|  |
| --- |
| 1、理论上一个像素每次只绘制一次是最优的，由于重叠的布局被绘制多次，导致更多的CPU和GPU操作调用，如果耗时操作16.67ms，就会出现掉帧，即卡顿，这个不是好的体验。  所以要减少过度绘制。过度绘制要么是有不必要的绘制，或者是设计时绘制了太多的内容。  过度绘制的原因：太多的View叠加；复杂的层级叠加；更长的inflation时间。xml布局层级越复杂，节点越多，解析耗时越长。  2、工具：Hierachy Viewer，Tracer for OpenGL、Show GPU overdraw。  可以通过开发者选项，打开Show GPU Overdraw选项，观察UI上的Overdraw情况。显示：蓝色、淡绿、淡红，深红代表了4种不同程度的Overdraw的情况。  3、优化：  （1）优化的目的，就是让布局结构简单，减少层级。  （2）选用合适的容器控件，善用RelativeLayout、ConstraintLayout。  （3）使用布局标签include、merge、ViewStub。  （4）merge标签是作为include标签的一个辅助来使用，减少引用布局文件产生的多余布局嵌套。  （5）viewstub是view的子类。他是一个轻量级View，可以用来在程序运行时简单的填充布局文件。使用viewStub替换view.gone()；  （6）去掉window的默认背景。  （7）去掉控件不必要的背景。  （8）慎用Alpha：假如对一个View做Alpha转化，需要先将View绘制出来，然后做Alpha转化，最后将转换后的效果绘制在界面上。  （9）自定义控件中onDraw()的优化，减少绘制叠加。 |

# 1、过度绘制概念，优化方法

## 1.1 Overdraw（过度绘制）

理论上一个像素每次只绘制一次是最优的，由于重叠的布局被绘制多次，导致更多的CPU和GPU操作调用，如果耗时操作16.67ms，就会出现掉帧，即卡顿，这个不是好的体验。

## 描述的是屏幕上的某个像素在同一帧的时间内被绘制了多次。在多层次的UI结构里面，如果不可见的UI也在做绘制的操作，就会导致某些像素区域被绘制了多次，浪费大量的CPU以及GPU资源。

## 可以通过开发者选项，打开Show GPU Overdraw的选项，观察UI上的Overdraw情况。

蓝色、淡绿、淡红，深红代表了4种不同程度的Overdraw的情况，我们的目标就是尽量减少红色Overdraw，看到更多的蓝色区域。

（Layout Inspector的使用）

## 1.2 常用的布局优化方法

1. 优化的目的，就是让布局结构简单，减少层级，减少overdraw。
2. 选用合适的容器控件，善用RelativeLayout、ConstraintLayout。使用相对布局Relativelayout代替线性布局，不是一定的。让布局树的层级更加的扁平。
3. Android自带了工具，使用Hierarchy View的查看层级数。
4. 进行复用布局时，使用抽象布局标签include、merge、ViewStub。
5. include标签用于将布局中公共部分提取出来。
6. merge标签是作为include标签的一个辅助来使用，减少引用布局文件产生的多余布局嵌套。
7. viewstub是view的子类。他是一个轻量级View， 隐藏的，没有尺寸的View。他可以用来在程序运行时简单的填充布局文件。使用viewStub替换view.gone()；

（8）去掉window的默认背景。当我们使用了Android自带的一些主题时，window会被默认添加一个纯色的背景，这个背景是被DecorView持有的。

去掉window的背景可以在onCreate()中setContentView()之后调用

getWindow().setBackgroundDrawable(null);

或者在theme中添加

android:windowbackground="null"；

（9）去掉不必要的背景，如果一些控件都被重叠，背景自然就没有必要的了。

（10）慎用Alpha：假如对一个View做Alpha转化，需要先将View绘制出来，然后做Alpha转化，最后将转换后的效果绘制在界面上。通俗点说，做Alpha转化就需要对当前View绘制两遍，可想而知，绘制效率会大打折扣，耗时会翻倍，所以Alpha还是慎用。

如果一定做Alpha转化的话，可以采用缓存的方式。

view.setLayerType(LAYER\_TYPE\_HARDWARE);

doSmoeThing();

view.setLayerType(LAYER\_TYPE\_NONE);

