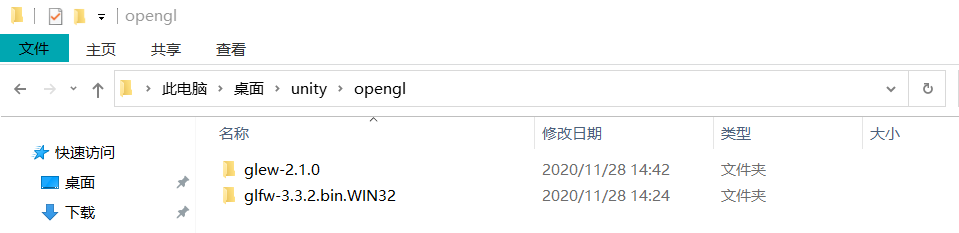


这一小段代码展现了你以后使用OpenGL时常见的工作流。我们首先创建一个对象，然后用一个id保存它的引用（实际数据被储存在后台）。然后我们将对象绑定至上下文的目标位置（例子中窗口对象目标的位置被定义成GL\_WINDOW\_TARGET）。接下来我们设置窗口的选项。最后我们将目标位置的对象id设回0，解绑这个对象。设置的选项将被保存在objectId所引用的对象中，一旦我们重新绑定这个对象到GL\_WINDOW\_TARGET位置，这些选项就会重新生效。

尝试搭建opengl环境

书上的搭建glfw以及cmake的方式报错



直接在vs/c++导入这两个包的include和lib

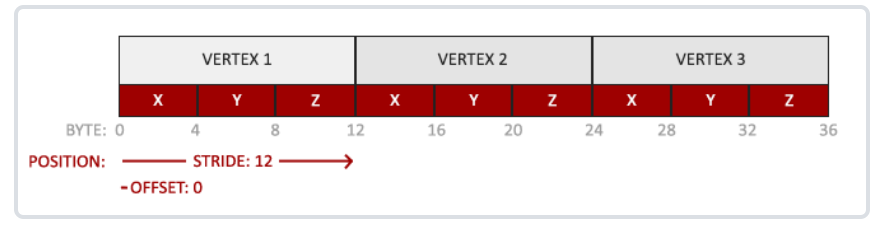
双缓冲(Double Buffer)

应用程序使用单缓冲绘图时可能会存在图像闪烁的问题。 这是因为生成的图像不是一下子被绘制出来的，而是按照从左到右，由上而下逐像素地绘制而成的。最终图像不是在瞬间显示给用户，而是通过一步一步生成的，这会导致渲染的结果很不真实。为了规避这些问题，我们应用双缓冲渲染窗口应用程序。前缓冲保存着最终输出的图像，它会在屏幕上显示；而所有的的渲染指令都会在后缓冲上绘制。当所有的渲染指令执行完毕后，我们交换(Swap)前缓冲和后缓冲，这样图像就立即呈显出来，之前提到的不真实感就消除了。

* glfwSwapBuffers函数会交换颜色缓冲（它是一个储存着GLFW窗口每一个像素颜色值的大缓冲），它在这一迭代中被用来绘制，并且将会作为输出显示在屏幕上。

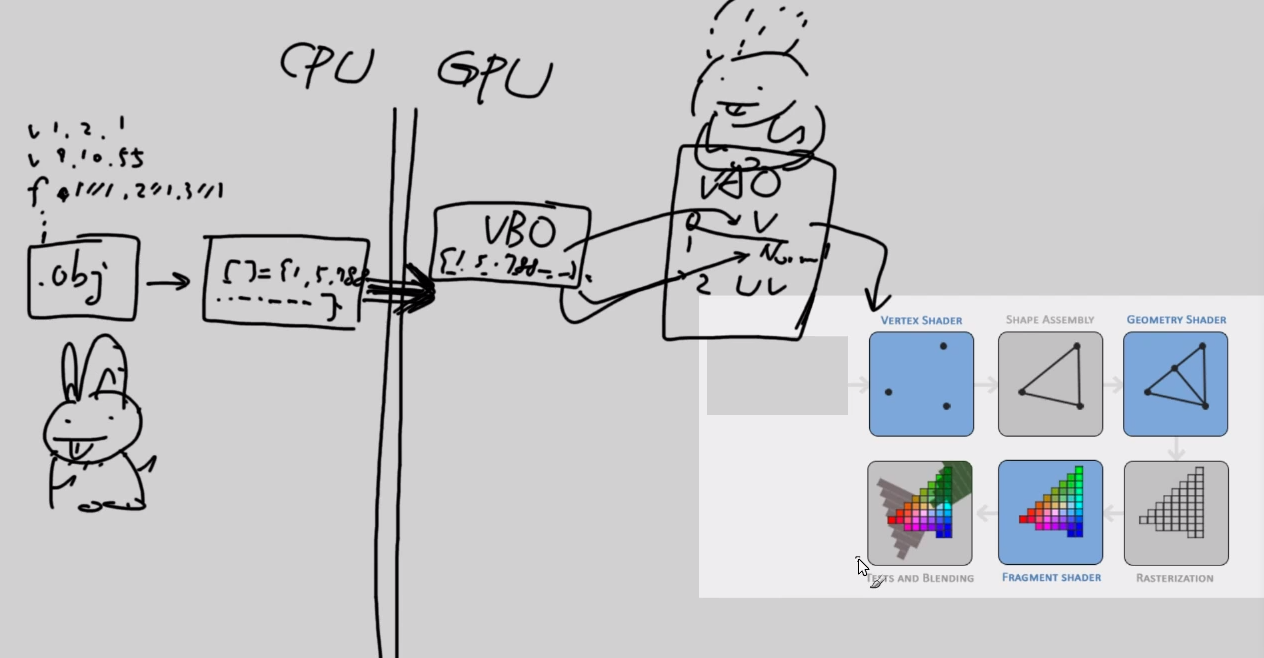
链接顶点属性

顶点缓冲数据会被解析为下面这样子：



* 位置数据被储存为32位（4字节）浮点值。
* 每个位置包含3个这样的值。
* 在这3个值之间没有空隙（或其他值）。这几个值在数组中紧密排列(Tightly Packed)。
* 数据中第一个值在缓冲开始的位置。





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

glBufferData是一个专门用来把用户定义的数据复制到当前绑定缓冲的函数。它的第一个参数是目标缓冲的类型：顶点缓冲对象当前绑定到GL\_ARRAY\_BUFFER目标上。第二个参数指定传输数据的大小(以字节为单位)；用一个简单的sizeof计算出顶点数据大小就行。第三个参数是我们希望发送的实际数据。

第四个参数指定了我们希望显卡如何管理给定的数据。它有三种形式：

* GL\_STATIC\_DRAW ：数据不会或几乎不会改变。
* GL\_DYNAMIC\_DRAW：数据会被改变很多。
* GL\_STREAM\_DRAW ：数据每次绘制时都会改变。

