实验2.1 OpenGL鼠标键盘的交互

// 这个实验和之前的实验略有不同，之前的实验需要完成鼠标左键点击事件、鼠标右键呼出小菜单、绘制子窗口以及子窗口的键盘响应，由于我们把库更新为GLFW，其中不包含创建小菜单以及子窗口的API，在网上查到的解决方法可能要修改GLFW的源码再重新编译，比较麻烦，因此目前这个实验只有对鼠标（点击和滚轮滚动）以及键盘事件的响应。

1. 实验目的
2. 学会OpenGL的简单键盘交互。
3. 掌握OpenGL的简单鼠标交互。
4. 进一步巩固OpenGL的基本图元绘制基础。
5. 理论背景
6. 键盘交互事件

当鼠标位于窗口内并且键盘有某个键被按下或释放，就会产生键盘事件。当发生键盘事件时，系统会返回事件发生窗口的指针、键盘按键的索引、事件发生的类型（按下或者释放）等信息。

1. 鼠标交互事件

GLFW可以响应两种类型的鼠标交互，分别为鼠标按键的点击与鼠标滚轮的滚动。当发生鼠标点击事件时，系统会返回点击按键的索引、事件发生的类型等信息，当发生滚轮滚动事件时，系统会返回滚轮在x、y轴方向上的偏移等信息。

1. 代码讲解

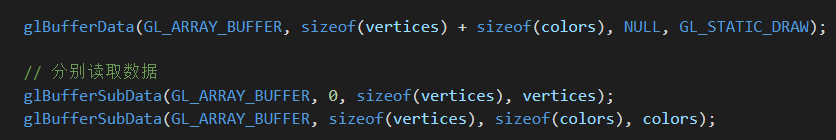
本次实验需要加入鼠标键盘操作来控制绘制效果。

1. 传输数据glBufferSubData()函数（以mainWindowInit为例）

之前我们对坐标、颜色这些不同顶点属性定义了多个vbo进行数据传递，但是我们也可以只使用一个顶点缓存对象vbo(vertex buffer object)，用来处理和管理各种顶点数据。与实验1.2传递数据使用glBufferData()函数不同，实验2.1开始将使用更方便的glBufferSubData()函数。



我们可以只用创建一个vbo，先使用glBufferData()函数分配一个缓存空间。再使用glBufferSubData()函数用于向vbo中拷贝数据，他需要从给定从offset开始的一定范围大小的数据替换到现存缓存中。



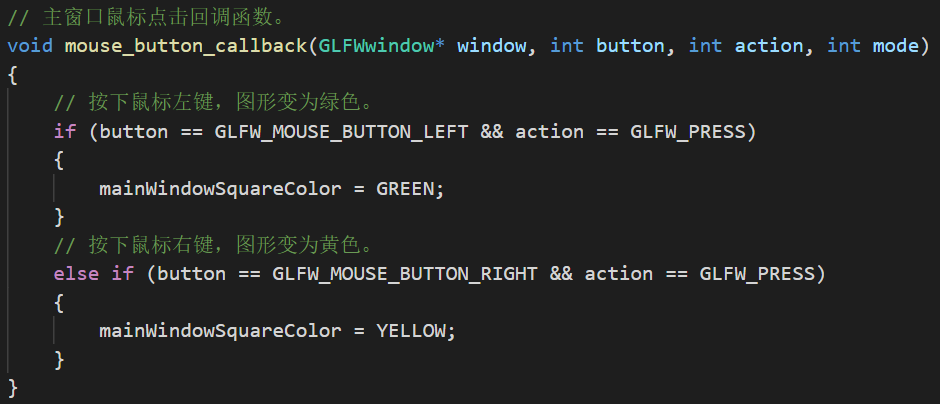
void glBufferSubData(GLenum target, GLintptr offset, GLsizeiptr size, const GLvoid \*data);

