Programmable hardware

绝大多数显卡允许部分GPU pipeline能被开发者改变

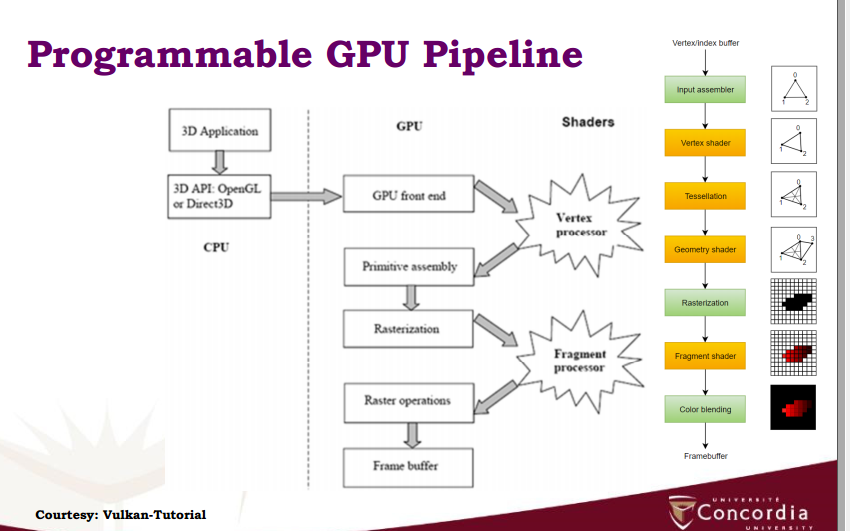
这些programmable parts是

1.vertex processor

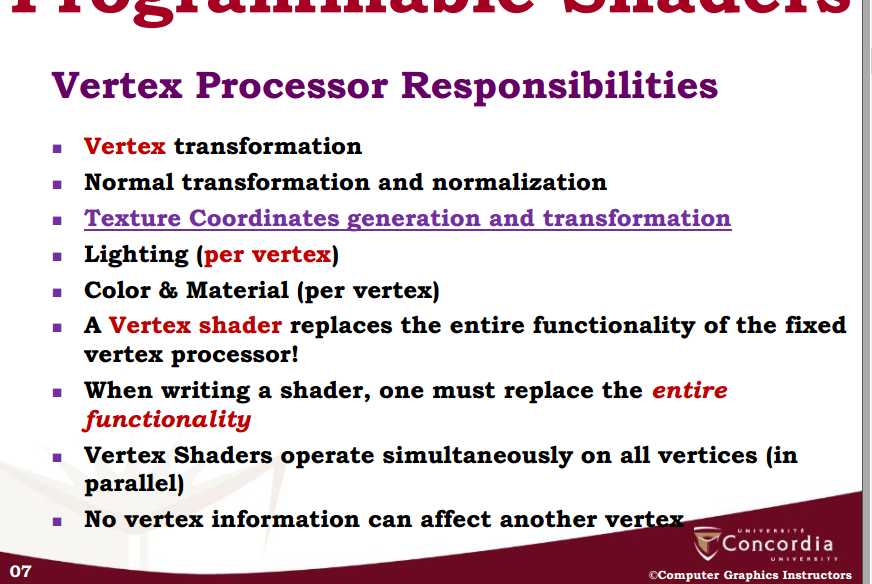
2.fragment (Pixel) Processor

Opengl还允许一些别的part

这些program被称作shaders



GPU通过vertex processor将VERTEX在坐标系中定位，光栅化以后又用fragment 上色



VERTEX PROCESSOR作用

1.Vertice实际坐标的转换

2.纹理坐标生成与转换

