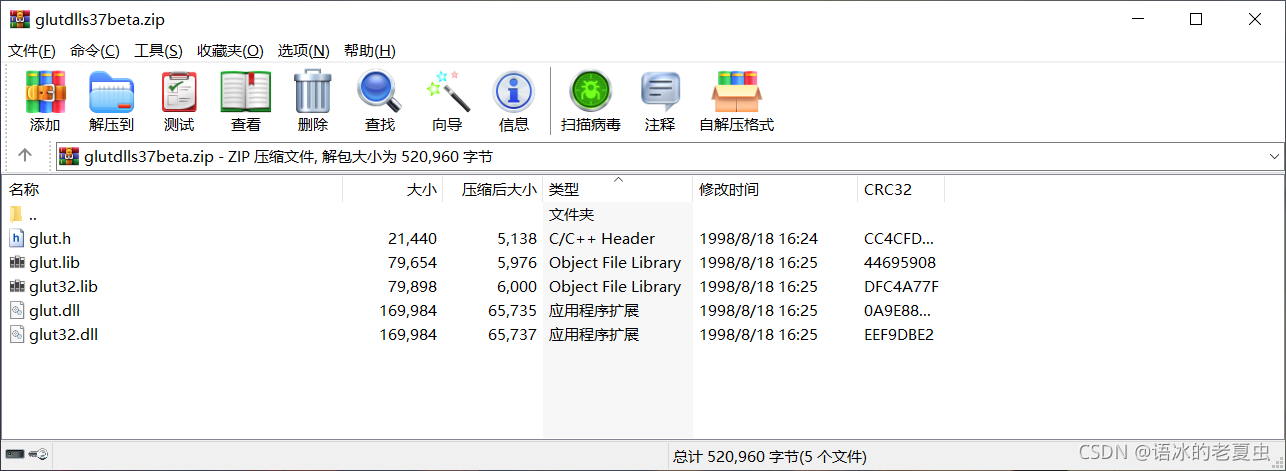
目前最后版本GLUT v3.7的历史可追溯至1998年8月，且该项目似乎已经被废弃。它的许可证禁止任何人发布修改后的库代码。



 glut32.dll和glut.dll有什么区别？glut32.dll是为微软封装的，官方的。而glut.dll是sgi封装的。

 官网地址：

[GLUT - The OpenGL Utility Toolkit](http://www.opengl.org/resources/libraries/glut/)  
<http://www.opengl.org/resources/libraries/glut/glutdlls37beta.zip>

