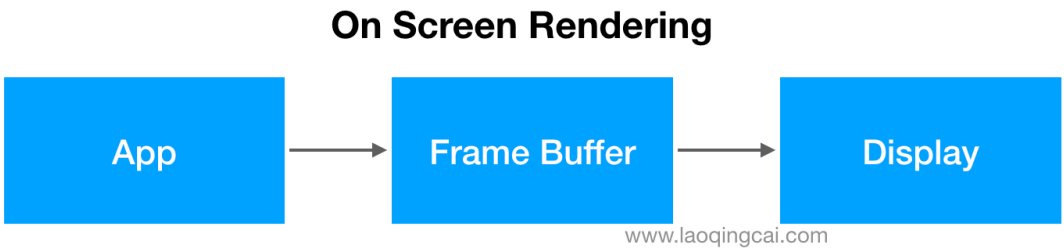
iOS中，GPU的渲染方式有两种：当前屏幕渲染和离屏渲染。

当前屏幕渲染

当前屏幕渲染，英文为 On-Screen Rendering。CPU、GPU不停的将内容渲染完成放入frame buffer帧缓冲区中，显示屏幕从frame buffer中获取内容显示。

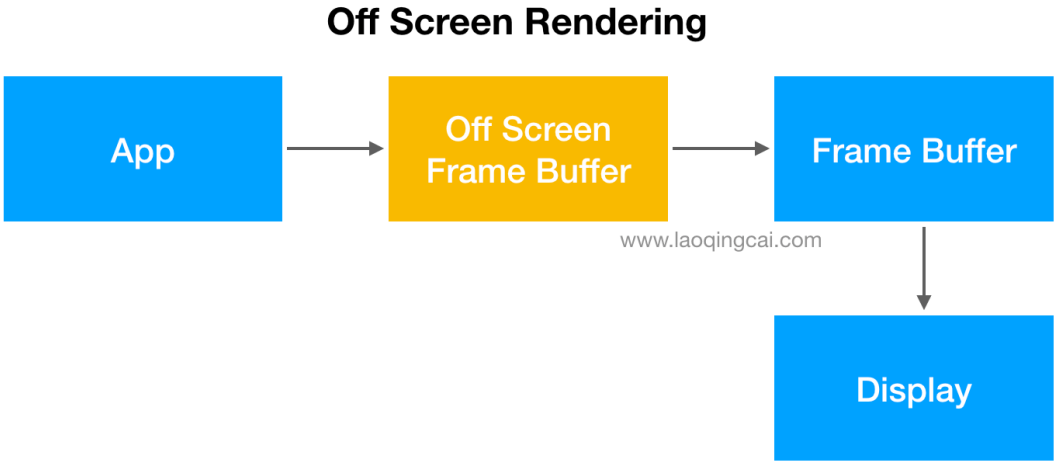
整个流程如下图：



离屏渲染

离屏渲染，英文为 Off-Screen Rendering。需要先创建离屏渲染帧缓冲区 offscreen frame buffer，然后逐一将内容渲染放入其中，完成后对离屏渲染缓冲区做阴影叠加、裁剪等操作，最后将结果拷贝或切换到帧缓冲区 frame buffer中，显示屏幕从 frame buffer 中获取内容并显示。

整个流程如下图：



为什么需要新开缓冲区?

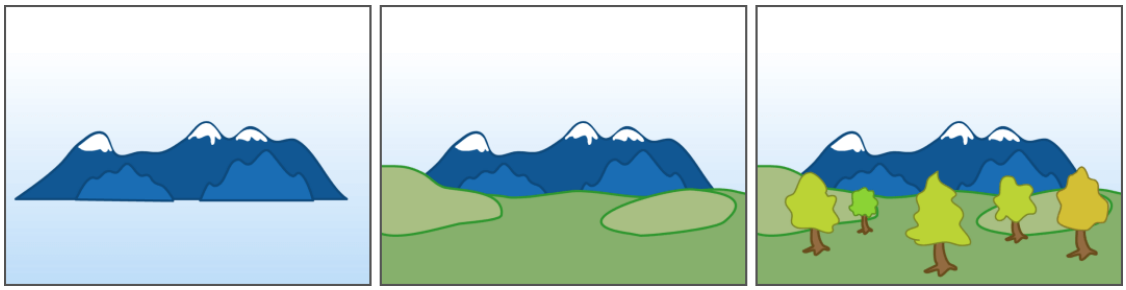
一个很明显的问题是，为什么需要新开缓冲区？使用当前屏幕缓冲区为何不行?

为解决上述问题，先来了解一下画家算法。

画家算法

画家算法，也叫做优先填充算法，它是三维计算机图形学中处理可见性问题的一种解决方法（三维场景投影到二维平面图）。画家算法首先将场景中的多边形根据深度进行排序，然后按照由远及近的顺序进行描绘。这样可以将不可见的部分覆盖，解决可见性的问题。

画家算法示例：



画家算法有个局限：就是无法在后面一层渲染完成后，再回去修改前面的图层，因为前面的图层已经被覆盖了。

画家算法和离屏渲染有何关系呢？简单来说，iOS 当前屏幕缓冲区渲染使用的就是画家算法，普通view的绘制等，是完全没问题的。

然而，对于有**前后依赖**的图层，如阴影、裁剪等，画家算法是无法实现的。此时就需要申请一个临时缓冲区，对该临时缓冲区做特殊处理。

具体为：首先申请临时缓冲区，然后按照画家算法渲染缓冲区，渲染完成后，再对这个缓冲区做最后操作，如阴影叠加、裁剪等。最后，再把临时缓冲区拷贝或者切换到当前的缓冲区上，交给显示器显示。

iOS中设置圆角会触发离屏渲染，其实就是发生了裁剪。

以UIImageView来说，当UIImageView设置image后，实际上视图有两层，一层是view，一层是image。GPU在绘制时，先绘制view层，再绘制image层。设置masksToBounds属性后，需要对image层和view层都裁剪。当前缓冲区是无法对前一层操作的，所以此时会开辟临时缓冲区。

如何检测离屏渲染?

可以通过Xcode检测离屏渲染。

方式是 Xcode->Debug->View Debuging->Rendering->Color Offscreen Rendered Yellow，黄色区域表示离屏渲染。

离屏渲染为何耗性能?

1. 开辟临时缓存空间
2. 缓存区切换，上下文切换，上下文对象比较大，切换操作会有性能消耗
3. 内存拷贝。需要将临时缓存区渲染拷贝到当前缓存区

由于每一帧渲染都需要执行上述操作，因此如果屏幕上触发离屏渲染的操作过多，会导致GPU渲染时间过长造成卡顿。