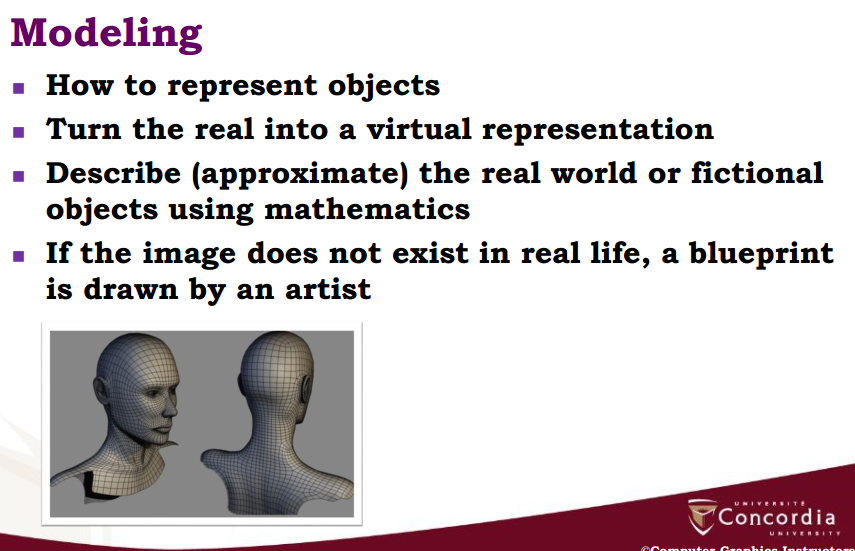
Modeling 建模

object表现的方式，把object表现成实际视觉图像

用数学来描述真实或虚构的物体

如果这个图像生活中不存在，创作者也可以画出蓝图



Animation动画

Rendering, 表现，渲染：就是给modeling上色，打光阴影，细节啥的

image processing图像处理

scene completion场景完成

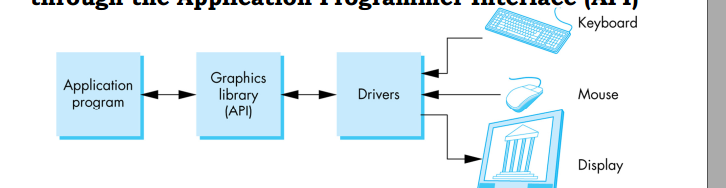
高动态范围high dynamic range:增加明暗比

LECTURE 2

OpENGL 就是一个graphics API, API就是一组function

API Application Programmer Interface：应用程序编程接口,是一组function，用来指定我们生成一张image所需要的东西：objects ,viewer, light source, material

Api还提供了输入输出：接受keyboard mouse,输出给display



OPLENGL Library结构

1. OpenGL(GL):核心图形

那些以gl开头的function

1. OpenGL Framework Library (GLFW):让编程者可以创造窗口，并且同时管理了input device和event

用glfw开头的function

1. OpenGL Extension Wrangler Library(GLEW) :提供了一种run-time机制，决定目标平台支持什么OPENGL EXTENSION

用glew开头的function

1. GLM

Opengl mathematics，数学向的function C++ LIBRARY

Immediate-mode graphics 立刻生效的图形：在V3.1前的版本

Primitives基本参数并不存储在系统里，而是更倾向于尽快的直接经过系统来达到可能的rendering

每次在应该用程序里面指定一个顶点，他都会被送往GPU

每次场景都会被重画

Retained mode graphics 3.1版本开始

我们希望尽量减少CPU GPU之间的数据传递

解决方案，

把所有的VERTEX与ATTRIBUTE都存在array里， 然后再把一整个array发送给GPU这样来立即render

这几乎是ok的，但是每一次我们需要render都会send一次array

新解决方案

把所有vertex与atttribute都存在array里，然后把这个array存储在GPU里，这样就可以多次render

Graphics Pipeline图形流水线

一个scene场景包含多个object

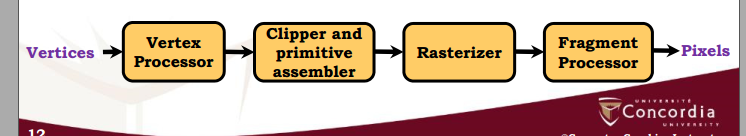
每一个object有一组geometric primitives组成 //几何元件