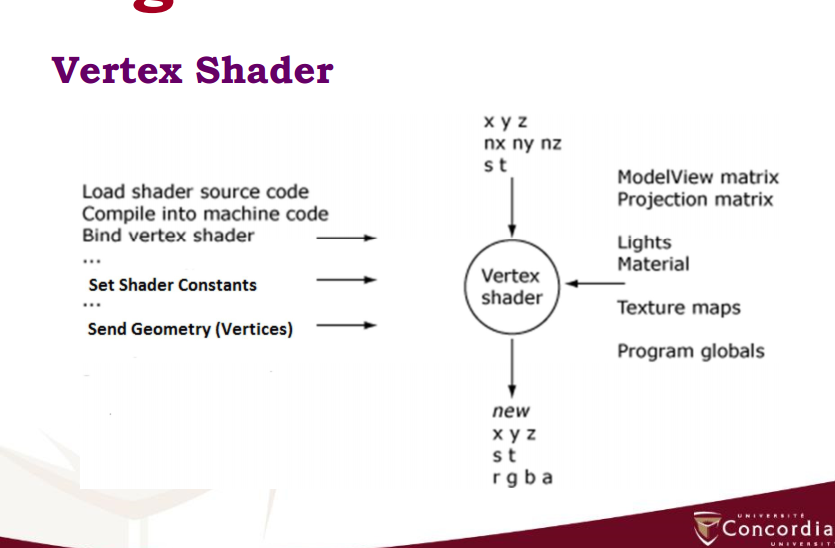
3.Lighting （每个 vertex）

4.每个vertex的颜色与material

5.一个vertex shader会完全替换默认vertex processor

6.vertex shader同时对所有vertice生效（平行的）

7.vertex的Information不会影响到其他vertex



Fragment processor

只有在光栅化vertex data以后才对像素格进行操作

允许以下programming

1.对interpolated vertex values 的操作（PHONG，在下节lecture）

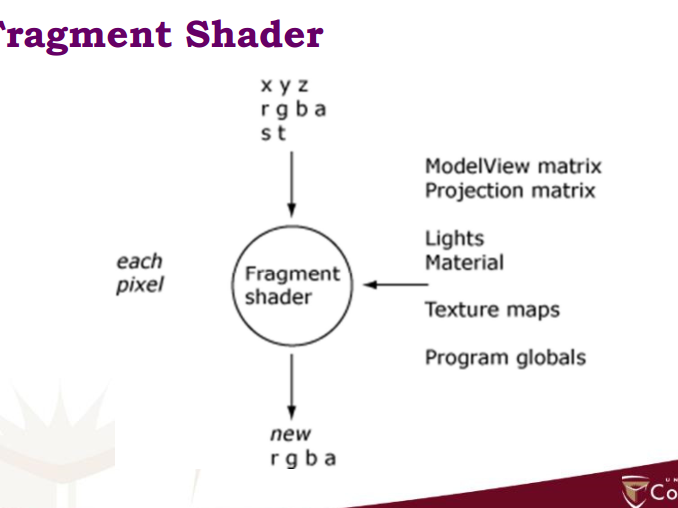
