到目前为止，我们已经使用了很多屏幕缓冲了：用于写入颜色值的颜色缓冲、用于写入深度信息的深度缓冲和允许我们根据一些条件丢弃特定片段的模板缓冲。这些缓冲结合起来叫做帧缓冲(Framebuffer)，它被储存在内存中。OpenGL允许我们定义我们自己的帧缓冲，也就是说我们能够定义我们自己的颜色缓冲，甚至是深度缓冲和模板缓冲。

我们目前所做的所有操作都是在默认帧缓冲的渲染缓冲上进行的。默认的帧缓冲是在你创建窗口的时候生成和配置的（GLFW帮我们做了这些）。有了我们自己的帧缓冲，我们就能够有更多方式来渲染了。

和OpenGL中的其它对象一样，我们会使用一个叫做glGenFramebuffers的函数来创建一个帧缓冲对象(Framebuffer Object, FBO)：

unsigned int fbo;

glGenFramebuffers(1, &fbo);

这种创建和使用对象的方式我们已经见过很多次了，所以它的使用函数也和其它的对象类似。首先我们创建一个帧缓冲对象，将它绑定为激活的(Active)帧缓冲，做一些操作，之后解绑帧缓冲。我们使用glBindFramebuffer来绑定帧缓冲。

glBindFramebuffer(GL\_FRAMEBUFFER, fbo);

一个完整的帧缓冲需要满足以下的条件：

* 附加至少一个缓冲（颜色、深度或模板缓冲）。
* 至少有一个颜色附件(Attachment)。
* 所有的附件都必须是完整的（保留了内存）。
* 每个缓冲都应该有相同的样本数。

帧缓冲是否完整

if(glCheckFramebufferStatus(GL\_FRAMEBUFFER) == GL\_FRAMEBUFFER\_COMPLETE) // 执行胜利的舞蹈

现在我们已经创建好一个纹理了，要做的最后一件事就是将它附加到帧缓冲上了：

glFramebufferTexture2D(GL\_FRAMEBUFFER, GL\_COLOR\_ATTACHMENT0, GL\_TEXTURE\_2D, texture, 0);

创建一个渲染缓冲对象的代码和帧缓冲的代码很类似：

unsigned int rbo;

glGenRenderbuffers(1, &rbo);

类似，我们需要绑定这个渲染缓冲对象，让之后所有的渲染缓冲操作影响当前的rbo：

glBindRenderbuffer(GL\_RENDERBUFFER, rbo);