而每一个geometric primitive由一组vertices组成

一个复杂场景有Million个vertices

必须要通过相同的方法来处理每一个vertice，这样才能在framebuffer里形成一个图像

Pipeline包括了四步生成图像的过程



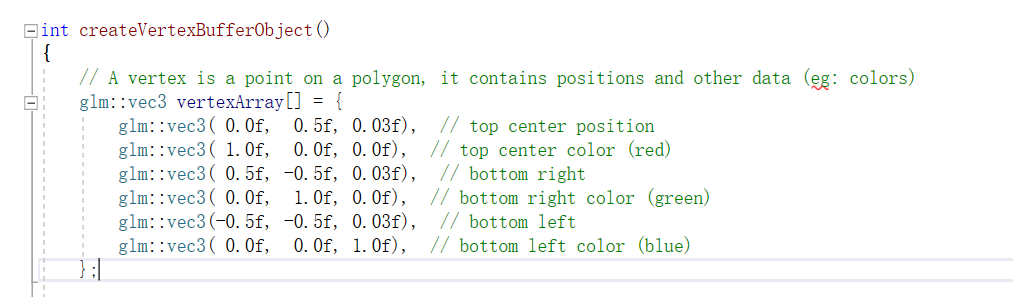
Graphic Pipeline由：Opengl，显卡驱动与显卡硬件来Implement。 程序员并不用主动Implement it。

我们也能用特殊程序叫做shader着色器来控制pipeline

Vertices

Vertices用world coordinates来描述（xyz坐标系）

包含很多信息，position,color,normal….



Vec就是vector的意思，可以用来描述position，color等

Vertex Processing，两个主要功能

1.进行坐标变换:coordinate transform，

Transformations是把一个map points从一个地方转换到另外一个地方的function,每一个地方都是基于world coordinate的

几乎渲染管道的每一部都会包含坐标变化

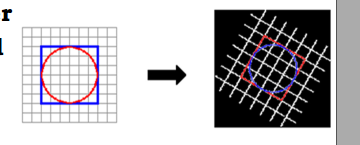
转换是理解3D电脑图形的核心

最好执行转换的方法就是matrix operations矩阵操作

每一个vertex的位置都会经历一系列的转化，，从model coordinate system到Screen coordinate system，

Model上的每个vertex用transformation matrix转化

这个matrix 叫做model-view-projection matrix，会考虑到model在世界中的位置，摄像机位置，以及projection parameter投影参数



2.计算每个vertex的颜色

根据真实的灯光模型计算颜色，该模型考虑了物体的表面特性和场景中的特征光源

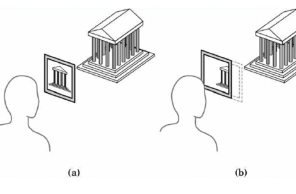
Clipping and primitive assembling

没有一个图片系统可以一次看见整个世界-》所以使用clipping volume来描述可见的部分 //剪裁体积

建材通常是自动完成的，通过你定义一个观察口

Clipping作用在primitives上

Primitive assembly必须先做好，//我们必须先设置好primiitive，即把vertices转化成primitives



有可能的情况“

1．事物还是可见，继续处理”

2. 不可见，停止处理

3. 部分可见， 处理可见的部分

在clipping volume部分的object的投影会变成最终的图片

投影也是transformation的形式

Rasterization 光栅化 //找点

­­­­­­­那些被clipper裁剪过后的primitives还是以vertices的形式表示的，我们必须将他进一步处理，来生成frame buffer里的像素//帧缓冲器