（11）自定义控件中onDraw()的优化，减少绘制叠加。

# 2、过度绘制

GPU过度绘制概念：GPU过度绘制指的是在屏幕一个像素上绘制多次(超过一次),比如一个TextView后有背景，那么显示文本的像素至少绘了两次，一次是背景，一次是文本。GPU过度绘制或多或少对性能有些影响,设备的内存带宽是有限的，当过度绘制导致应用需要更多的带宽(超过了可用带宽)的时候性能就会降低。带宽的限制每个设备都可能是不一样的。

过度绘制的原因：太多的View叠加；复杂的层级叠加；更长的inflation时间。xml布局层级越复杂，节点越多，解析耗时越长。

通俗地讲，如果一个布局十分复杂，那么就需要来排查是否出现了过度绘制，如果出现了，那么很可能会造成刷新率下降，造成卡顿的现象。那么什么是过度绘制呢？过度绘制就是在同一个区域中叠加了多个控件。这就像小时候我们画画，白纸就是没有绘制的画板，如果我们画了一个房子，涂上了红色，又在上面画了窗户，图上了棕色，窗户上又画了蓝色的玻璃，这重重复的叠加就是过度绘制，在白纸上的结果是，过度绘制的区域纸会被水笔浸的比较湿，在手机上就会出现显示较慢。如果说这是感性的认识，那么我就引用下面一段话来理性的解释一下：

1、布局文件是一个xml文件，inflate布局文件其实就是解析xml，根据标签信息创建相应的布局对象并做关联。xml中的标签和属性设置越多，节点树的深度越深，在解析时要执行的判断逻辑、函数的嵌套和递归就越多，所以时间消耗越多；

2、inflate操作只是布局影响的第一个环节，一个界面要显示出来，在requestLayout后还要执行一系列的measure、layout、draw的操作，每一步的执行时间都会受到布局本身的影响。而界面的最终显示是所有这些操作完成后才实现的，所以如果布局质量差，会增加每一步操作的时间成本，最终显示时间就会比较长。

现在，我们就来说说如何查看是否有过度绘制，和如何避免它吧。

## 2.1 查看是否存在过度绘制

Android提供了三个工具来帮助辨别和解决重绘问题：Hierachy Viewer，Tracer for OpenGL和Show GPU overdraw。前两个可以在ADT工具或者独立的monitor工具中找到，最后一个是在开发者选项的一部分.

0.优化工具介绍

Lint工具:Eclipse中,点击即可,下面的窗口中会出现提示,根据提示和具体解决办法消除。Android Studio自带Lint工具,不合理或者需要优化和注意的地方,会用黄色标记出来。Lint工具不仅对布局有很好的优化建议,对代码中不合理的活着存在潜在风险的模块也会提出优化建议,所以一个好的建议是:多使用Lint工具检查自己的应用,尽量消除所有的建议。Lint工具可以用命令行来运行,具体使用可以参考:[tools.android.com](http://androidperformance.com/2014/10/20/tools.android.com" \t "https://blog.csdn.net/guofengpu/article/details/_blank)。

Hierarchy Viewer:此工具是一个ADT工具（或者monitor,最新版本的SDK建议不使用独立的HV工具,而是直接在monitor中进行操作.）的一部分，可以被用作对视图层级进行快速解读。在处理布局问题时特别有用，对于性能问题也很适用。Hierarchy Viewer默认只能在非加密设备使用，例如工程机，工程平板或者模拟器。为了能够在任何手机上使用Hierarchy Viewer，你得在你的应用中添加ViewServer，这是一个开源库,使用方法可以参考这里。连接上设备，打开Hierarchy Viewer(定位到tools/目录下，直接执行hierarchyviewer的命令，选定需要查看的Process，再点击Load View Hierarchy会显示出当前界面的布局Tree。在每个模块的Traffic light上有三个灯，分别代表了Measure, Layout and Draw三个步骤的性能。

1. GPU过渡绘制：对于过度绘制的[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "https://blog.csdn.net/guofengpu/article/details/_blank)主要通过人工进行测试，也是发现应用过渡绘制的首选途径。通过打开开发者选项中的显示GPU过度绘制(魅族手机:设置—辅助功能–开发人员工具–硬件加速渲染—调试GPU过渡绘制— 显示过渡绘制区域.）来进行测试（PS：只有android4.2及以上的版本才具备此功能)。

2. 颜色标识: 从好到差:蓝-绿-淡红-红

1. 蓝色1x过度绘制  
 2. 绿色2x过度绘制  
 3. 淡红色3x过度绘制  
 4. 红色超过4x过度绘制

3. 验收标准:

