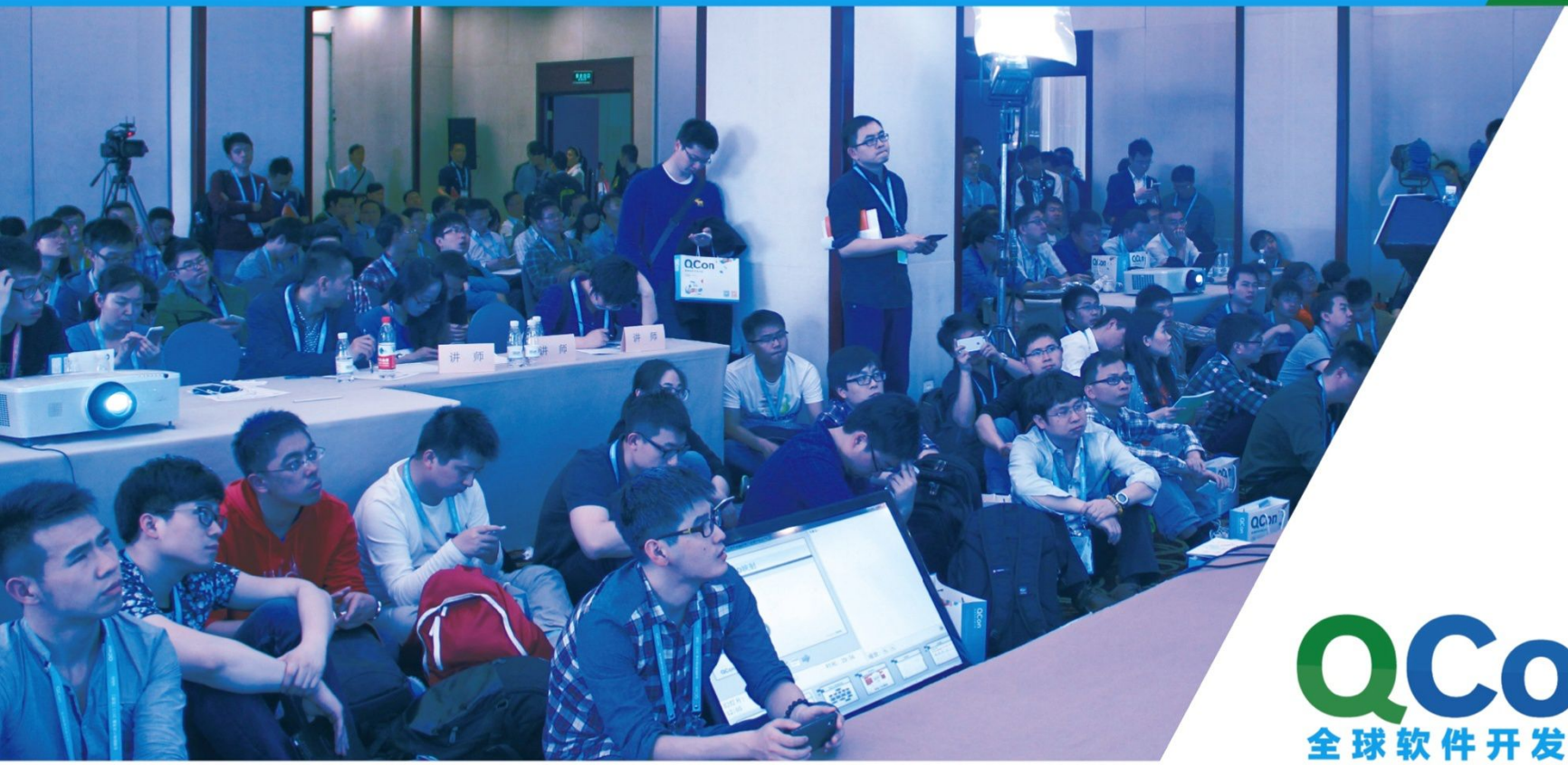
QCon全球软件开发大会

International Software Development Conference



QCon
全球软件开发大会

Android应用程序UI硬件加速渲染技术

罗升阳

2015.10.16

About Speaker

- CSDN博客《老罗的Android之旅》作者
- 书籍《Android系统源代码情景分析》作者
- 博客：http://blog.csdn.net/Luoshengyang
- 微博：http://weibo.com/shengyangluo

Android UI真的不如iOS UI流畅吗

Android应用程序运行在虚拟机上?