2.纹理的进入以及应用

3.fog

4.上色

所有的fragment shader平行运行

同样的没有像素格可以进入另一个像素格

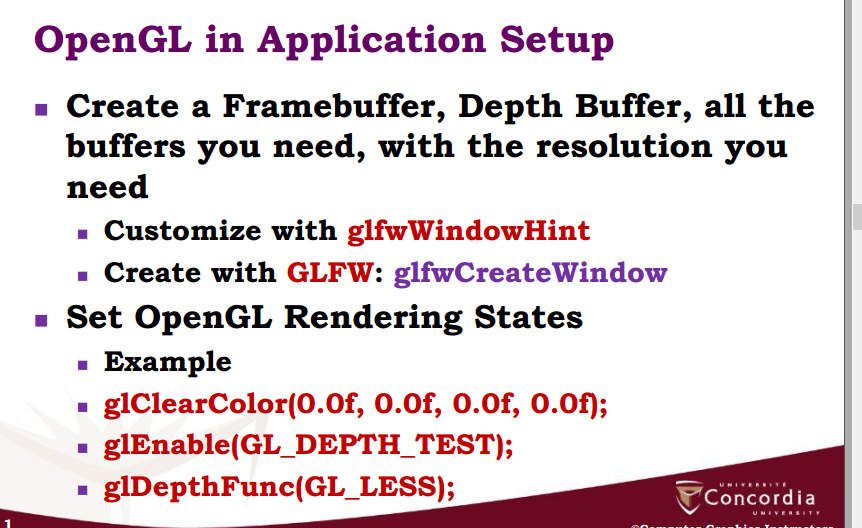


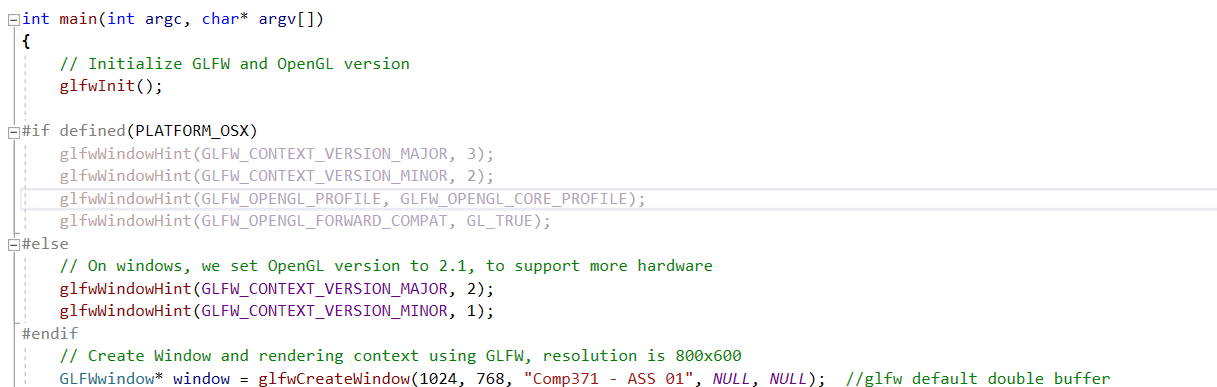
Opengl应用程序设置

1.使用glfwWindowHint来customize//按照要求定制

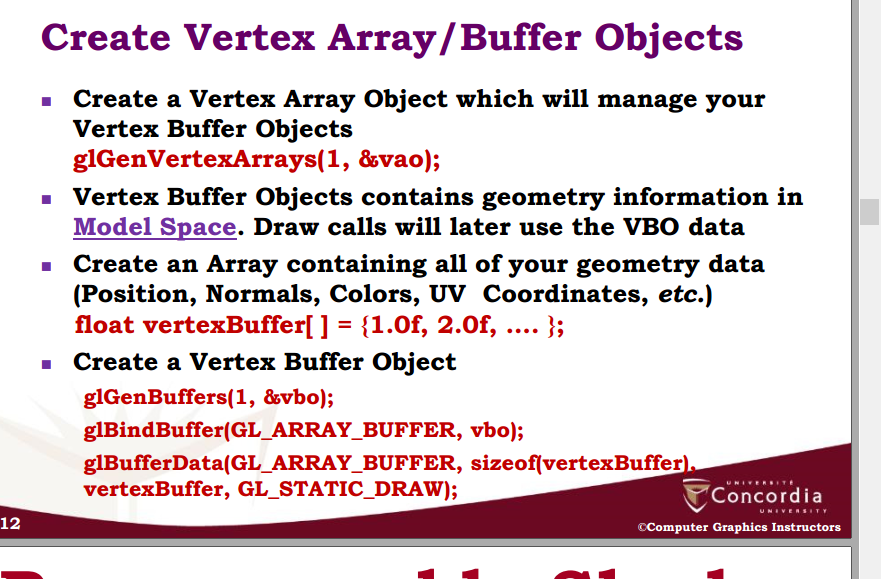
使用glfwCreateWindow创造窗口

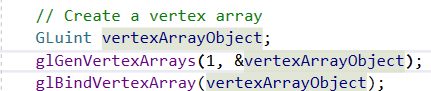
2.使用glClearColor glenablE等来设置opengl rendering states 绘画状态

