Default Framebuffer

即屏幕，创建OpenGL Context时，准确的说是在ChoosePixelFormat的时候设定了该default framebuffer有什么缓冲区，有多少位。

OpenGL Context创建完成时，默认的渲染目标就是屏幕，当然也可以调用glBindFramebuffer(GL_FRAMEBUFFER,0)来手动地绑定屏幕为当前的渲染目标。（渲染目标：我们想把物体渲染到的地方）。注意glBindFramebuffer的第二个参数0，代表的就是default framebuffer的Handle。

通常，我们在进行一些简单的渲染时，直接渲染到屏幕（即default framebuffer）就足够可以了。但是有时候需要使用一些比较高阶的渲染技术时、例如FXAA、deferred shading、Shadow Map等等Mutiple Pass Rendering时，就需要进行Offscreen Render（离线渲染）。此时我们需要先把物体绘制到一个临时的framebuffer上，而不是直接绘制到屏幕上，这就需要我们采用FBO（Framebuffer Object）来达到目标。

Framebuffer Object （FBO）

FBO是由OpenGL创建的Framebuffer。我个人将之理解为OpenGL模拟Default framebuffer而创建的。跟VBO，VAO等OpenGL Object一样，FBO也有Gen，Bind，Delete老三样操作。既然是模拟default framebuffer，那么就应该有为渲染所用的各种缓冲区。这些缓冲区是需要我们自己去生成的（又一大波Gen、Bind、Delete操作袭来）。这些缓冲区即所谓的RenderBuffer。

那什么又是RenderBuffer呢？我也是理解了好久。其实FBO里的Renderbuffer都可以理解成一幅图片（image）。不同格式的图片有不同的用途。例如GL_RGBA的格式可以与FBO的GL_COLOR_ATTACHMENTi绑定用作颜色缓冲区。GL_DEPTH_COMPONENT格式的图片可以与GL_DEPTH_ATTACHMENT绑定作为深度缓冲区。等等。。。。

除了与RenderBuffer绑定，也可以与一幅纹理绑定。纹理也是一幅图片，与Color RenderBuffer的区别是：我们渲染到绑定了纹理的FBO，渲染的结果直接在纹理中，可以直接用下一个pass的纹理操作。