**opengl的FBO**

http://blog.csdn.net/xiajun07061225/article/details/7283929/

**OpenGL默认的帧缓存包括：颜色缓存、深度缓存、模板缓存、累积缓存。**

**OpenGL扩展帧缓存（即FBO）包括：颜色缓存、深度缓存、模板缓存。**

在opengl渲染管线中，几何数据和纹理经过多次转化和测试，最后以二维像素的形式显示在屏幕上。opengl管线的最终渲染目的地被称为帧缓冲。帧缓冲是一些二维数组和opengl所使用的存储区的集合：颜色缓存、深度缓存、模板缓存和累积缓存。一般情况下，帧缓存完全由window系统生成和管理，由opengl使用，这个默认的帧缓存称为“window系统生成”（window-system-provided）的帧缓存。

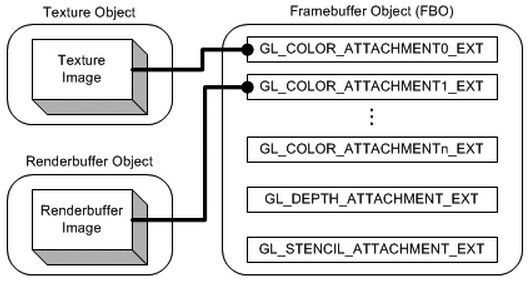
FBO最常见的应用就是：渲染到纹理（render to texture），通过这项技术可以实现发光效果、环境映射、阴影映射等很炫的效果，FBO被推荐用于把数据渲染到纹理对象，相对于其他同类技术，如数据拷贝或交换缓冲区等，使用FBO技术会更高效并且容易实现。

在opengl扩展中，GL\_EXT\_framebuffer\_object提供了一种创建额外的不能显示的帧缓存对象接口。为了和默认的“window系统生成”的帧缓存区别，这种缓存就称为应用程序帧缓存（application-createframebuffer）。通过使用帧缓存对象（FBO），opengl可以将显示输出到引用程序帧缓存对象，而不是传统的“window系统生成”帧缓存，而且，它完全受opengl控制。

类似于window系统提供的帧缓存，一个FBO也包含一些存储颜色、深度和模板数据的区域**（注意：没有累积缓存）**，我们把FBO中这些逻辑缓存称为“帧缓存关联图像”，它们是一些能够和一个帧缓存对象关联起来的二位数组像素。

有两种类型的“帧缓存关联图像“：纹理图像（texture images）和渲染缓存图像（renderbufer images）。如果纹理对象的图像数据关联到帧缓存，opengl执行的是”渲染到纹理“操作，如果渲染缓存的图像数据关联到帧缓存，opengl执行的是离线渲染（offscreen rendering）。

这里要注意的是，渲染缓存对象是在GL\_EXT\_framebuffer\_object扩展中定义的一种新的存储类型，在渲染过程中它被用作存储单幅二维图像。下面这幅图显示了帧缓存对象、纹理对象和渲染缓存对象之间的联系。多个纹理对象或者渲染缓存对象能够通过关联点关联到一个帧缓存对象上。



在一个帧缓存对象中有多个颜色关联点（GL\_COLOR\_ATTACHMENT0\_EXT... GL\_COLOR\_ATTACHMENTn\_EXT），一个深度关联点（GL\_DEPTH\_ATTACHMENT\_EXT）和一个模板关联点（GL\_STENCIL\_ATTACKMENT\_EXT）。每个FBO中至少有一个颜色关联点，其数目与实体显卡相关。可以通过GL\_MAX\_COLOR\_ATTACKMENTS\_EXT来查询颜色关联点的最大数目。FBO有多个颜色关联点的原因是这样可以同时将颜色而换成渲染到多个FBO关联去。这种“多渲染目标（multiple rendertargets，MRT）”可以通过GL\_ARC\_draw\_buffers扩展实现，需要注意的是：FBO本身并没有任何图像存储区，只有多个关联点。

FBO提供了一种高效的切换机制，将前面的帧缓存关联图像从FBO分离，然后把新的帧缓存关联图像关联到FBO。在帧缓存关联图像之间切换比在FBO之间切换要快得多，FBO提供了glFramebufferTexture2DEXT来切换2D纹理对象和glFramebufferRenderbufferEXT来切换渲染缓存对象。

1）glGenFramebuffersEXT(GLsizei n, GLuint \*ids)

该函数需要两个参数：第一个是要创建的帧缓存的数目，第二个是指向村塾一个或者多个ID的变量或数组的指针。它返回未使用的FBO的ID。ID为0表示默认帧缓存，即window系统提供的帧缓存

2）glBindFramebufferEXT(GLenum target, GLuint id)

一旦一个FBO被创建，在使用它之前必须绑定

第一个参数target应该是GL\_FRAMEBUFFER\_EXT，第二个参数是FBO的ID号，一旦FBO被绑定，之后的所有opengl操作都会对当前所绑定的FBO造成影响，ID号为0表示缺省帧缓存，即默认的window提供的帧缓存，因此，在glBindFramebufferEXT中将ID号设置为0可以解绑定当前的FBO

渲染缓存对象

另外，渲染缓存是为离线渲染而新引进的，它允许将一个场景直接渲染都一个渲染缓存对象，而不是渲染到纹理对象中，渲染缓存对象是用于存储单幅图像的数据存储区域，该图像按照一种可渲染的内部格式存储，它用于存储没有相关纹理格式的opengl逻辑缓存，比如模板缓存或深度缓存。

void glGenRenderbuffersEXT(GLsizei n, GLuint\* ids)

void glDeleteRenderbuffersEXT(GLsizei n, const GLuint\* ids)

一旦一个渲染缓存被创建，它返回一个非零的正整数，ID为0是opengl保留值

void glBindRenderbufferEXT(GLenum target, GLuint id)

和opengl中其他对象一样，在引用渲染缓存之前必须绑定当前渲染缓存对象，target参数应该是GL\_RENDERBUFFER\_EXT

void glRenderbufferStorageEXT(GLenum target, GLenum internalFormat, GLsizei width, GLsizei height)