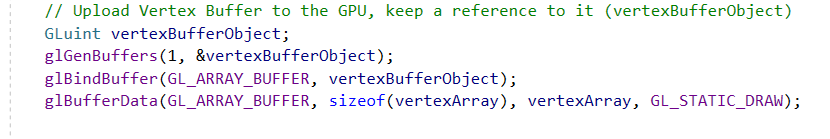




创造vertex array与buffer objects//VAO VBO



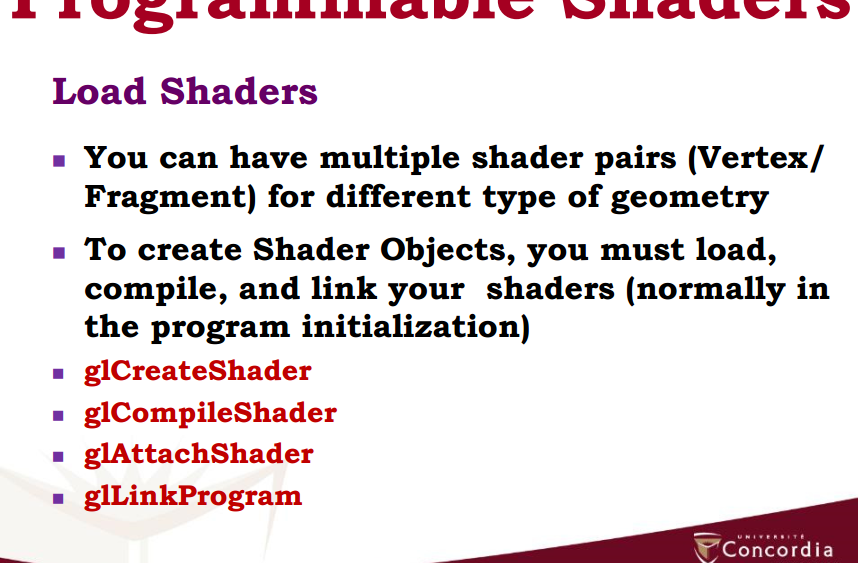




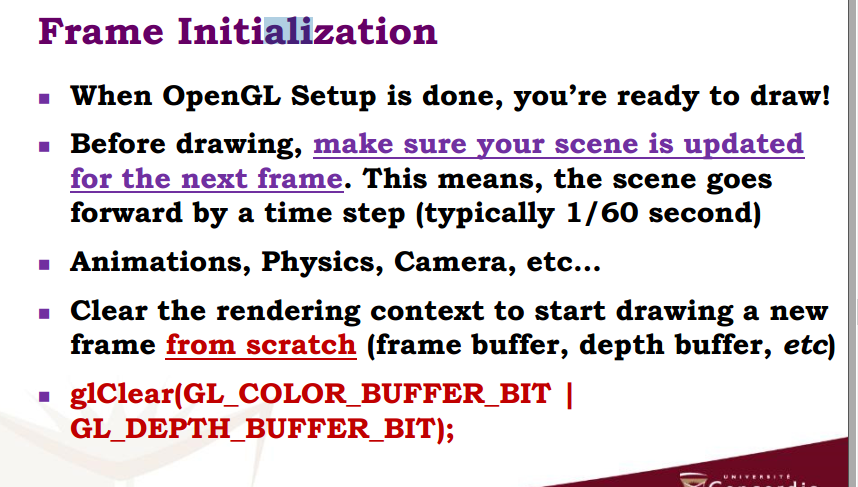
用一个float array 或vec3 array来包含vertice信息

Vbo同理

为了创造一个shader，你必须load ,compile, link….