例子：三个vertices能描述一个纯色的三角形,rasterizer会决定那些像素点会在这个多边形里

把vertices转成window coordinates 二维坐标

----

最后输出是每个primitive的一组fragments 碎片

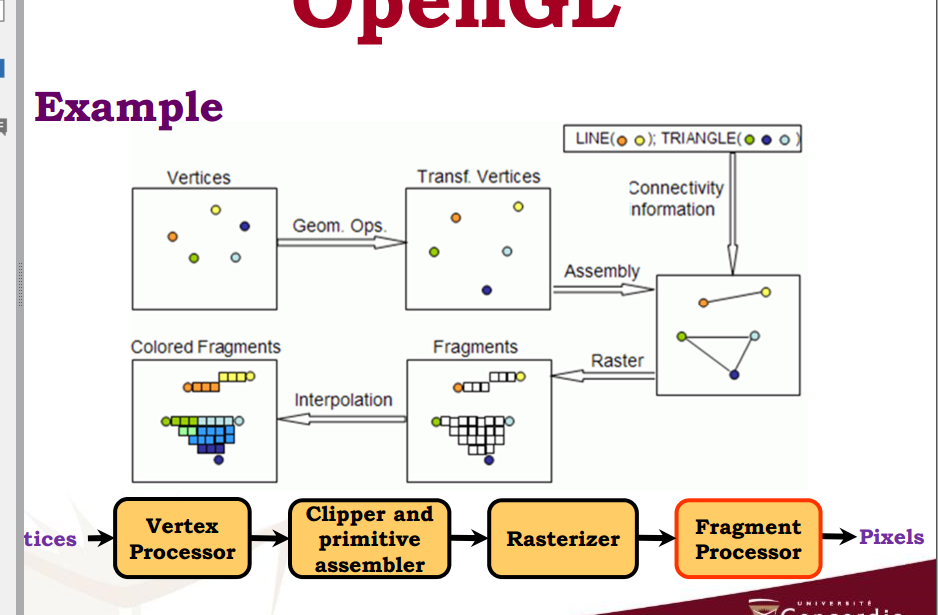
一个fragment就是一个潜在pixel，包含信息包括颜色，位置，深度等

Fragment Processing 碎片处理 //上色

光栅化生成的只是fragments, fragment processing这一步将fragment正式转化成pixel，在frame buffer上

转化过程的例子：决定哪些fragment是不可见的，颜色可能被改变因为纹理和凹凸，半透

明效果通过将fragment的颜色与pixel的颜色相连//意思就是不是单纯的填充纯色



Vertices就是有颜色的端点，通过vertex processor重新排列，然后clipper and primitive assembler这一步是先组装基元primitive，再切割

Raster就是光栅化，决定用哪些像素点来填充，生成各个带有信息但实际没真正上色的fragment, fragment processor就是正式表示，做出渐变色等等

Opengl除了function，还需要两个东西来表现出一个object，

Geometry几何形状—PRIMITIVES //primitives就指各种最基本的形状，点线面啥的

Appearance外观->attributes

Primitives

绝大部分API都支持primitives

Point(1D object)

线段Line segments （2D project）

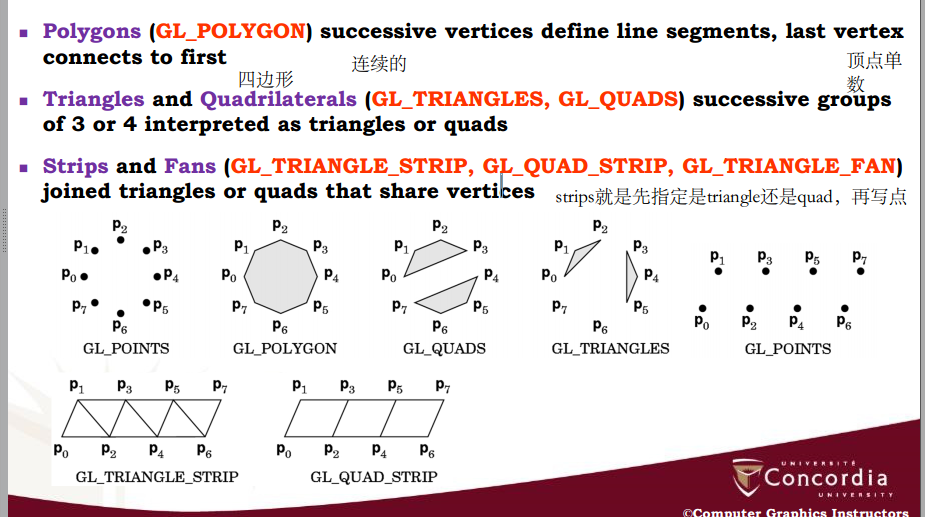
Polygons 多边形(3D object)