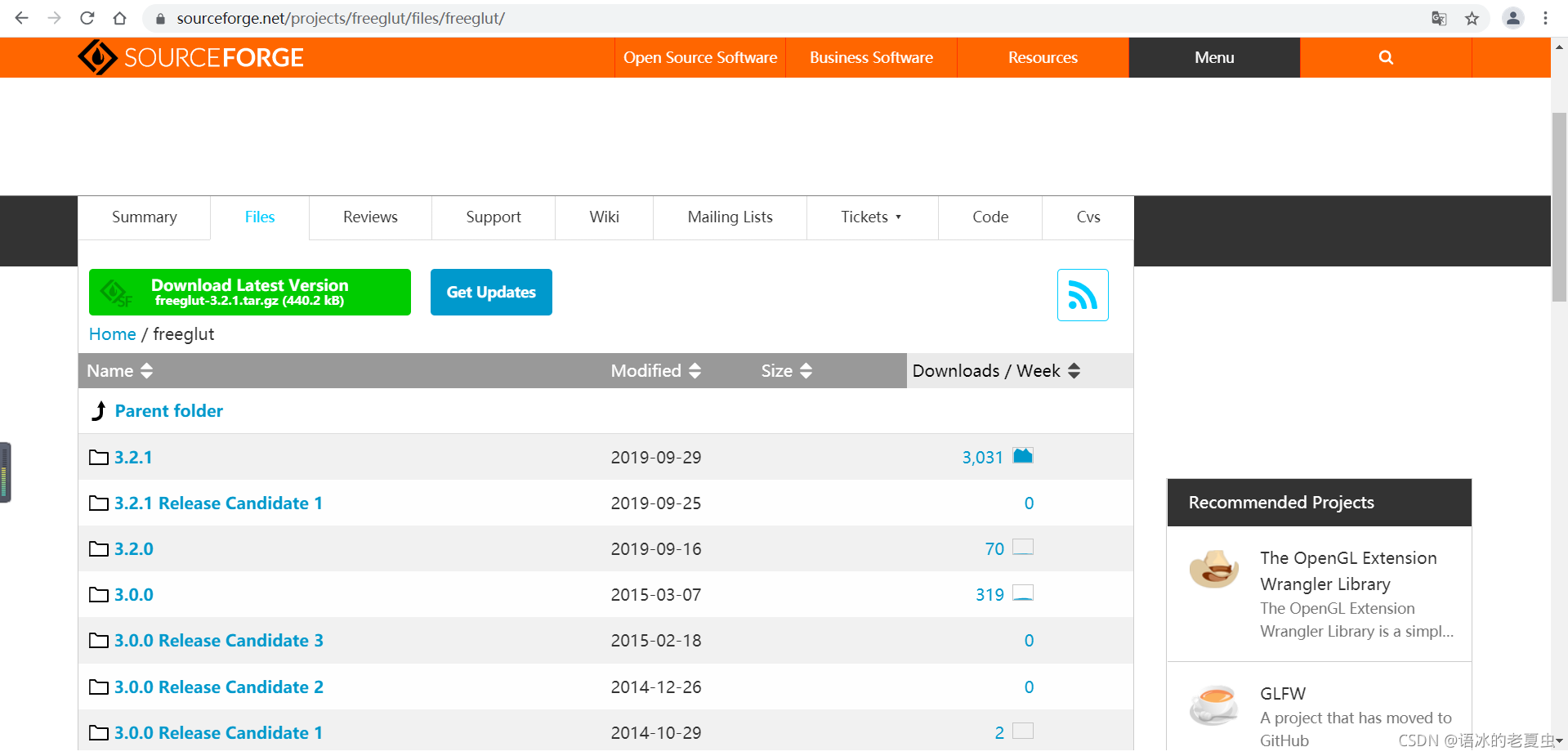


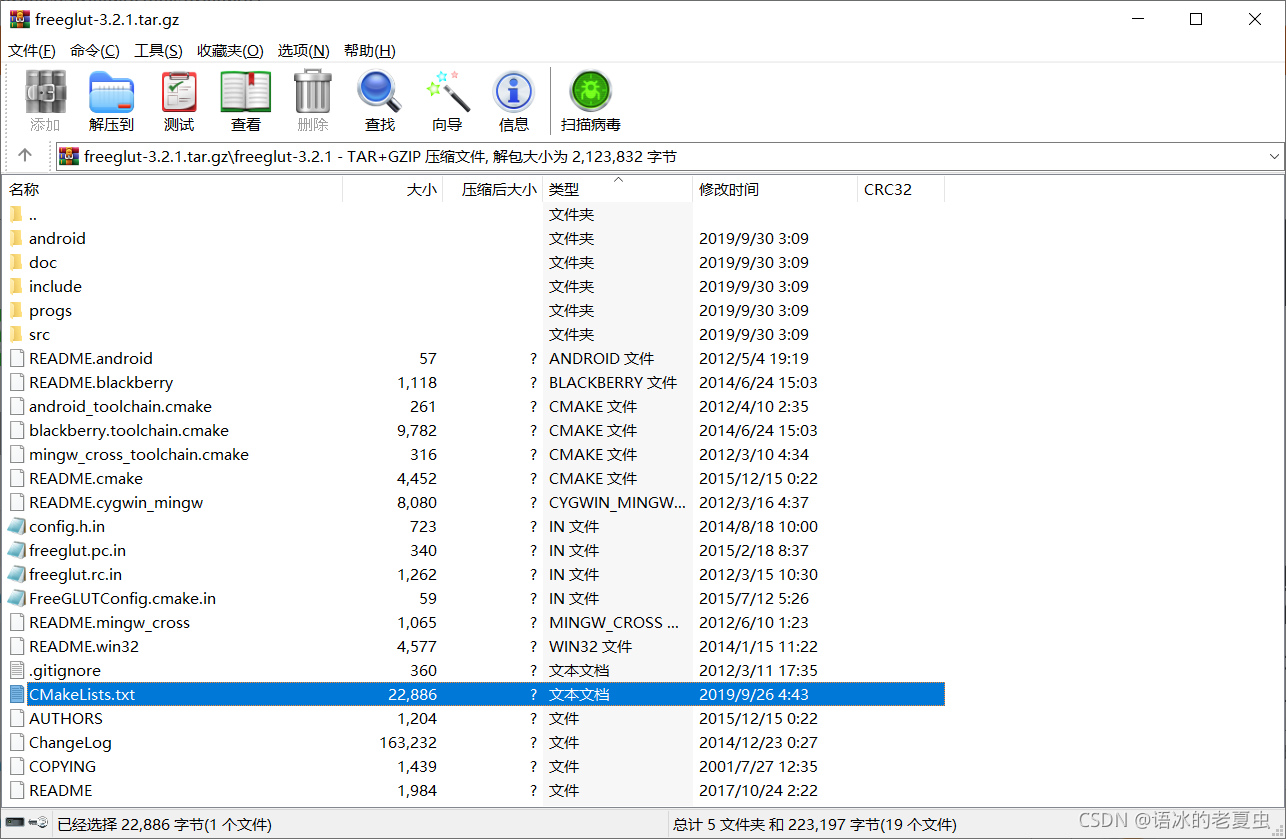
**2、freeglut下载**

freeglut is an open-source alternative to the OpenGL Utility Toolkit (GLUT) library.

freeglut最初由帕维尔·W. Olszta与安德烈亚斯·乌姆巴赫和史蒂夫·贝克的贡献。史蒂夫是现在官方的拥有者/维护者。帕维尔在1999年12月1日开始freeglut的开发。目前，该项目几乎可以100%的替代原来的GLUT，只有少数差别（如the abandonment of SGI-specific features，按钮盒子和动态视频分辨率）和 其他一小部分程序Bug。

这个库是原始的GLUT库修改之后得到的，它既提供32位、也提供64位的编译环境。





官网地址：  
  
<http://freeglut.sourceforge.net/>  
[freeglut download | SourceForge.net](https://sourceforge.net/projects/freeglut/)

Martin Payne's Windows binaries (MSVC and MinGW)：<http://www.transmissionzero.co.uk/software/freeglut-devel/>

DLL放置目录：  
32位的DLL X:\Windows\System32  
64位的DLL X:\Windows\SysWOW64

**3、代码示例**

**3.1 打印显卡驱动信息**

#include <windows.h>

#include <stdio.h>

#include <GL/freeglut.h>

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(300, 300);

glutInitWindowPosition(100, 100);

glutCreateWindow("OpenGL Version");

const GLubyte\* szVendor = glGetString(GL\_VENDOR);

const GLubyte\* szRender = glGetString(GL\_RENDERER);

const GLubyte\* szGLVersion = glGetString(GL\_VERSION);

const GLubyte\* szGluVersion = gluGetString(GLU\_VERSION);

printf("OpenGL实现厂商的名字：%s\n", szVendor);

printf("渲染器标识符：%s\n", szRender);

printf("OpenGL实现的版本号：%s\n", szGLVersion);

printf("GLU工具库版本：%s\n", szGluVersion);

system("pause");

return 0;

}

**3.2 绘制一个三角形**

#include <GL/freeglut.h>

void renderScene(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glBegin(GL\_TRIANGLES);