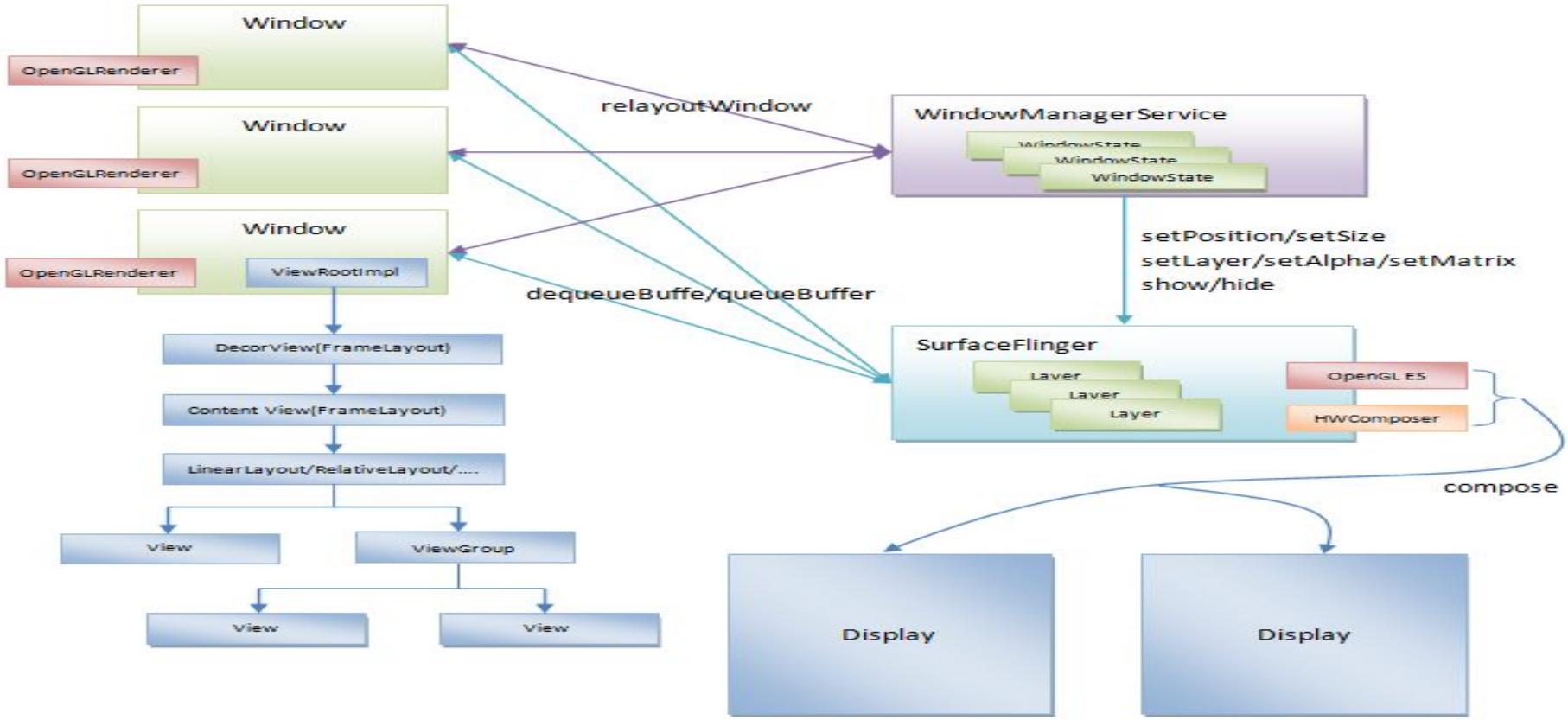
Android应用程序UI没有硬件加速?

Android应用程序没有独立的渲染线程?

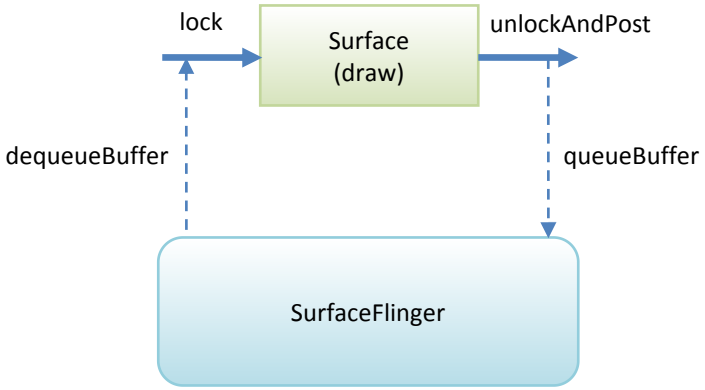
Agenda

- Android系统UI整体架构概述
- Android应用程序UI渲染方式
- Android应用程序UI视图结构
- Android应用程序UI渲染模型
- Android应用程序UI异步动画
- Android应用程序UI渲染优化