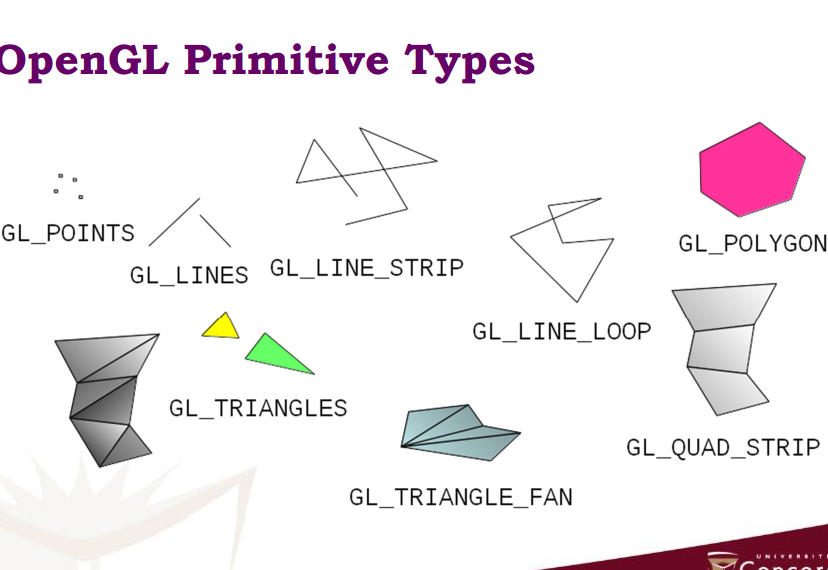
一些曲线或者表面例如quadric二次曲面，参数多项式

Primitives通过指定vertices来描述

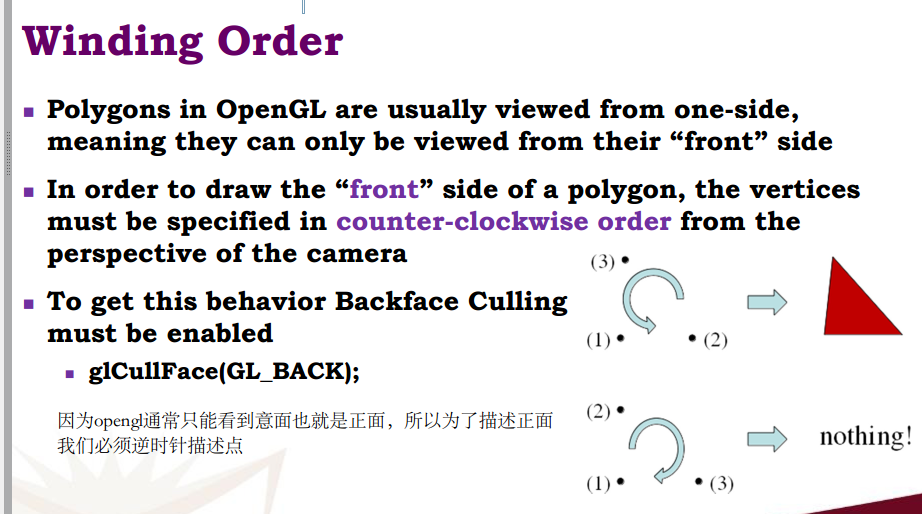
显卡支持多种不同type得primitives，我们必须要选择正确的

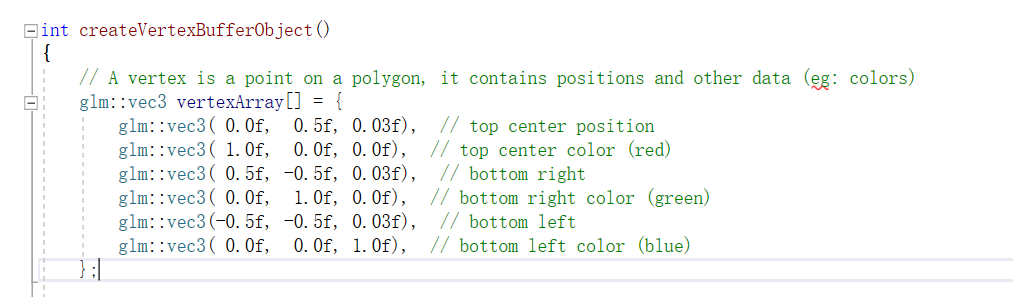
OPENGL的primitive types





Fan的特点是所有人共享一个端点



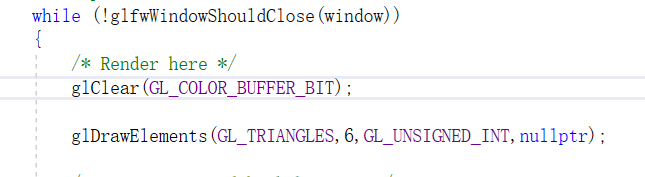