1. 控制过度绘制为2x  
 2. 不允许存在4x过度绘制  
 3. 不允许存在面积超过屏幕1/4区域的3x过度绘制（淡红色区域）

最理想的是蓝色,一个像素只绘制一次，合格的页面绘制是白色、蓝色为主，绿色以上区域不能超过整个的三分之一，颜色越浅越好。

## 2.2总结原因：

****1.太多重叠的背景****

重叠着的背景有时候是有必要的，有时候是没必要的。这要视你的项目具体情况而定。

****2.太多叠加的View****

或者本来这个UI布局就很复杂或者你是为了追求一个炫丽的视觉效果，这都有可能使得很多view叠加在一起。这个情况非常普遍，下面的建议中会谈谈怎么减少这种情况带来的影响。

****3.复杂的Layout层级****

复杂的层级关系，这个在布局中也很常见，下面也会说这种情况怎么做可以尽可能的减少过度绘制。

****建议：****

****1.太多重叠的背景****

这个问题其实最容易解决，建议前期在设计时尽量保持整体背景统一，另外开发可以检查你在布局和代码中设置的背景，有些背景是被隐藏在底下的，它永远不可能显示出来，这种没必要的背景一定要移除，因为它很可能会严重影响到app的性能。

****2.太多重叠的view****

第一个建议是：使用ViewStub来加载一些不常用的布局，它是一个轻量级且默认不可见的视图，可以动态的加载一个布局，只有你用到这个重叠着的view的时候才加载，推迟加载的时间。第二个建议是：如果使用了类似viewpager+Fragment这样的组合或者有多个Fragment在一个界面上，需要控制Fragment的显示和隐藏，尽量使用动态地Inflation view，它的性能要比SetVisiblity好。

****3.复杂的Layout层级****

这里的建议比较多一些，首先推荐用Android提供的布局工具Hierarchy Viewer来检查和优化布局。第一个建议是：如果嵌套的线性布局加深了布局层次，可以使用相对布局来取代。第二个建议是：用标签来合并布局，这可以减少布局层次。第三个建议是：用标签来重用布局，抽取通用的布局可以让布局的逻辑更清晰明了。记住，这些建议的最终目的都是使得你的Layout在Hierarchy Viewer里变得宽而浅，而不是窄而深。

# 3、一些经验

（1）Android目前的主流设备分辨率为480×800、720×1280、1080×1920(荣耀6)，单位是像素。在做界面设计的时候请重点关注这几个分辨率。其他还有240X320、320X480,2280X1080(honor10)。

（2）在制作高清原型图的时候，推荐使用720\*1280像素作为基础分辨率，这个尺寸720×1280中显示完美，在1080×1920中看起来也比较清晰；切图后的图片文件大小也适中，应用的内存消耗也不会过高。

（3）在720×1280分辨率下，对用户来说，正常文本的大小是27px较舒适，注释文本21px较舒。

（4）在设计移动客户端的时候，记得时刻遵守48dp原则。我们通常把48dp作为可触摸控件的标准。

（5）这是因为，一般来说,48dp 转化为一个物理尺寸约9毫米。建议的目标大小为 7-10毫米的范围,这是一个用户手指能准确并且舒适触摸的区域。

（6）在每个UI元素之间，最好保留8dp的空白区域。

# 4、ViewStub

## 4.1 什么是ViewStub？

ViewStub是一个看不见，没有大小，不占位置的view.当ViewStub可见或者调用inflate()方法的时候，布局就会被加载替代。

为什么使用ViewStub？

ViewStub可以避免同一时间加载大量组件而产生卡顿影响用户体验。可以将一些暂时用不到的组件用ViewStub延迟他们的绘制加载。使用GONE属性是可以隐藏组件的，但GONE还是会初始化组件。

## 4.2 merge

什么是merge？

android xml文件布局的一个组件，配合include使用。默认布局跟Fragment一样。当他加载时不会生成ViewGroup容器。

merge根节点内的控件布局取决于<include>这个布局的父布局是哪个布局。

为什么使用merge，主要是为了减少组件嵌套。

基本上merge是配合include使用。

merge的使用限制：

merge 只能作为XML布局的根标签使用。

当Inflate以merge开头的布局文件时，必须指定一个父ViewGroup，并且必须设定attachToRoot为true（参看inflate(int, android.view.ViewGroup, Boolean)方法）。

## 4.3 include

include标签使一个layout布局文件加载另外一个layout布局。方便布局重用。

include可以添加很多属性，但是建议还是只使用一个layout属性来指明所要加载的layout文件。其他属性不建议使用。