



## UI 相关面试题

### 一、UIView 与 CALayer

<单一职责原则>

UIView 为 CALayer 提供内容，以及负责处理触摸等事件，参与响应链

CALayer 负责显示内容 contents

### 二、事件传递与视图响应链：

```
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event;  
  
- (BOOL)pointInside:(CGPoint)point withEvent:(UIEvent *)event;
```

如果事件一直传递到 UIApplication 还是没处理，那就会忽略掉

### 三、图像显示原理

- 1.CPU:输出位图
- 2.GPU :图层渲染，纹理合成
- 3.把结果放到帧缓冲区(frame buffer)中
- 4.再由视频控制器根据 vsync 信号在指定时间之前去提取帧缓冲区的屏幕显示内容
- 5.显示到屏幕上

CPU 工作

- 1.Layout: UI 布局，文本计算
- 2.Display: 绘制
- 3.Prepare: 图片解码
- 4.Commit: 提交位图

GPU 渲染管线(OpenGL)

顶点着色，图元装配，光栅化，片段着色，片段处理

#### 四、UI 卡顿掉帧原因

iOS 设备的硬件时钟会发出 Vsync（垂直同步信号），然后 App 的 CPU 会去计算屏幕要显示的内容，之后将计算好的内容提交到 GPU 去渲染。随后，GPU 将渲染结果提交到帧缓冲区，等到下一个 VSync 到来时将缓冲区的帧显示到屏幕上。也就是说，一帧的显示是由 CPU 和 GPU 共同决定的。

一般来说，页面滑动流畅是 60fps，也就是 1s 有 60 帧更新，即每隔 16.7ms 就要产生一帧画面，而如果 CPU 和 GPU 加起来的处理时间超过了 16.7ms，就会造成掉帧甚至卡顿。

## 五、滑动优化方案

CPU:

把以下操作放在子线程中

- 1.对象创建、调整、销毁
- 2.预排版（布局计算、文本计算、缓存高度等等）
- 3.预渲染（文本等异步绘制，图片解码等）

GPU:

纹理渲染，视图混合

一般遇到性能问题时，考虑以下问题:

是否受到 CPU 或者 GPU 的限制?

是否有不必要的 CPU 渲染?

是否有太多的离屏渲染操作?

是否有太多的图层混合操作?

是否有奇怪的图片格式或者尺寸?

是否涉及到昂贵的 view 或者效果?

view 的层次结构是否合理?

## 六、UI 绘制原理

异步绘制:

```
[self.layer.delegate displayLayer: ]
```

代理负责生成对应的 bitmap

设置该 bitmap 作为该 layer.contents 属性的值

## 七、离屏渲染

On-Screen Rendering:当前屏幕渲染，指的是 GPU 的渲染操作是在当前用于显示的屏幕缓冲区中进行

Off-Screen Rendering:离屏渲染，分为 CPU 离屏渲染和 GPU 离屏渲染两种形式。GPU 离屏渲染指的是

GPU 在当前屏幕缓冲区外新开辟一个缓冲区进行渲染操作

应当尽量避免的则是 GPU 离屏渲染

GPU 离屏渲染何时会触发呢?

圆角（当和 maskToBounds 一起使用时）、图层蒙版、阴影，设置

```
layer.shouldRasterize = YES
```

为什么要避免 GPU 离屏渲染?

GPU 需要做额外的渲染操作。通常 GPU 在做渲染的时候是很快的，但是涉及到 offscreen-render 的时候情况就可能有些不同，因为需要额外开辟一个新的缓冲区进行渲染，然后绘制到当前屏幕的过程需要做 onscreen 跟 offscreen 上下文之间的切换，这个过程消耗会比较昂贵，涉及到 OpenGL 的 pipeline 跟 barrier，而且 offscreen-render 在每一帧都会涉及到，因此处理不当肯定会对性能产生一定的影响。另外由于离屏渲染会

增加 GPU 的工作量, 可能会导致 CPU+GPU 的处理时间超出 16.7ms, 导致掉帧卡顿。所以可以的话应尽量减少 offscreen-render 的图层