glVertex3f(-0.5, -0.5, 0.0);

glVertex3f(0.5, 0.0, 0.0);

glVertex3f(0.0, 0.5, 0.0);

glEnd();

glutSwapBuffers();

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, (char\*\*)argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(100, 100);

glutInitWindowSize(800, 480);

glutCreateWindow("第一个glut程序");

glutDisplayFunc(renderScene);

glutMainLoop();

return 0;

}

**3.3 使用VBO绘制一个三角形**

 顶点缓冲对象（Vertex Buffer Objects，VBO）

#include <stdio.h>

#include <GL/glew.h>

#include <GL/freeglut.h>

GLuint VBO;

static void RenderSceneCB()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glEnableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, VBO);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, 0);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glDisableVertexAttribArray(0);

glutSwapBuffers();

}

static void InitializeGlutCallbacks()

{

glutDisplayFunc(RenderSceneCB);

}

static void CreateVertexBuffer()

{

Vector3f Vertices[3];

Vertices[0] = Vector3f(-1.0f, -1.0f, 0.0f);

Vertices[1] = Vector3f(1.0f, -1.0f, 0.0f);

Vertices[2] = Vector3f(0.0f, 1.0f, 0.0f);

glGenBuffers(1, &VBO);

glBindBuffer(GL\_ARRAY\_BUFFER, VBO);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(Vertices), Vertices, GL\_STATIC\_DRAW);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA);

glutInitWindowSize(1024, 768);

glutInitWindowPosition(100, 100);

glutCreateWindow("glut + VBO + triangle ");

InitializeGlutCallbacks();

// Must be done after glut is initialized!