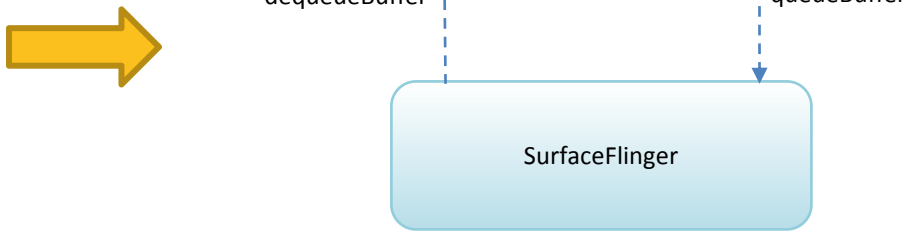
Android系统UI整体架构概述



Android应用程序UI渲染方式



Software

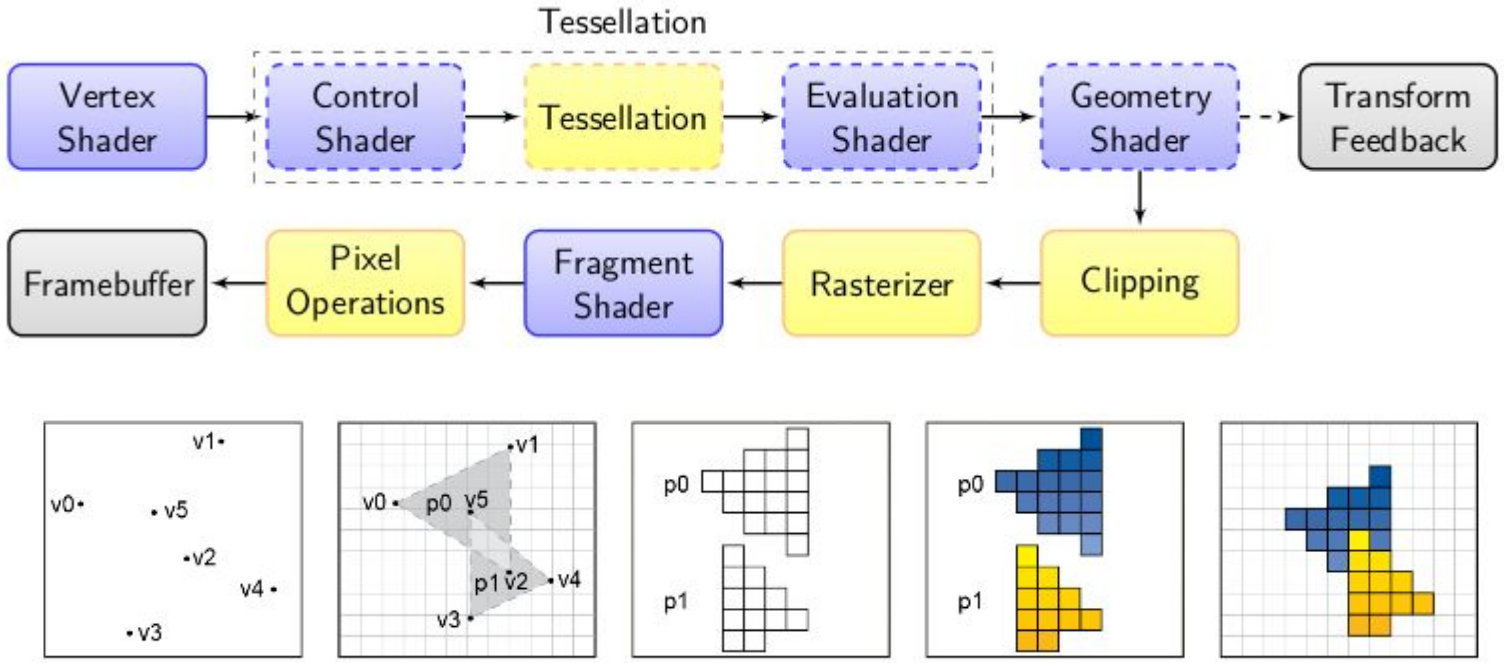


Hardware

Android应用程序UI渲染方式

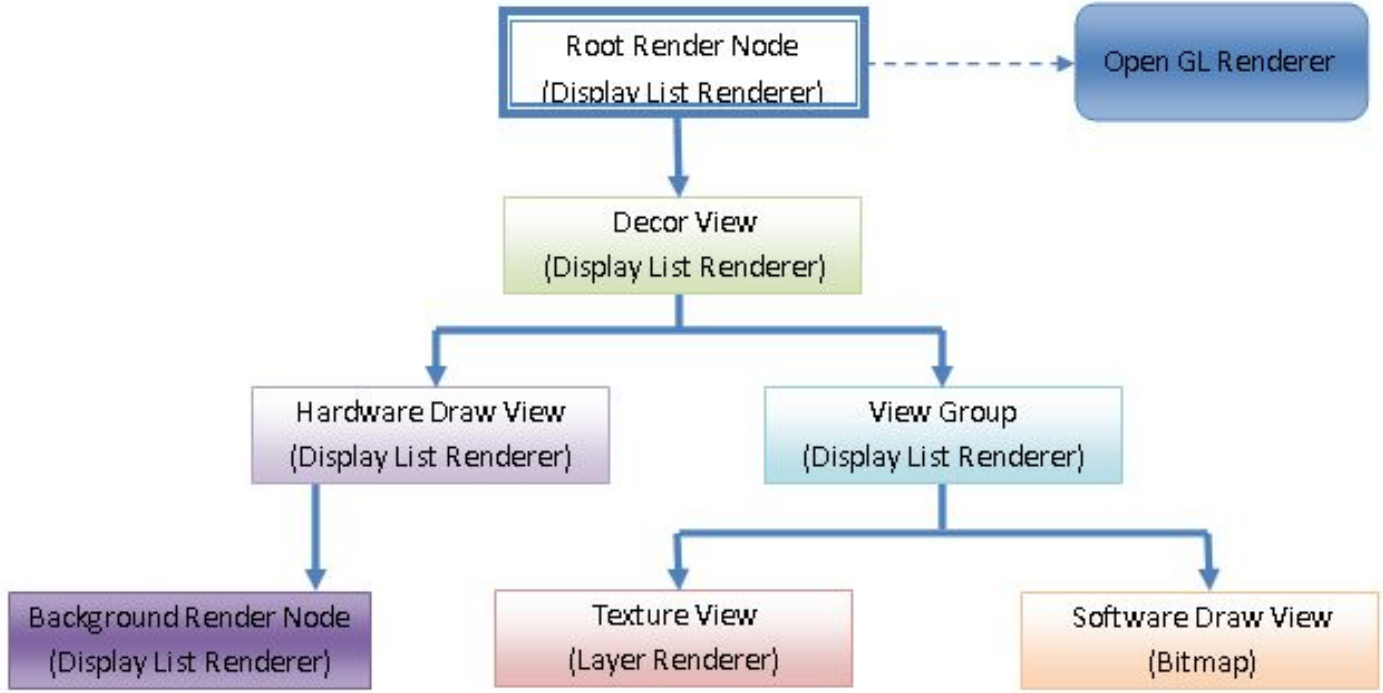
- 硬件渲染进化史：Run fast, smooth, and responsively
  - Honeycomb
    - OpenGLRenderer
  - Ice Cream Sandwich
    - TextureView - OpenGL ES texture views
    - Required to support hardware-accelerated 2D drawing
  - Jelly Bean
    - Extend vsync to all drawing and animations to ensure a constant framerate
    - Tripple buffering
    - Synchronizing touch to vsync timing
    - Predict what your move for a more reactive UI
    - Each time you touch the screen, boost the CPU to reduce latency - CPU input boost
  - KitKat
    - Optimize memory usage to support entry-level devices that have as as little as 512MB RAM, including kernel samepage merging(KSM), swap to zRAM, etc.
    - ART
  - Lollipop
    - Replace Dalvik with ART
    - Render thread
    - Compacting GC
    - Background GC and Foreground GC