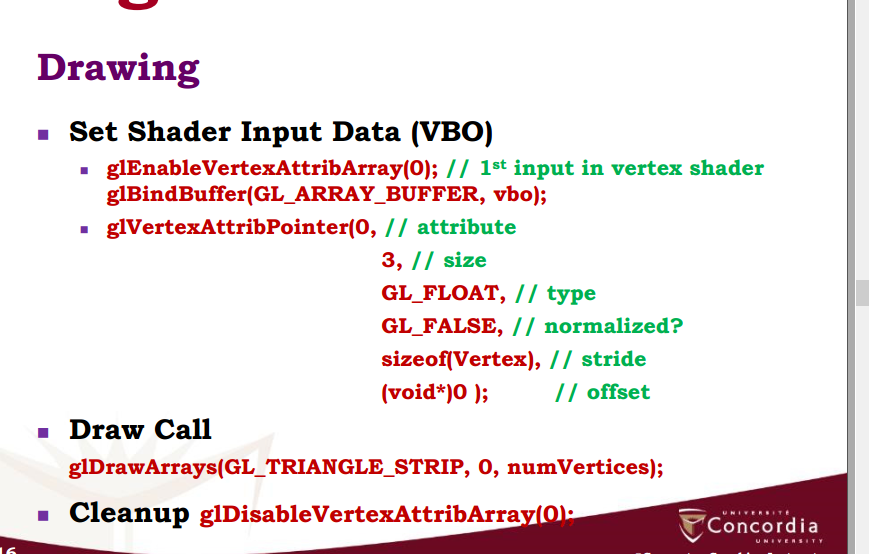


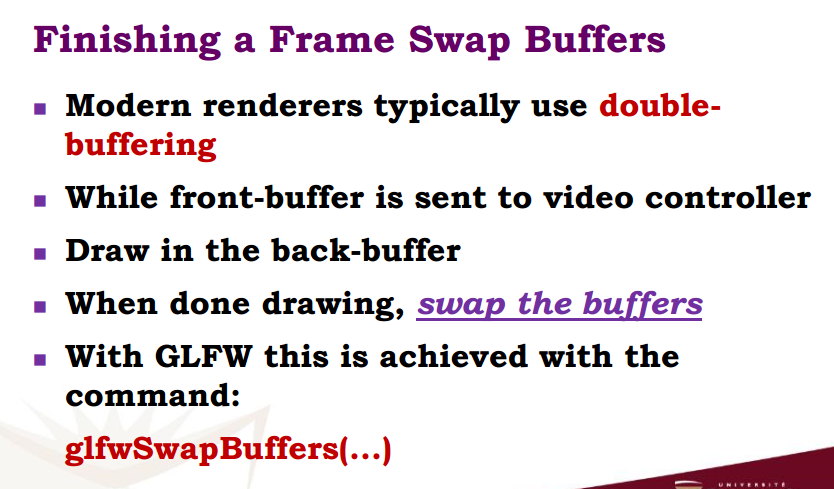
然后在draw的时候要确认每一帧都清除了rendering context

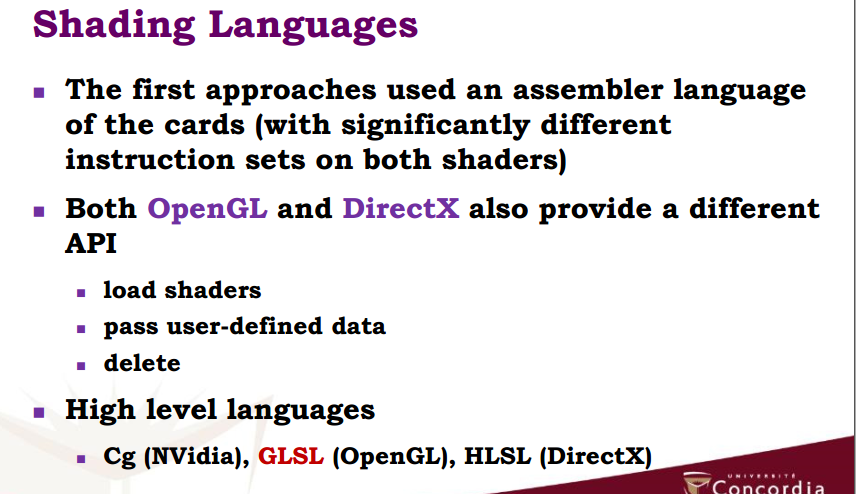


Drawing，首先使用shader，然后设定shader内容，再bind vao

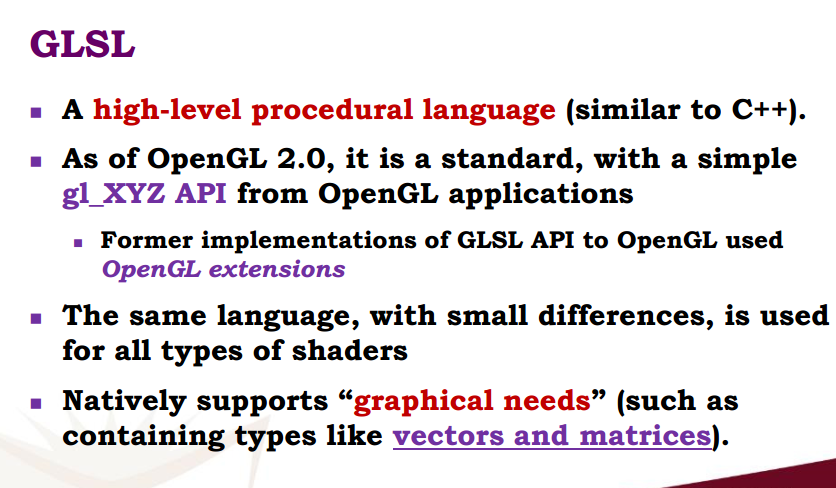








Shading language就是能够改变显卡pipeline的language，opengl 就是GLSL