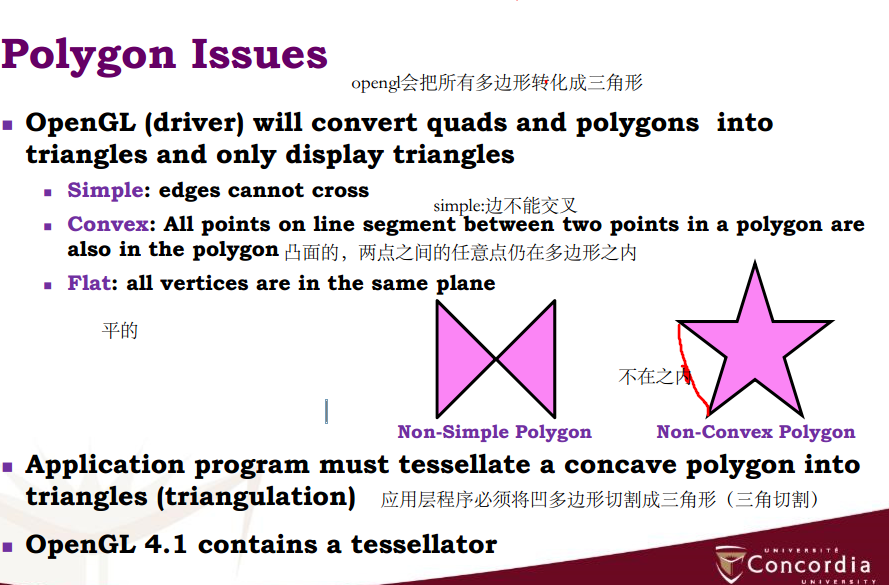
所以描述verticearray的时候要逆时针

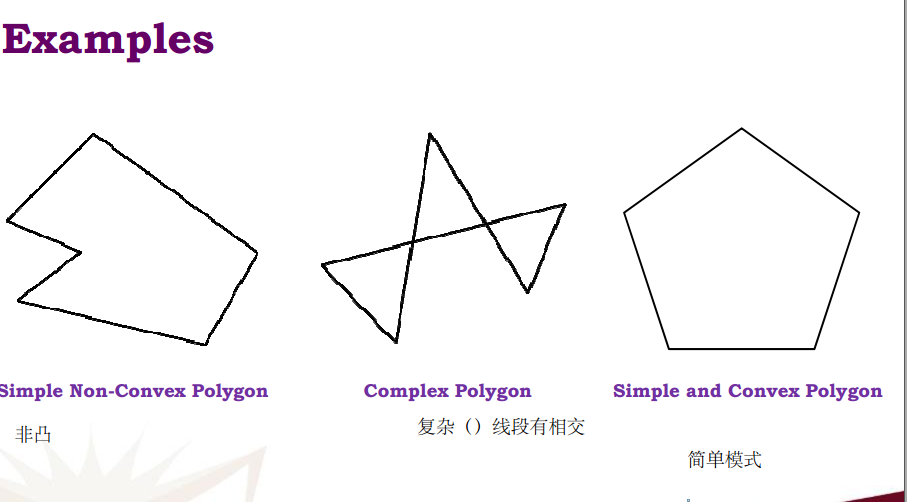


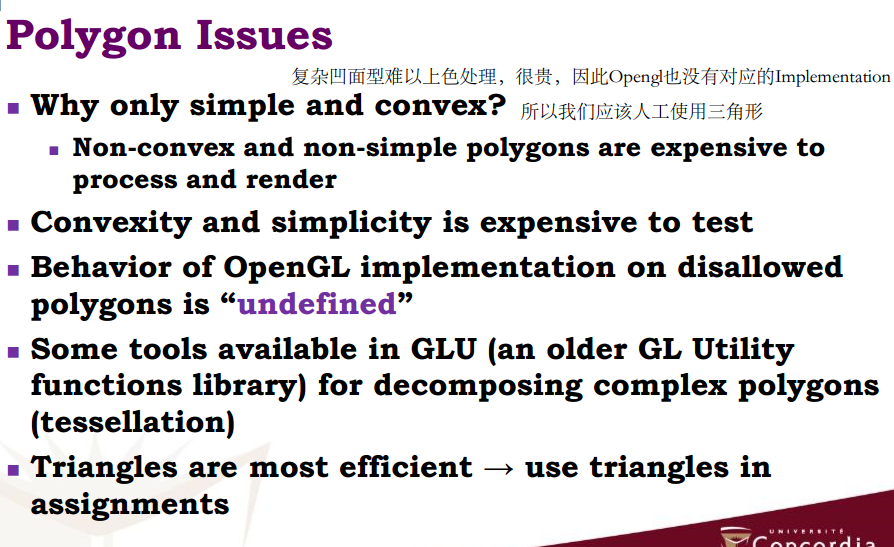
Primitive一般用在GldrawArrays

如果用了index buffer



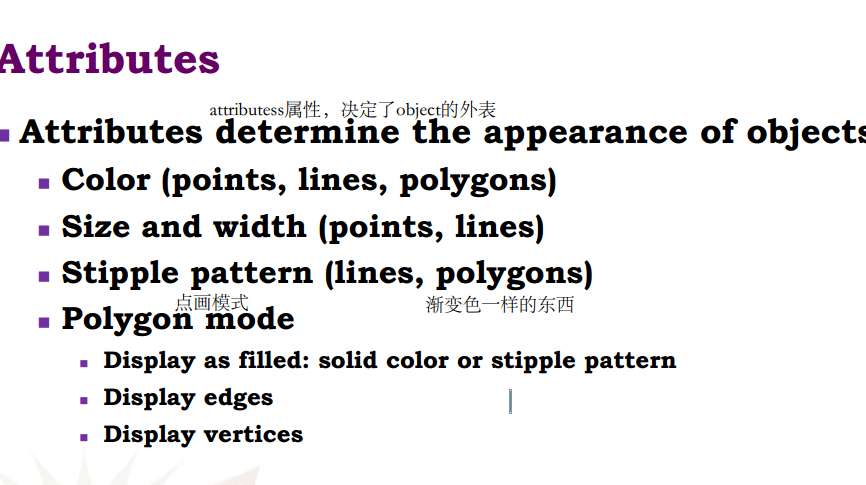


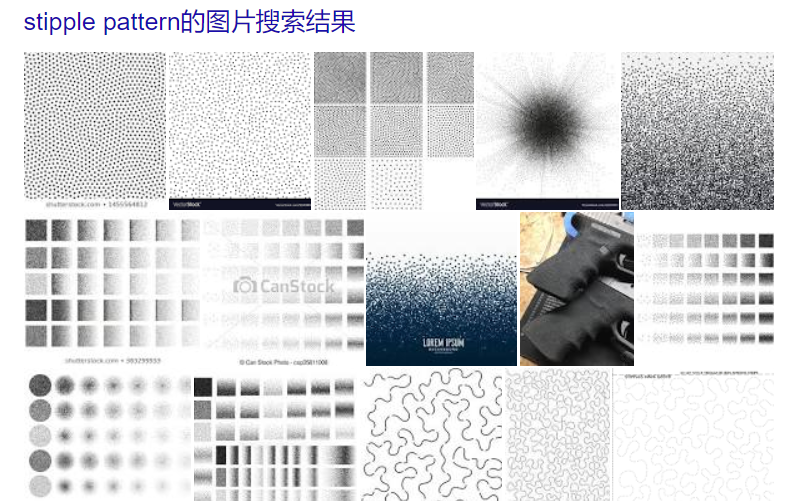