Android应用程序UI渲染方式



Simplified Open GL Rendering Pipeline Overview

Android应用程序UI视图结构



Android应用程序UI视图结构

- Render Node
  - 每一个View都是一个Render Node
  - 如果View设置有Background，那么Background也是一个Render Node
- Display List
  - 保存绘制命令的一个缓冲区
  - 第一次渲染时，每一个View都需要构造自己的Display List -- onDraw
  - 每次绘制时，只有需要更新的View才需要重建Display List，否则复用上一次构建的
  - 简单的View属性发生变化，也不需要重建Display List
  - 每一个View都有两个Display List，一个在Main Thread使用，一个在Render Thread 使用，为并行UI渲染提供基础
  - 为全局UI渲染优化提供可能
- Display List Renderer
  - 将Canvas API调用转化为命令写入到Display List中去
- Layer Renderer
  - 直接使用Texture或者FBO进行绘制
- Open GL Renderer
  - 从Root View开始，将所有View的Display List转化为Open GL命令进行渲染

Android应用程序UI视图结构

- 不是所有的Canvas API都是GPU能够支持的

	First supported API level
Canvas	
drawBitmapMesh() (colors array)	18
drawPicture()	✗
drawPosText()	16
drawTextOnPath()	16
drawVertices()	✗
setDrawFilter()	16
clipPath()	18
clipRegion()	18
clipRect(Region.Op.XOR)	18
clipRect(Region.Op.Difference)	18
clipRect(Region.Op.ReverseDifference)	18
clipRect() with rotation/perspective	18
drawArc()	21
drawRoundRect()	21
saveLayer() with RectF dimensions	21
saveLayer() with float dimensions	21
saveLayerAlpha() with RectF dimensions	21
saveLayerAlpha() with float dimensions	21

http://developer.android.com/guide/topics/graphics/hardware-accel.html

Android应用程序UI视图结构

- Software Draw View
  - View.setLayerType(View.LAYER_TYPE_SOFTWARE, null)
  - Draw on a canvas based a bitmap
  - Draw the bitmap in the display list
  - Draw the display list using GPU

Android应用程序UI视图结构

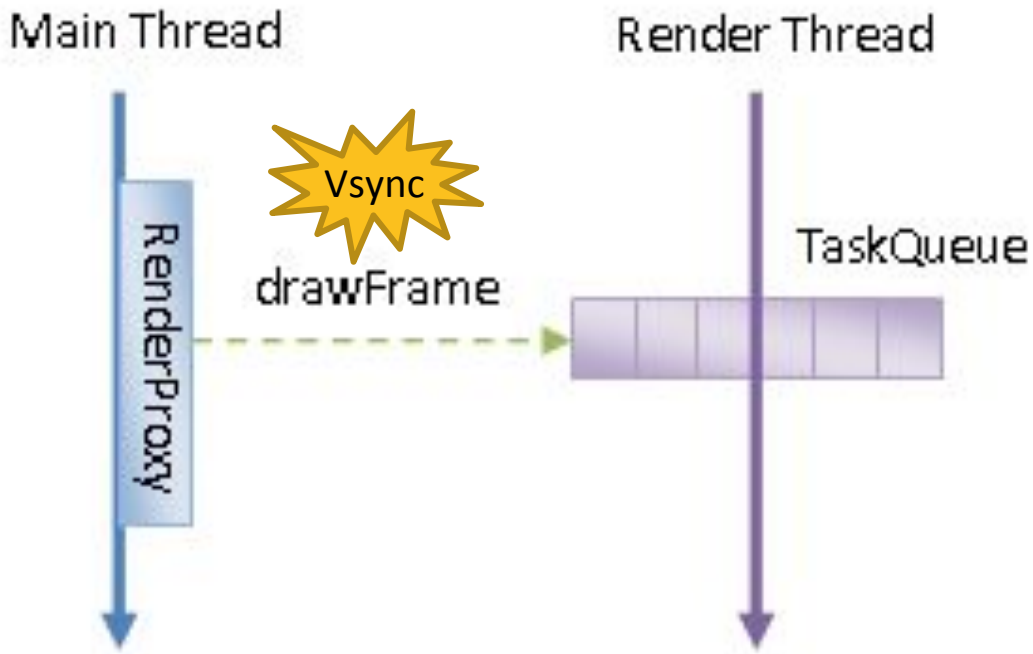
- Texture View
  - 通常先将一个View的绘制命令保存在一个Display List中，然后再使用Open GL API绘制该Display List
  - 直接将View的内容当作一个Open GL Texture处理，从而跳过Display List这一中间环节—Camera Preview、Video Play

Android应用程序UI渲染模型

- Main Thread
  - 处理用户输入
  - 处理系统消息
  - 处理自定义消息
  - 构建Display List – onDraw
- Render Thread
  - 将Display List渲染成UI
  - 执行动画

