这个版本的新东西有哪些（版本：4.3）？

Functions + shading language

描述了渲染管线的各个阶段。

在阅读指南前应该了解哪些？

第一章

读完本章后应该能够：

1）描述opengl的用意，在创建一幅图像时哪些可以做，哪些不能做

2）区分出opengl程序中常用的结构

3）列举出opengl渲染管线中各个渲染阶段

本章主要包括以下部分：

1）解释啥是opengl，什么可以做，什么不可以做和如何做

2）提供一个first look at opengl

3）介绍常用的命令

4）介绍渲染管线

5）详细分析一个例子

Opengl渲染管线大大概过程：

1）指定要绘制的几何图元的原始数据

2）调用shaders来处理输入进来的数据，来确定数据的位置、颜色和其他渲染属性

3）将以上输入的数学描述转换为屏幕上对应位置的fragmeng，并最终确定颜色和位置

4）可能还会进行额外的片段处理，如确定是否可见，混合信息等

Opengl是一个c-s系统，我们写的东西是在客户端，并且opengl的实现是由我们电脑的图形硬件厂商提供，有一些opengl的实现（比如那些和windows系统关联的），它们的客户端和服务端会通过network被执行在不同的机器上，在这种情况下，客户端会发行它们的opengl命令，并以特定的protocol传输到服务端，从而处理最终的图像。

Opengl应用程序的干的两件事：

1）初始化与待渲染对象相关的状态

2）渲染它们

Rendering是一个常见的概念，Opengl只是一个rendering system的例子而已

什么是shaders？它是一个专门指定硬件要执行的函数的东西

有4个shader stages可以调用，最常用的vertex shaders，用来处理顶点数据，fragment shaders用来操作光栅化后的片段，以上2个阶段是opengl必经阶段。

屏幕上的图片是由像素组成的，在显示中，像素是最小的可见单元，它们被存储在我们称之为framebuffer的内存块中，并由硬件来管理。

简单来说，一个典型的opengl程序的4步：

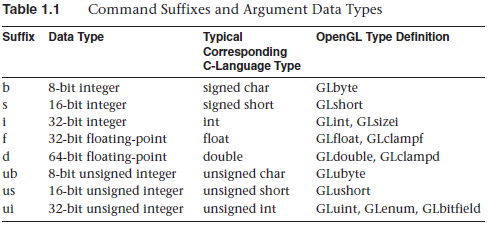
首先：include合适的头文件，声明要用到的全局变量和其他结构体等。

然后：init()。Init内容包括设置程序中需要渲染的数据，通常包括vertex或image数据，再指定shaders，再LoadShader到程序中，再将之前的数据关联到刚load的shader program中。

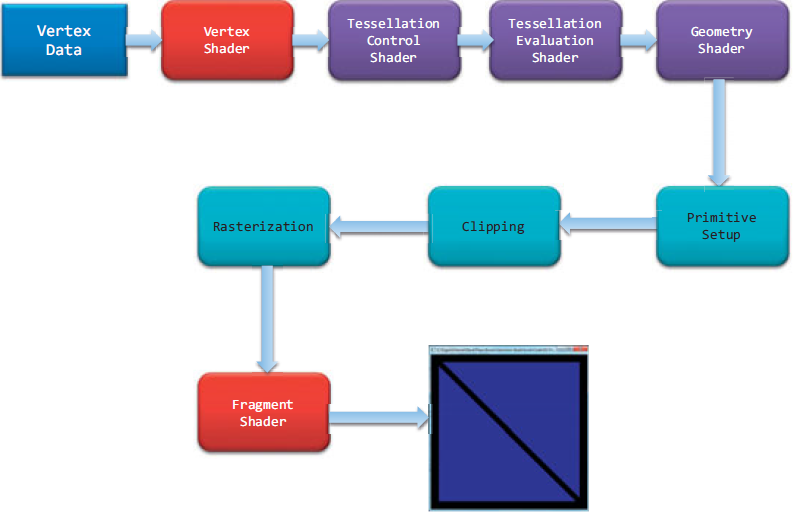
然后：display()。开始渲染，几乎所有的display都有相同的3个步骤：先glClear()，再发出渲染请求，再将渲染结果呈现到屏幕上。

最后：由main来创建一个窗口，并调用init,display，加入event loop。

最好是用opengl定义的类型，可以防止切平台时出现类型不匹配的问题。



渲染管线：



准备发送数据给opengl

Opengl要求所有的数据必须存放在由opengl服务端管理的buffer objects中，通常由glbufferdata()函数初始化，注：只是初始化并未发送，此时数据应该还在客户端。

发送数据给opengl

初始化后，就可以通过glDrawarrays()来请求渲染几何图形了，此时才真正将data发送到服务端。

Vertex shading

对于每一个需要被绘制的vertex，在启动绘制指令后，vertex shader都会被调用，根据vertex的数据进行处理。通常，一个应用程序可以拥有多个vertex shaders，但同一时刻，只能有一个被激活。

Tessellation shading（可选过程）

在vertex shading处理完vertex的数据后，tessellation shader接着处理这些数据（前提是tessellation shader是激活的状态），tessellation利用片（patchs）来描述一个对象的形状，并且可能会增加一些几何图元来使得模型看起来更好。Tessellation shading这个过程可能会用到2种shader来操作这些片数据（patch data）并产生一个最终的形状。

Geometry shading（可选过程）

Geometry shading允许对单个的图元进行额外的处理，包括创建一个新的图元

Primitive assemble

以上的阶段都是对vertices进行加工，本阶段将将vertices组装到几何图元中，为剪裁和光栅化做准备。

Clipping

修改vertex使得没有任何pixels在viewport之外，该阶段由opengl自动处理。

Resterization

Clipping后，数据会被传至resterizer进行光栅化，并生成片段（fragment）,可以认为，一个fragment就是一个候选pixel，它有一个home在framebuffer中，此时这个pixel仍可能会被reject，甚至永远不被渲染。详见下面2个阶段。

Fragment shading

这是最后一个渲染管线中可控的阶段。这个阶段中可以利用一个fragment shader来确定片段的颜色，也有可能中断管线的进程（当片段不通过测试时），这个过程称为fragment discard。

一句话解释vertex shading和fragment shading：

Vertex shading(包括tessellation和geometry shading)决定图元的位置；fragment shading决定图元的颜色。

Per-fragment operations

这是对单个的fragment进行额外的加工，在这个阶段，一个fragment的可见性会被深度测试和模板测试检测，如果fragment通过了所以打开的测试，将会被写入framebuffer，并更新该像素的颜色，或者，如果起用了混合处理的话，fragment的颜色将会和当前像素的颜色进行混合处理，将得到的新颜色写入framebuffer。

通常，pixel data来自图片文件（虽然也可以通过opengl渲染得到），且存储在texture map中，通过texture mapping来使用，它允许任意的纹理阶段来查询值。