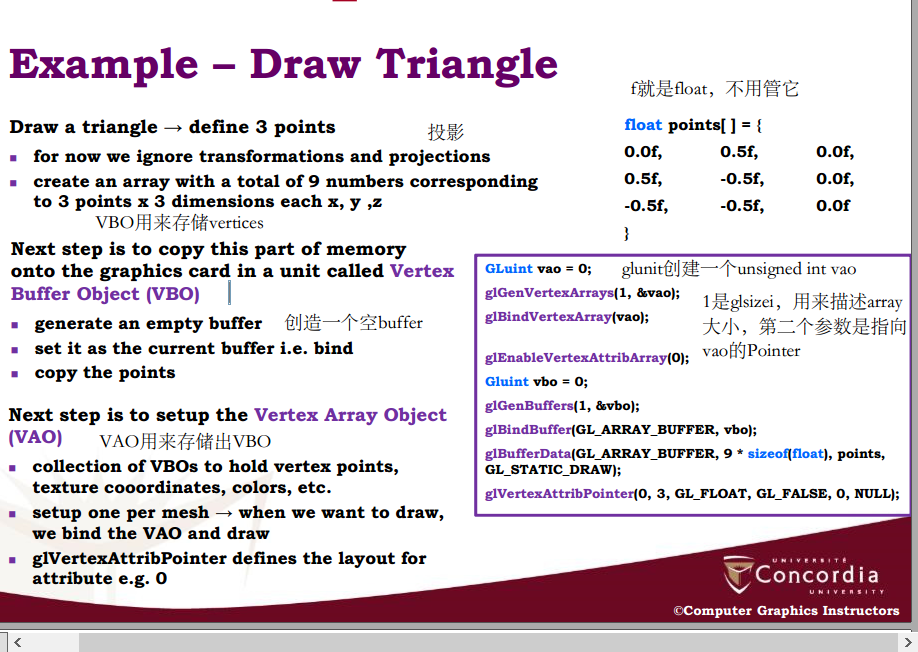


Primitive就是点的排列模型，

Attribute就是除了Position以外的vertex属性

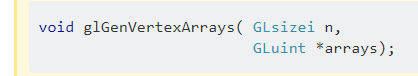






先说明什么叫vertex array，就是一个基本primitive，这里Point[]就是一个三角形，

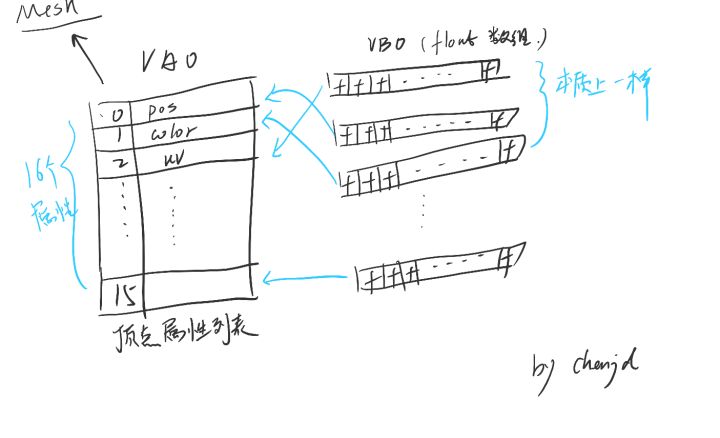
1是glsizei，用来描述array大小，第二个参数是指向vao的Pointer，然后这个pointer一定要指向一个GLUINT,这才有了第一步GLUINT vao



换句话说，1是尺寸，&vao中的vao是名字，这个vao是用来存储vertex array object的地方

GLBINDVERTEXARRAY， 将哪个作为当前VAO

这些都是初始化的过程



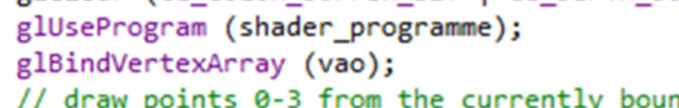
Shader

分为vertexshader与fragment shader

前者描述位置

后者描述颜色

两者合并组成program



在bind vao前使用shader