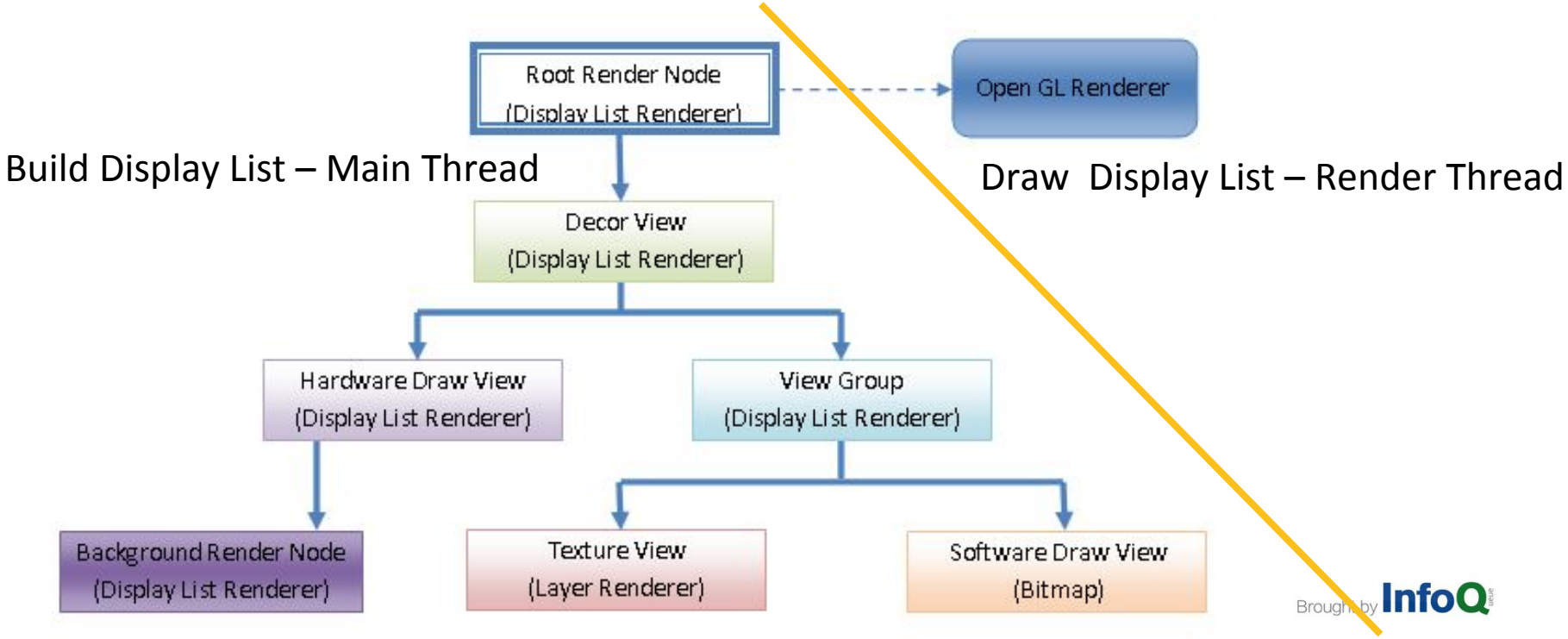
Android应用程序UI渲染模型

- Main Thread与Render Thread之间的通信



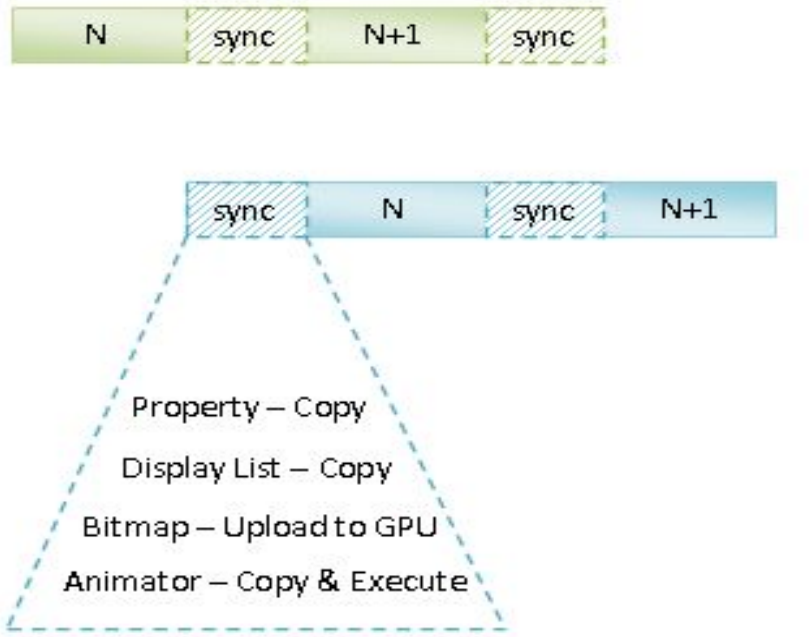
Android应用程序UI渲染模型

- Main Thread与Render Thread协作渲染



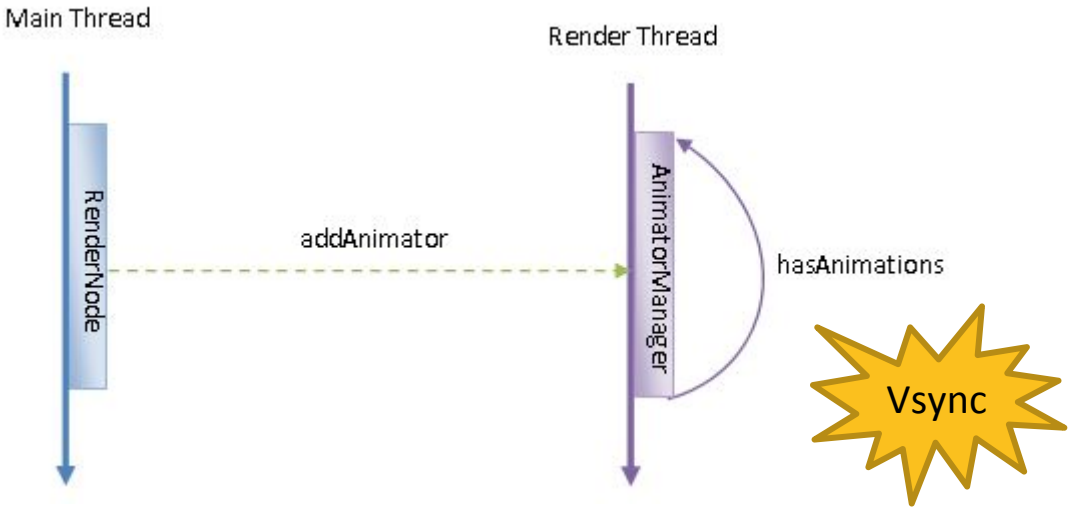
Android应用程序UI渲染模型

- Main Thread与Render Thread同步过程



Android应用程序UI异步动画

- ViewPropertyAnimator
  - RenderNodeAnimator.setAllowRunningAsync(true)
- All RenderNodeAnimator is allowed to run async