* target: 参考glBufferData，用来指定需要更新的缓冲区对象的类型
* offset: 指定了更新数据相对于缓冲区对象中原始数据开始位置的偏移量，也就是说要从什么地方开始更新原来的数据（以字节为单位）
* size：需要更新的数据量的大小
* data：一个指向新数据源的指针，将新的数据源拷贝到缓冲区对象中完成更新

glBufferSubData用来更新缓冲区对象中的数据，可以更新一部分或者整个缓冲区中的数据（取决于offset和size的取值），如果offset是0，而size正好是用来缓冲区存储数据的大小，那么整个数据就会被新的数据所替换。需要注意的是offset和size必须保证在原来缓冲区对象可以表示的范围之内。（假如原来缓冲区对象中存储了1024个字节，那么offset就不能大于1024，因为这样的话就没法更新到这个缓冲区内的数据了）

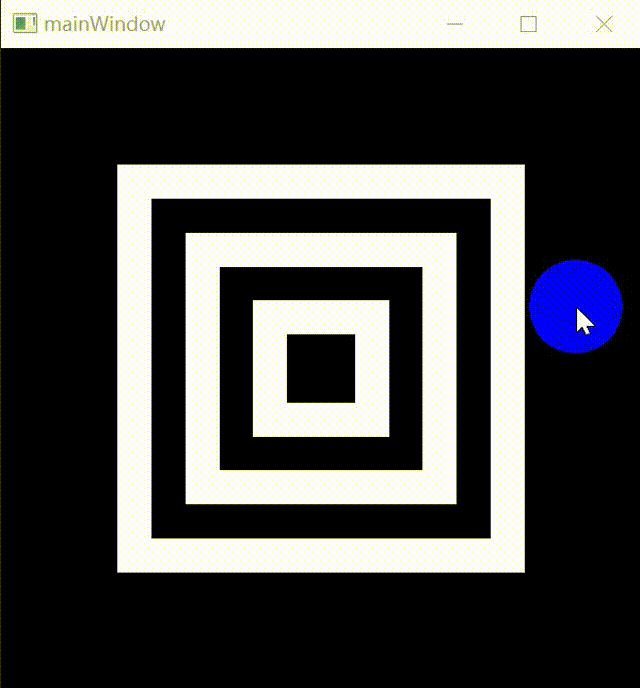
1. 定义鼠标点击回调函数

窗口的鼠标点击回调函数mouse\_button\_callback()的代码如下：



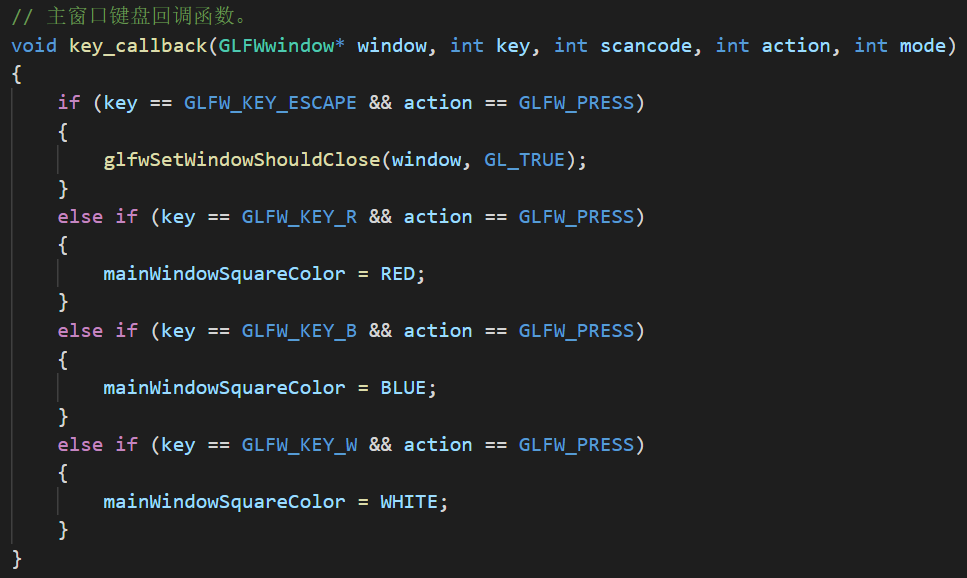
在这里，鼠标回调函数发挥的作用是在按下鼠标左键时，设置图形的颜色为绿色，按下鼠标右键时，设置图形的颜色为黄色。