GLenum res = glewInit();

if (res != GLEW\_OK) {

fprintf(stderr, "Error: '%s'\n", glewGetErrorString(res));

return 1;

}

glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

CreateVertexBuffer();

glutMainLoop();

return 0;

}

**3.4 使用VAO，VBO绘制一个矩形**

* 顶点缓冲对象（Vertex Buffer Objects，VBO）
* 顶点数组对象（Vertex Arrary Object，VAO）
* 索引缓冲对象（Element Buffer Object，EBO）

//使用VAO VBO绘制矩形

#include <GL/glew.h>

#include <GL/freeglut.h>

GLuint vboId;//vertex buffer object句柄

GLuint vaoId;//vertext array object句柄

//自定义初始化函数

void myInit()

{

glClearColor(0.0, 0.0, 0.0, 0.0);

//创建顶点数据

const GLfloat vertices[] = {

-0.5f,-0.5f,0.0f,1.0f,

0.5f,-0.5f,0.0f,1.0f,

0.5f,0.5f,0.0f,1.0f,

-0.5f,0.5f,0.0f,1.0f,

};

//创建VAO对象

glGenVertexArrays(1, &vaoId);

glBindVertexArray(vaoId);

//创建VBO对象

glGenBuffers(1, &vboId);

glBindBuffer(GL\_ARRAY\_BUFFER, vboId);

//传入VBO数据

glBufferData(GL\_ARRAY\_BUFFER, sizeof(vertices), vertices, GL\_STATIC\_DRAW);

//解除VBO绑定

glBindBuffer(GL\_ARRAY\_BUFFER, 0);

}

//调整窗口大小回调函数

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

}

//绘制回调函数

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

//绑定VBO

glBindBuffer(GL\_ARRAY\_BUFFER, vboId);

glEnableVertexAttribArray(0);

//解释顶点数据方式

glVertexAttribPointer(0, 4, GL\_FLOAT, GL\_FALSE, 0, 0);

//绘制模型

glDrawArrays(GL\_TRIANGLE\_FAN, 0, 4);

glBindBuffer(GL\_ARRAY\_BUFFER, 0);

glDisableVertexAttribArray(0);

glutSwapBuffers();

}

int main(int argc, char \*\*argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE);

glutInitWindowPosition(100, 100);

glutInitWindowSize(512, 512);

glutCreateWindow("Rectangle demo");

//初始化一个GLEW库,确保OpenGL API对程序完全可用。

GLenum err = glewInit();

if (GLEW\_OK != err)

{

/\* Problem: glewInit failed, something is seriously wrong. \*/

fprintf(stderr, "Error: %s\n", glewGetErrorString(err));

return;

}

//自定义初始化，生成VAO，VBO对象

myInit();

//设置回调函数

glutReshapeFunc(reshape);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**4、调试OpenGL**

#include <iostream>

#include <GL/glew.h>

#include <GL/freeglut.h>

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glDrawArrays(666, 666, 666);

glutSwapBuffers();

}

void keyboard(unsigned char key, int, int)

{

if (key == 27) exit(EXIT\_SUCCESS);

}

void GLAPIENTRY debug(GLenum source, GLenum type, GLuint id, GLenum severity, GLsizei length, const GLchar\* message, const void\* userParam)

{

std::cout << std::hex

<< "source:0x" << source << std::endl

<< "type:0x" << type << std::endl

<< "id:0x" << id << std::endl

<< "severity:0x" << severity << std::endl

<< "length:" << length << std::endl

<< "message:" << message << std::endl

<< "userParam:" << userParam << std::endl

<< std::dec;

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitContextFlags(GLUT\_FORWARD\_COMPATIBLE | GLUT\_DEBUG);

glutInitContextProfile(GLUT\_COMPATIBILITY\_PROFILE);

glutInitContextVersion(3, 3);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_DEPTH);

glutCreateWindow("");

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

glewInit();

glDebugMessageCallback(debug, nullptr);

glutMainLoop();

return 0;

}

**后续**

 如果你觉得该方法或代码有一点点用处，可以给作者点个赞；╮(￣▽￣)╭  
如果你感觉方法或代码不咋地//(ㄒoㄒ)//，就在评论处留言，作者继续改进。o\_O???  
谢谢各位童鞋们啦( ´ ▽ ` )ﾉ ( ´ ▽ ` )っ！！！