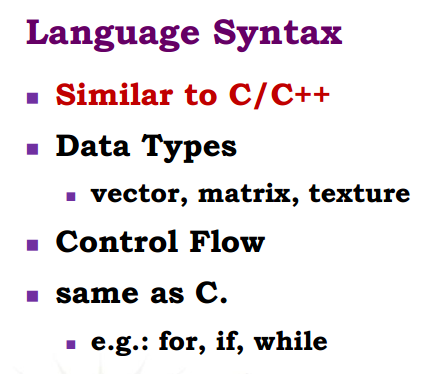
GlSL是一个high-level procedural language

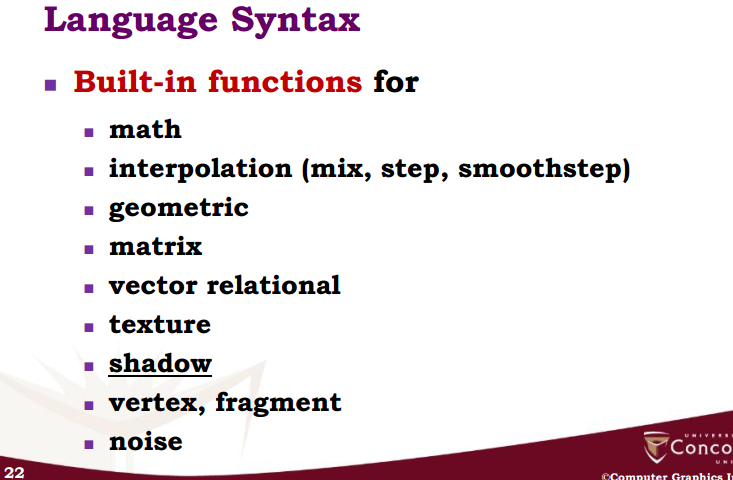
本地支持graphical needs 例如拥有vec mat等type

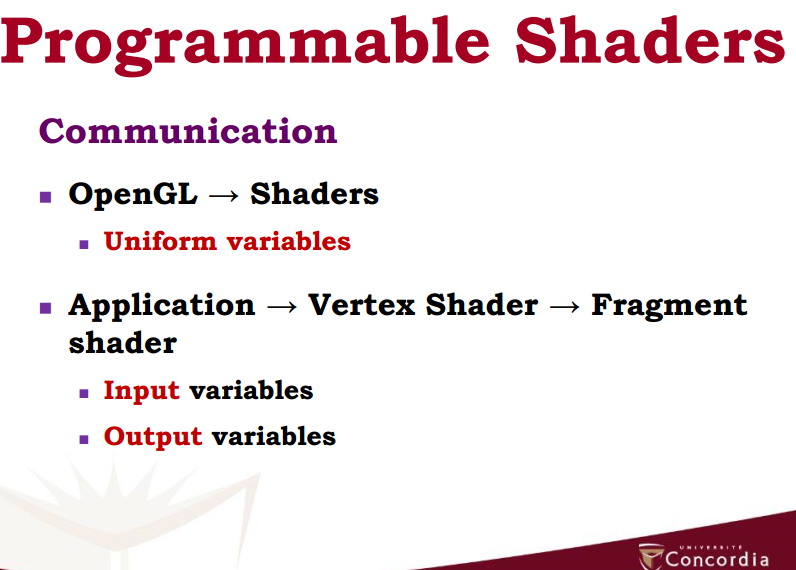


使用GLSL只要

Run glewinfo.exe





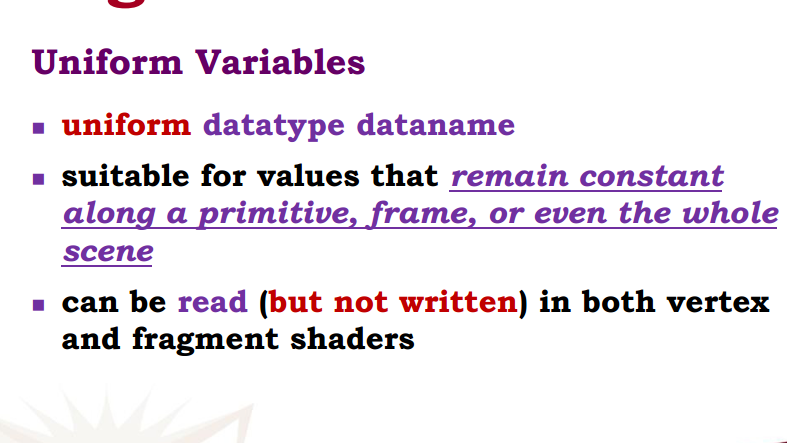


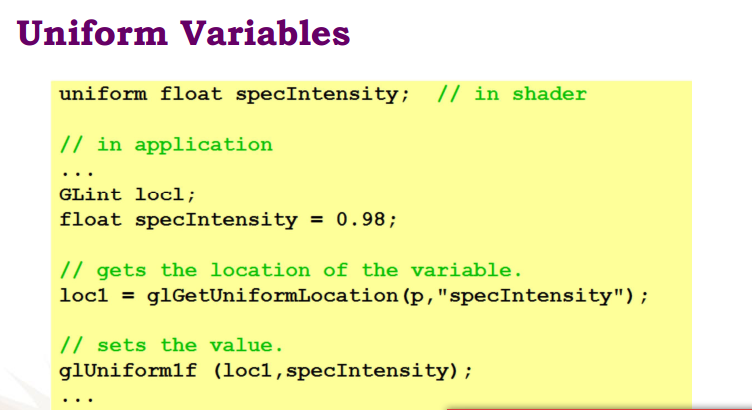