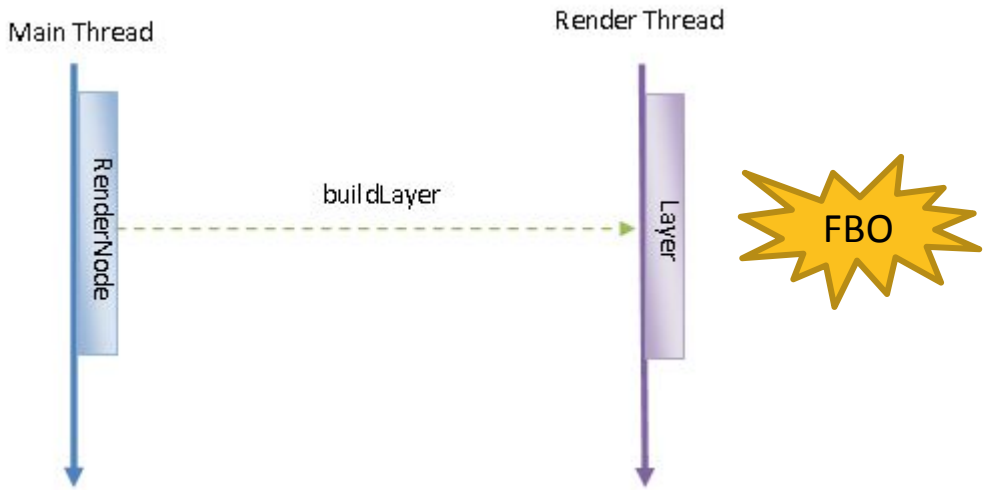


Android应用程序UI渲染优化

- Display List
  - Display List记录了View的某些属性值，例如位置、透明度等
  - 当View的上述属性发生变化时，不必重建整个Display List，单独记录，在渲染时直接应用在原来的Display List上，避免重新执行onDraw操作

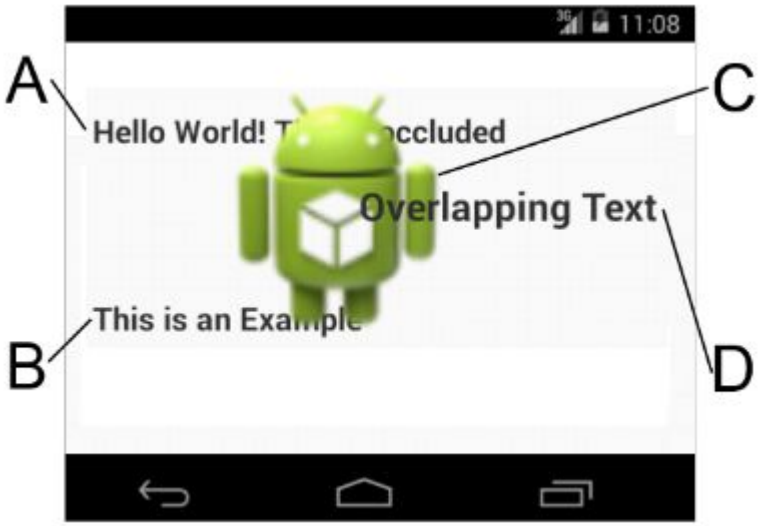
Android应用程序UI渲染优化

- ViewPropertyAnimator
  - ViewPropertyAnimator.withLayer
  - 在动画期间，临时将目标View的Layer Type设置为LAYER_TYPE_HARDWARE
  - 目标View将不通过Display List进行渲染，而是直接渲染在一个FBO上



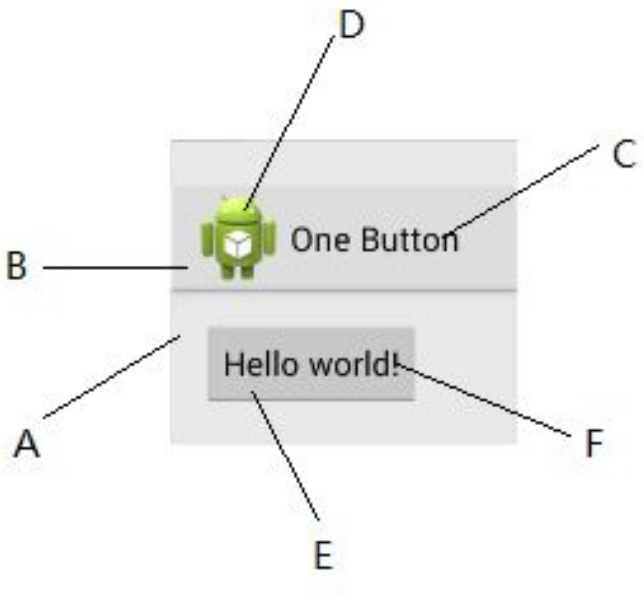
Android应用程序UI渲染优化

- Merging of Operations



Android应用程序UI渲染优化

- Merging of Operations



Android应用程序UI渲染优化

- Merging of Operations

a. The layout draws the background image, which is a linear gradient

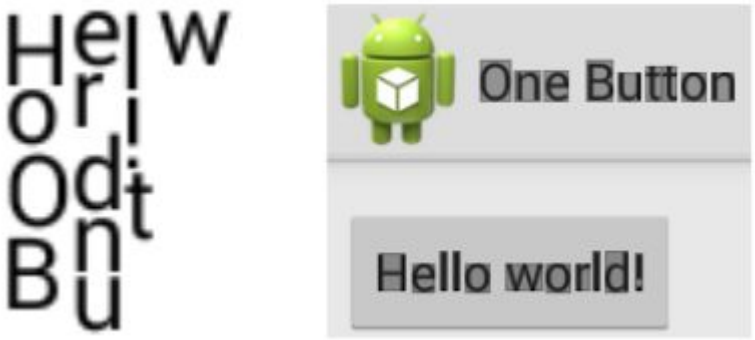
b. Both the ActionBar and Button background NinePatch are drawn, which on both on Asset Atlas texture. These two operations were merged into one batch

c. Text for the Button and the ActionBar is drawn, using the same font texture. Again, these two operations were merged into one batch, and the FontRenderer also used one font texture for both text elements