最后，在主函数main()中通过glfwSetMouseButtonCallback()函数对鼠标回调函数mouse\_button\_callback ()进行关联。在完成上述步骤后，将得到一个具有如下效果的主窗口。

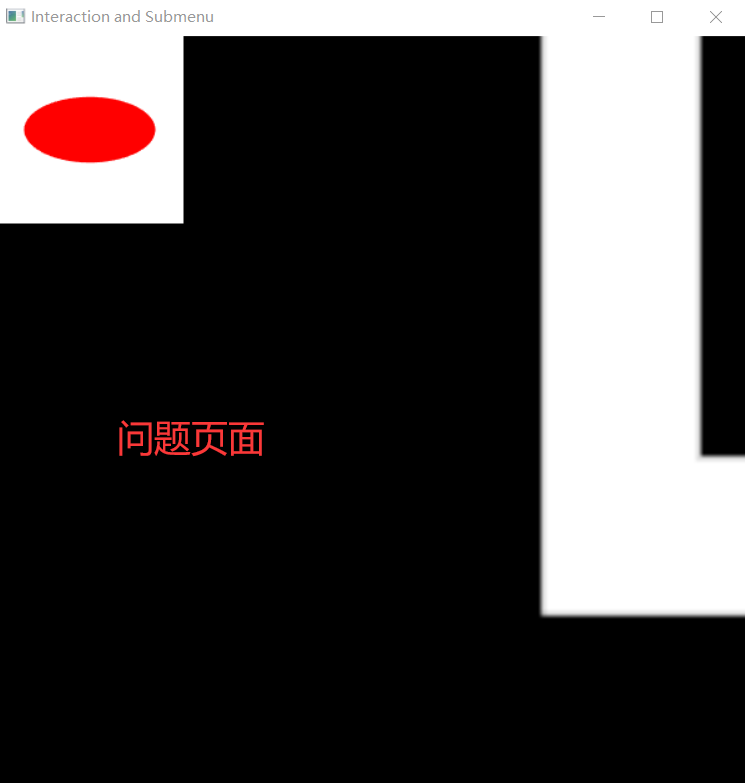


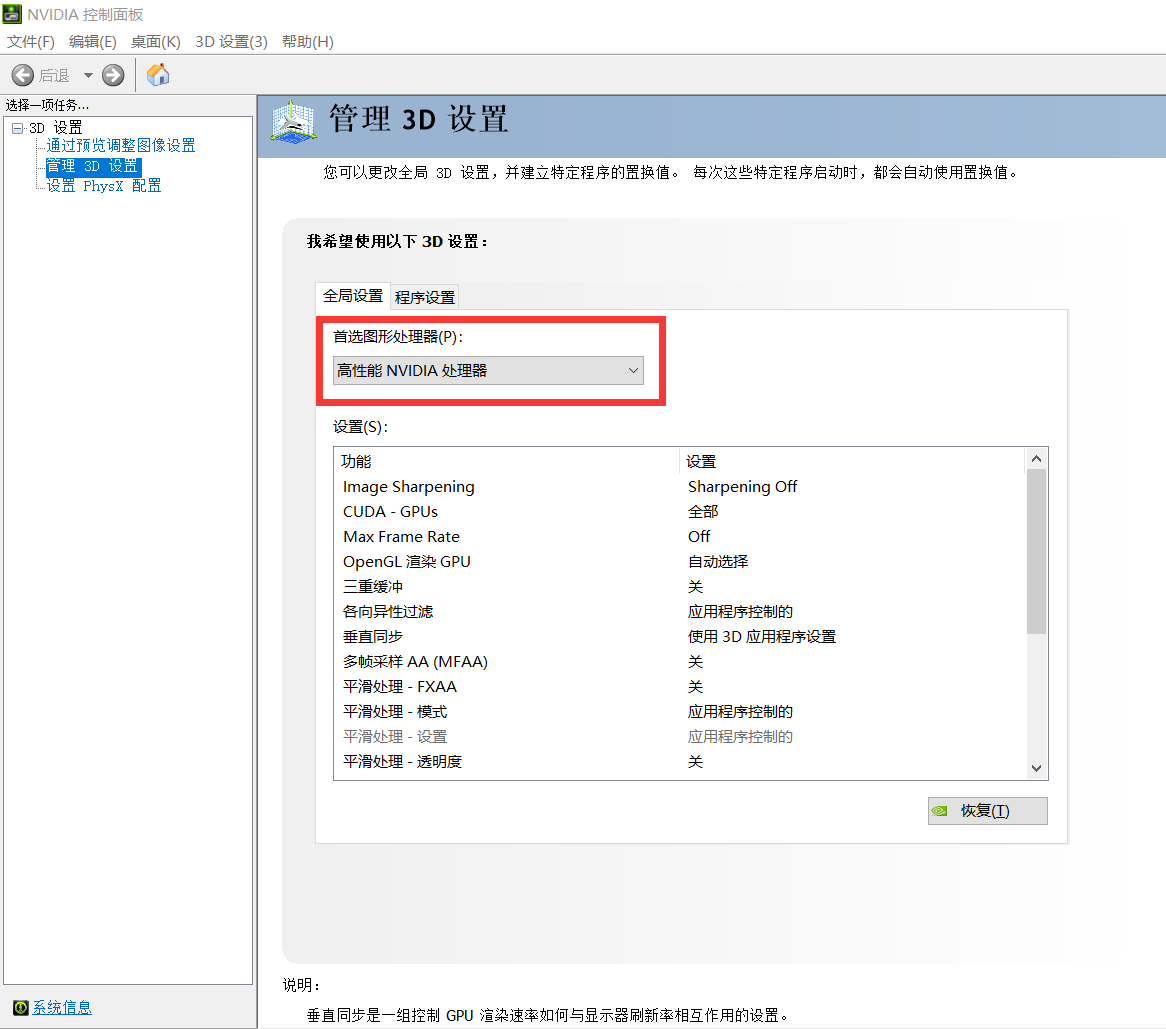
1. 定义键盘响应回调函数

主窗口的键盘响应回调函数实现了根据键盘按键对窗口中物体颜色的赋值，具体实现如下。



1. 课堂练习
2. 在主窗口中添加响应鼠标滚轮滚动的函数void sroll\_callback(GLFWWindow\* window, doubel xoffset, double yoffset)，要求实现的功能为通过鼠标滚轮的上（下）滚动控制窗口中图形的顺（逆）时针旋转，并在main()函数中使用glfwSetScrollCallback()函数进行链接绑定。
3. 有些Window电脑上可能会出现主窗口图形不正常缩放的情况，这可能是因为电脑显卡使用了集成显卡导致，解决方法如下：打开显卡控制器→管理3D设置→首选图形处理器→设置为“高性能NVIDIA处理器”，重新运行程序即可。





1. 函数小结
2. void mouse\_button\_callback(GLFWWindow\* window, int button, int action, int mode);

指定了函数func，当鼠标被按下或释放时将会被调用。

1. void key\_callback (GLFWWindow\* window, int key, int action, int scancode, int mode);

指定了函数func，当键盘上有按键被按下时，这个函数便会被调用。