Opengl 的communication， 一共有三种variable

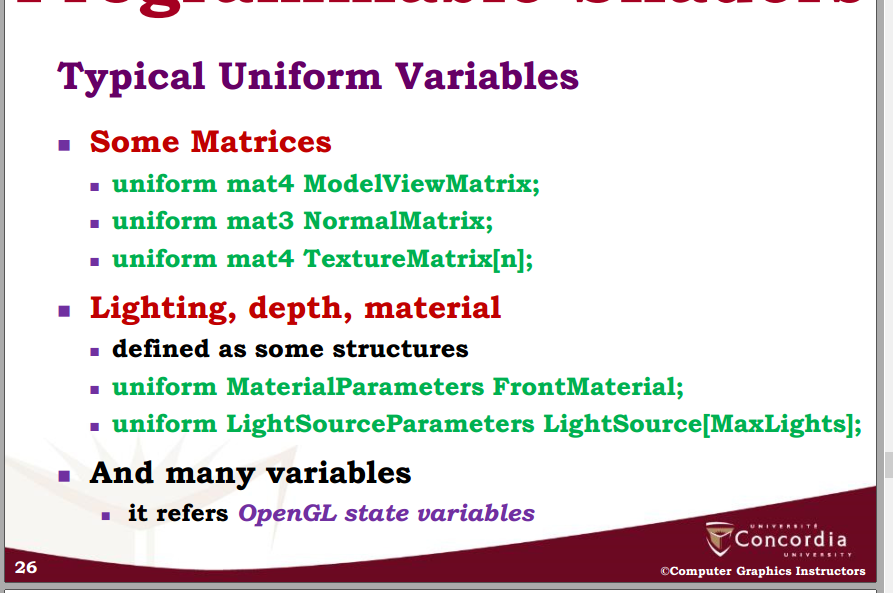
Uniform variable

格式是uniform datatype dataname

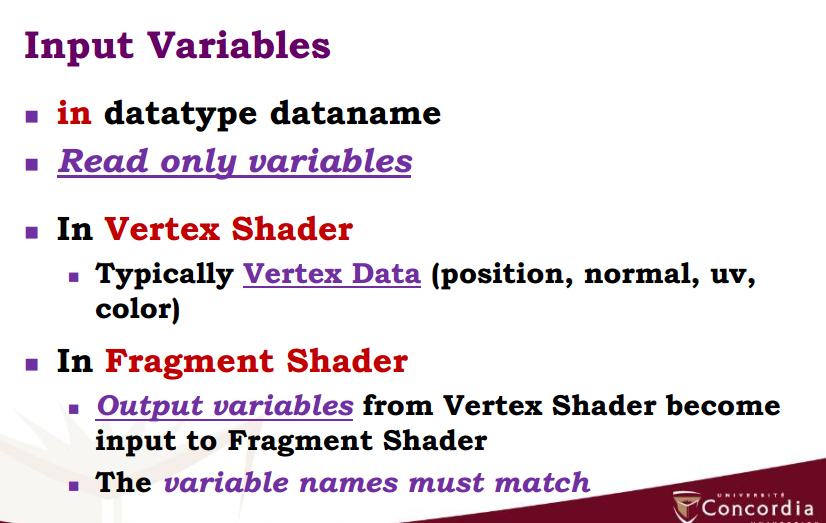
你需要这个value在一个primitive,frame里做一个常量，用uniform variable， 两个vertex shader 与fragment shader都能read，但不能write







Input variable