d. The application's icon is drawn

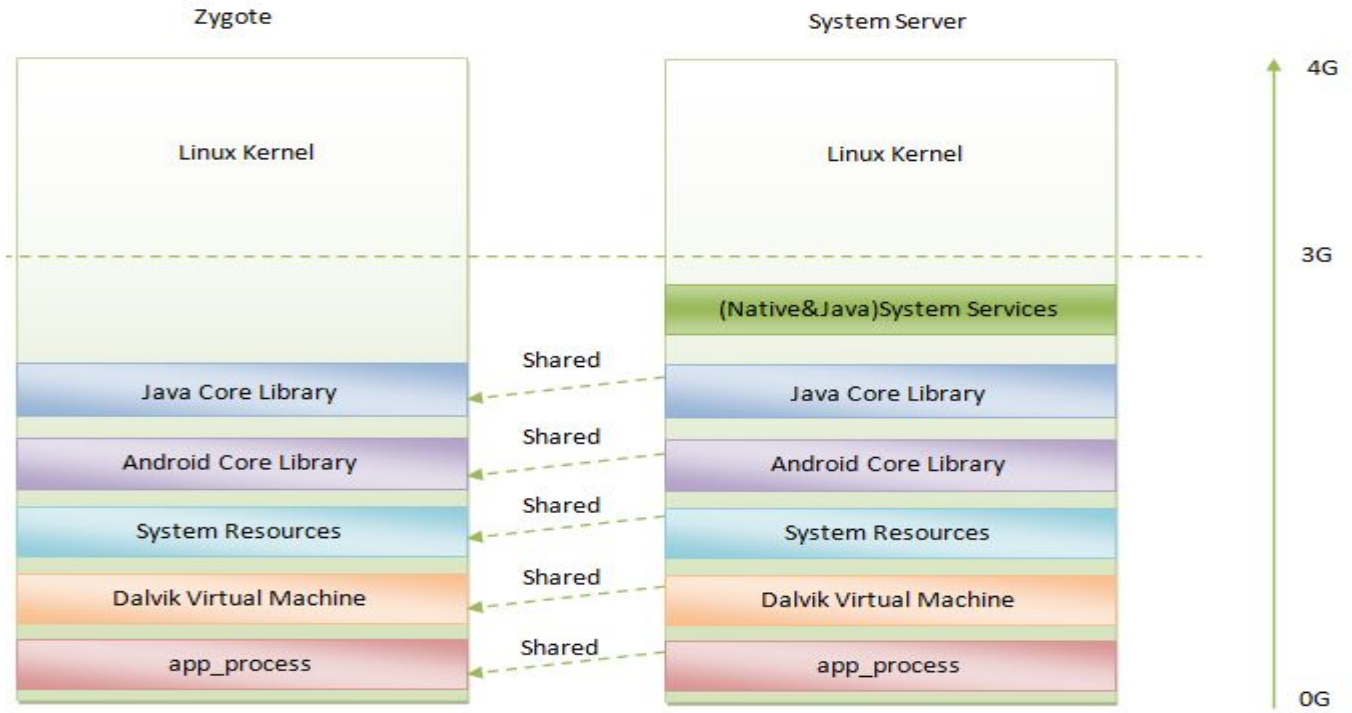
Android应用程序UI渲染优化

- Font Texture



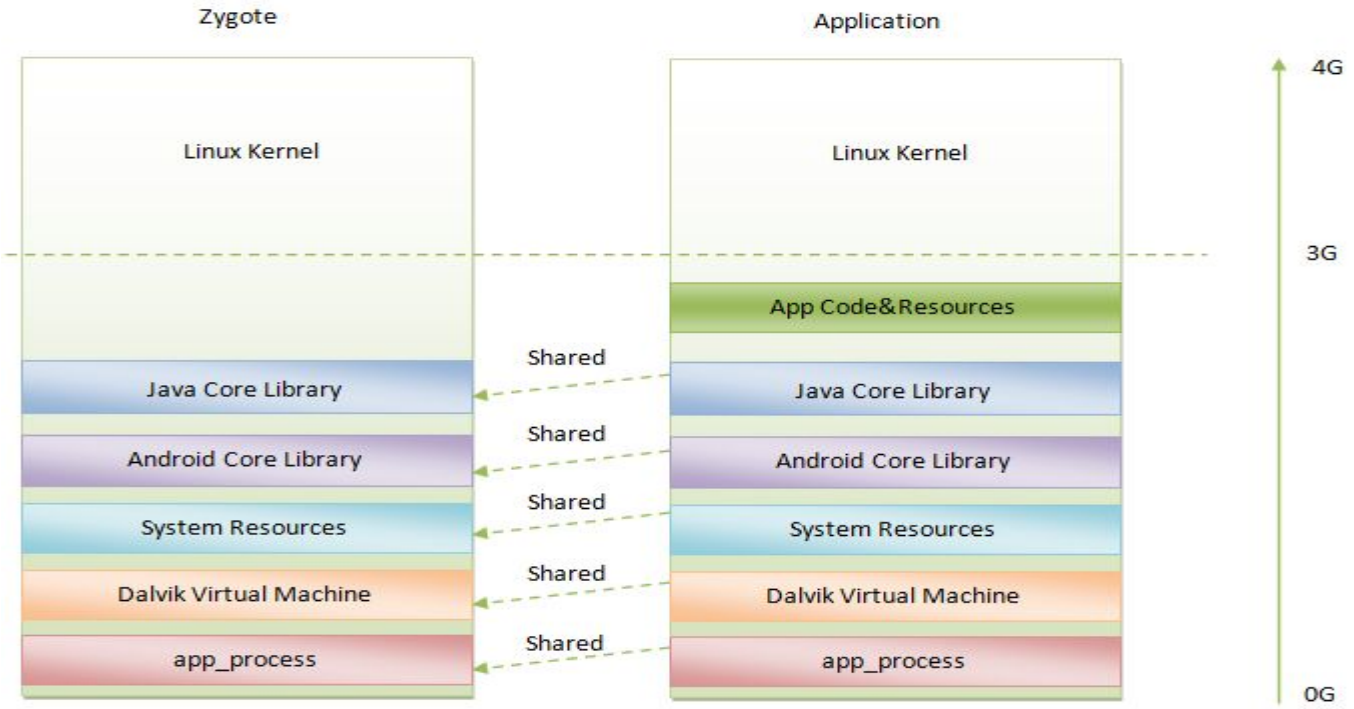
Android应用程序UI渲染优化

- Asset Atlas Texture – Resources sharing



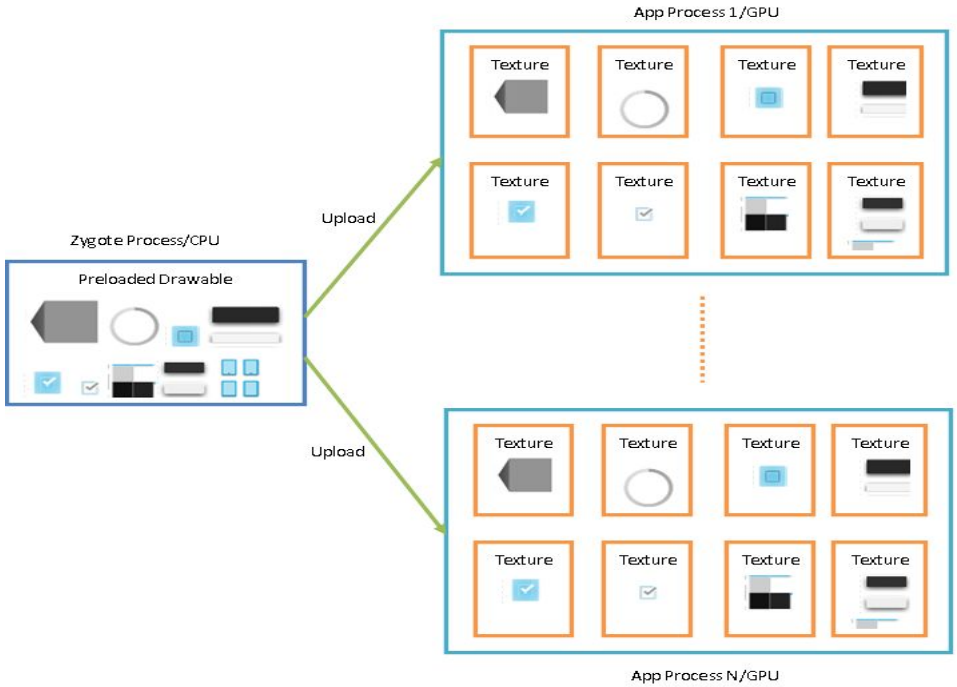
Android应用程序UI渲染优化