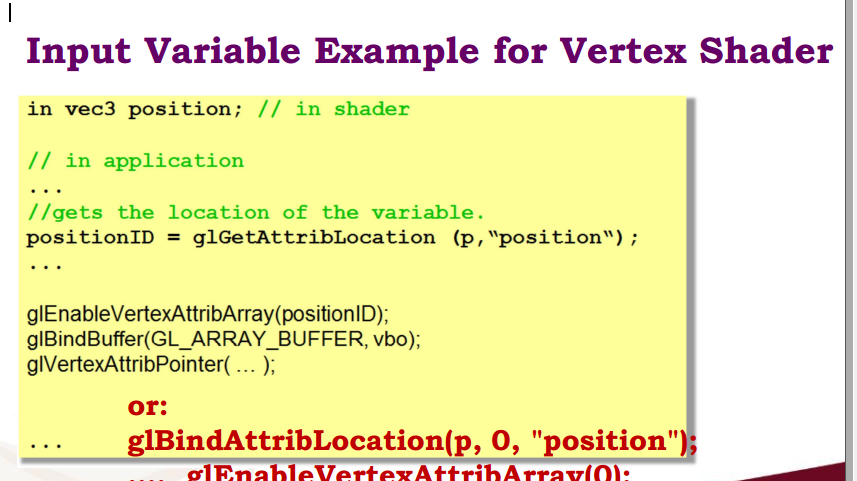


格式是in datatype datatyme

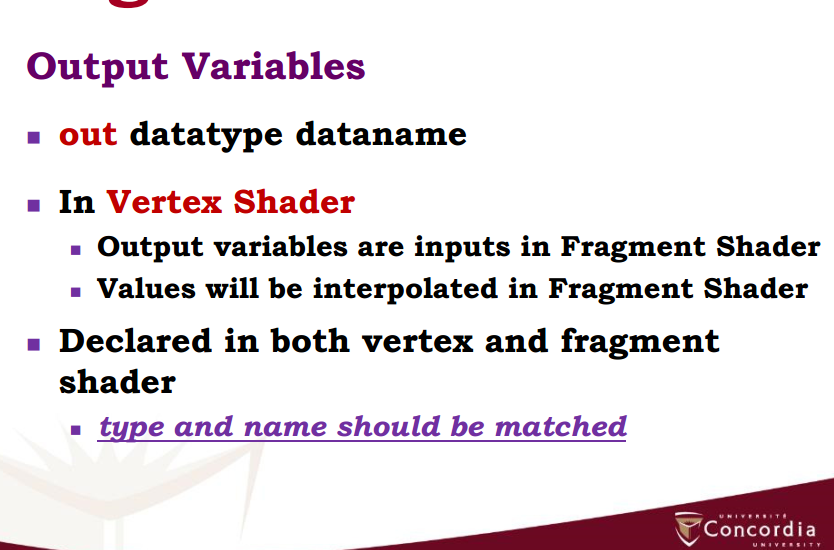
只能阅读

在vertex shader里 通常是vertex data

在fragment shader里是vertex shader的 output变成了fragment shader的input



Output variables



在vertex shader里

Output variable就是fragment shader的input

值会插入到fragment shader中

两个shader都要declare