- Asset Atlas Texture – Resources sharing



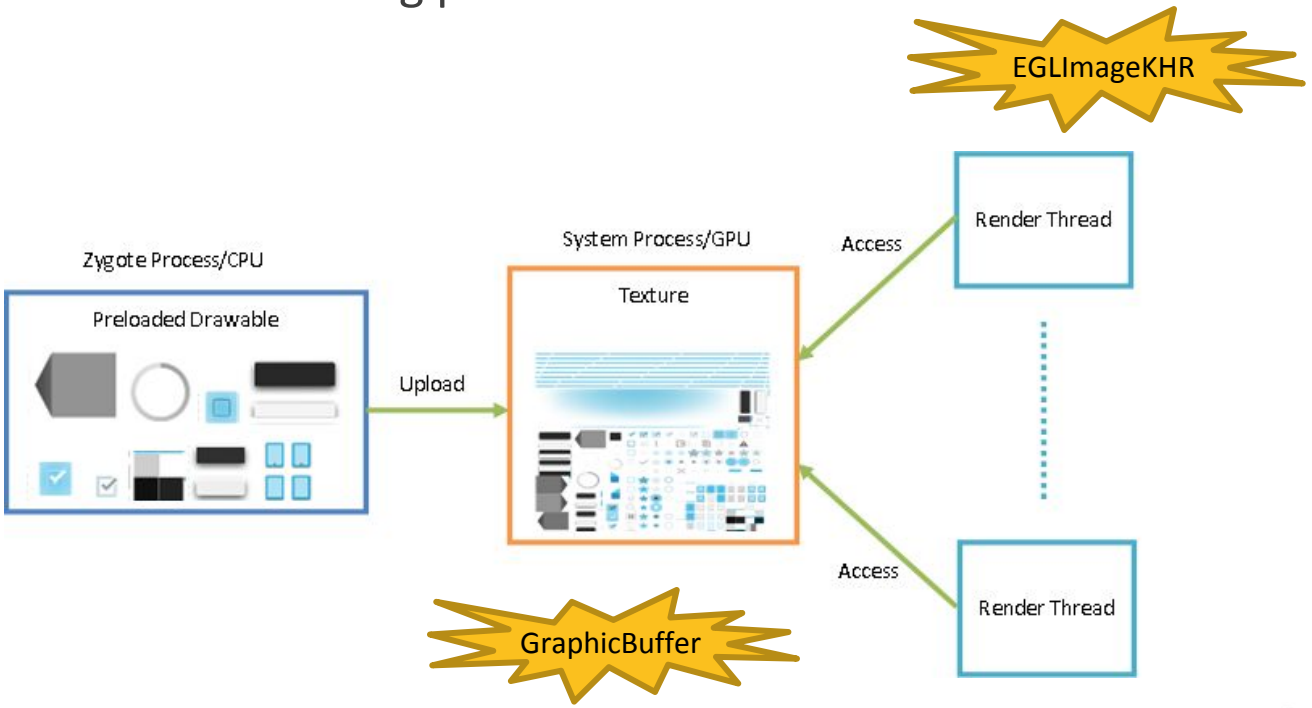
Android应用程序UI渲染优化

- Asset Atlas Texture – using preloaded drawable as texture in the past



Android应用程序UI渲染优化

- Asset Atlas Texture – using preloaded drawable as texture now



THANKS