Application Framework
View and ViewRootImpl

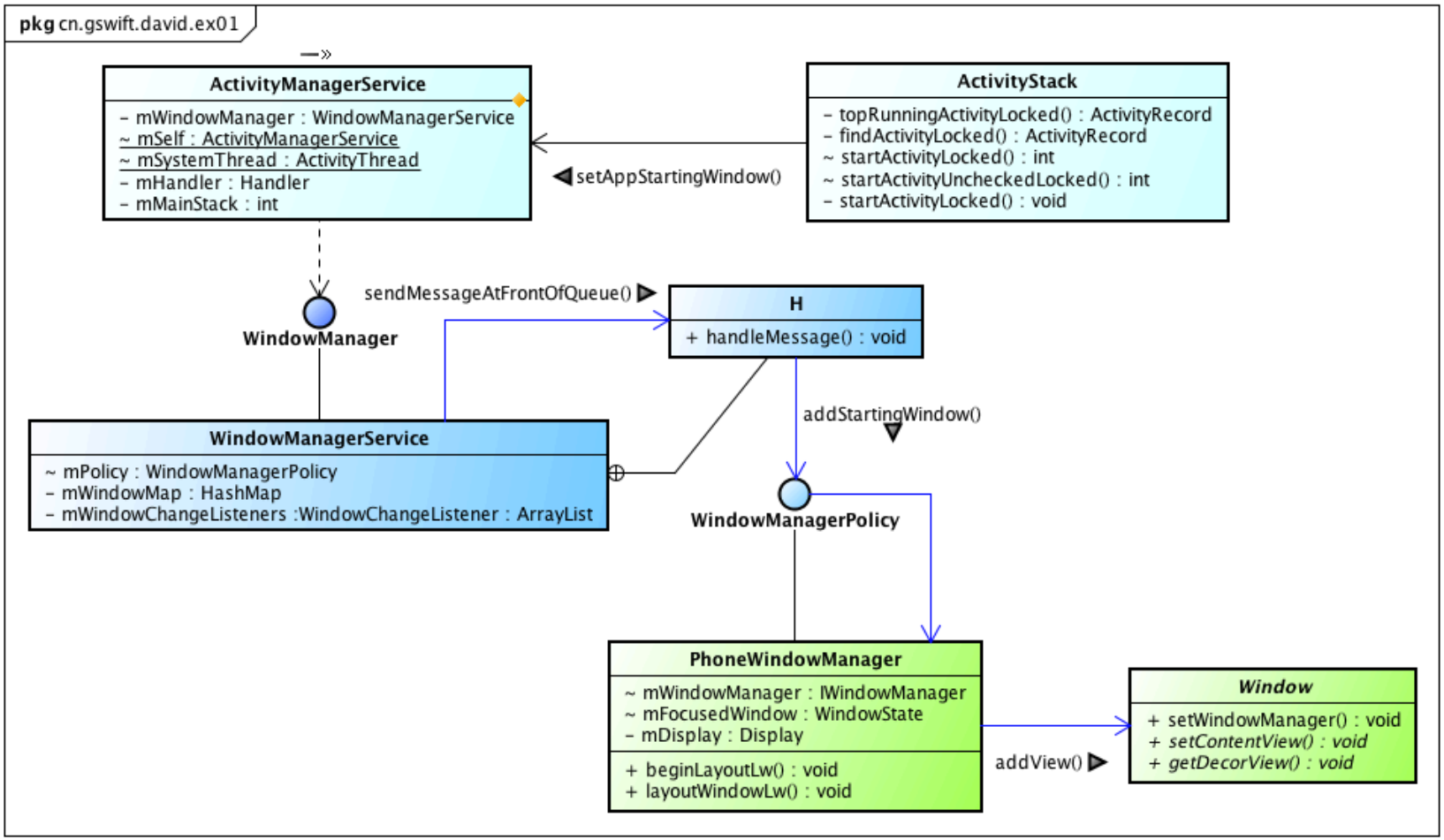
David Lau • China

Email: zhiyong.liu@aliyun.com Latest Update：2013-09-08

Source From: http://via.9gag.com

Window-View

AddView

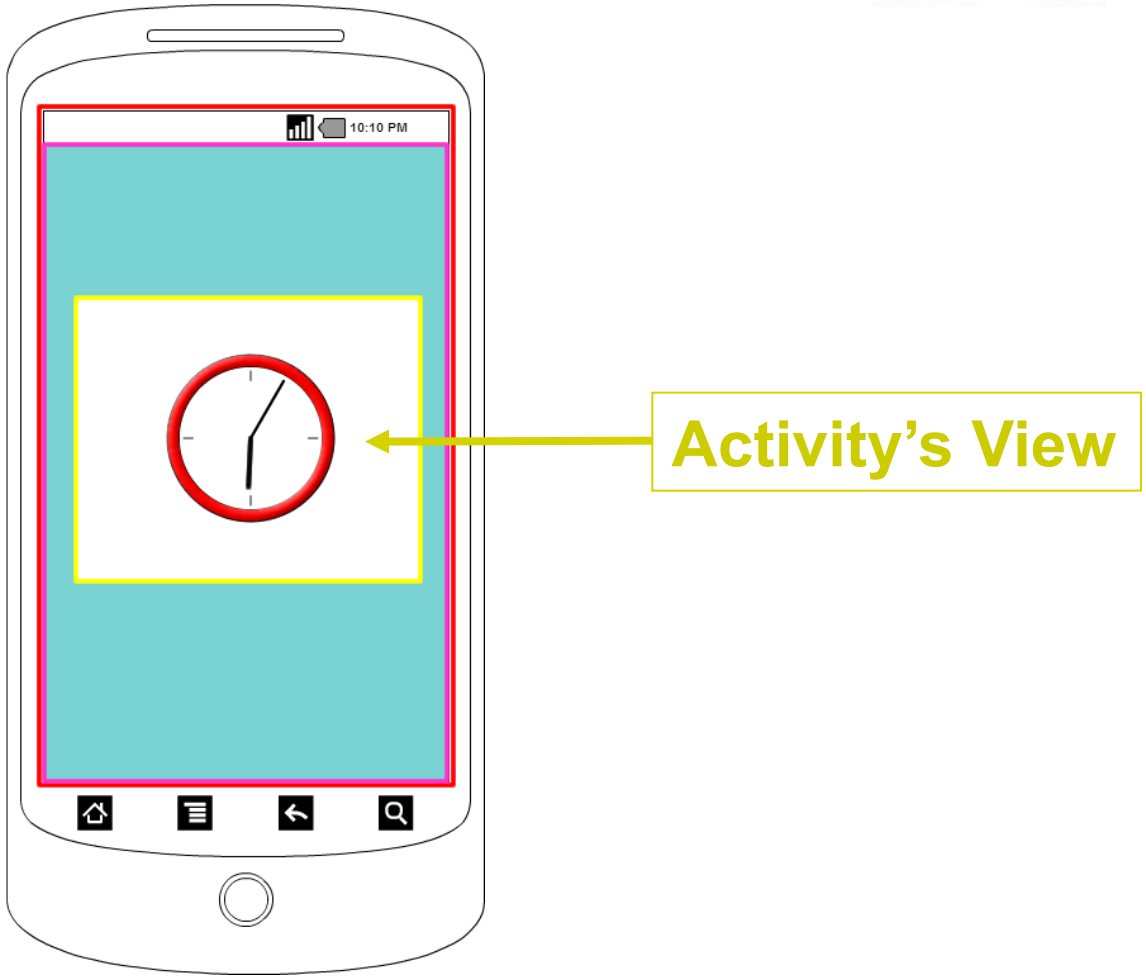

pkg cn.gswift.david.ex01
ActivityManagerService
- mWindowManager : WindowManagerService
~ mSelf : ActivityManagerService
~ mSystemThread : ActivityThread
- mHandler : Handler
- mMainStack : int
ActivityStack
- topRunningActivityLocked() : ActivityRecord
- findActivityLocked() : ActivityRecord
~ startActivityLocked() : int
~ startActivityUncheckedLocked() : int
- startActivityLocked() : void
setAppStartingWindow()
sendMessageAtFrontOfQueue()
WindowManager
H
+ handleMessage() : void
addStartingWindow()
WindowManagerService
~ mPolicy : WindowManagerPolicy
- mWindowMap : HashMap
- mWindowChangeListeners :WindowChangeListener : ArrayList
WindowManagerPolicy
PhoneWindowManager
~ mWindowManager : IWindowManager
~ mFocusedWindow : WindowState
- mDisplay : Display
+ beginLayoutLw() : void
+ layoutWindowLw() : void
addView()
Window
+ setWindowManager() : void
+ setContentView() : void
+ getDecorView() : void

View-ViewRootImpl

View

This class represents the basic building block for user interface components. A View occupies **a rectangular area** on the screen and **is responsible for drawing and event handling.**

View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.). The ViewGroup subclass is the base class for layouts, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.


10:10 PM
Activity's View

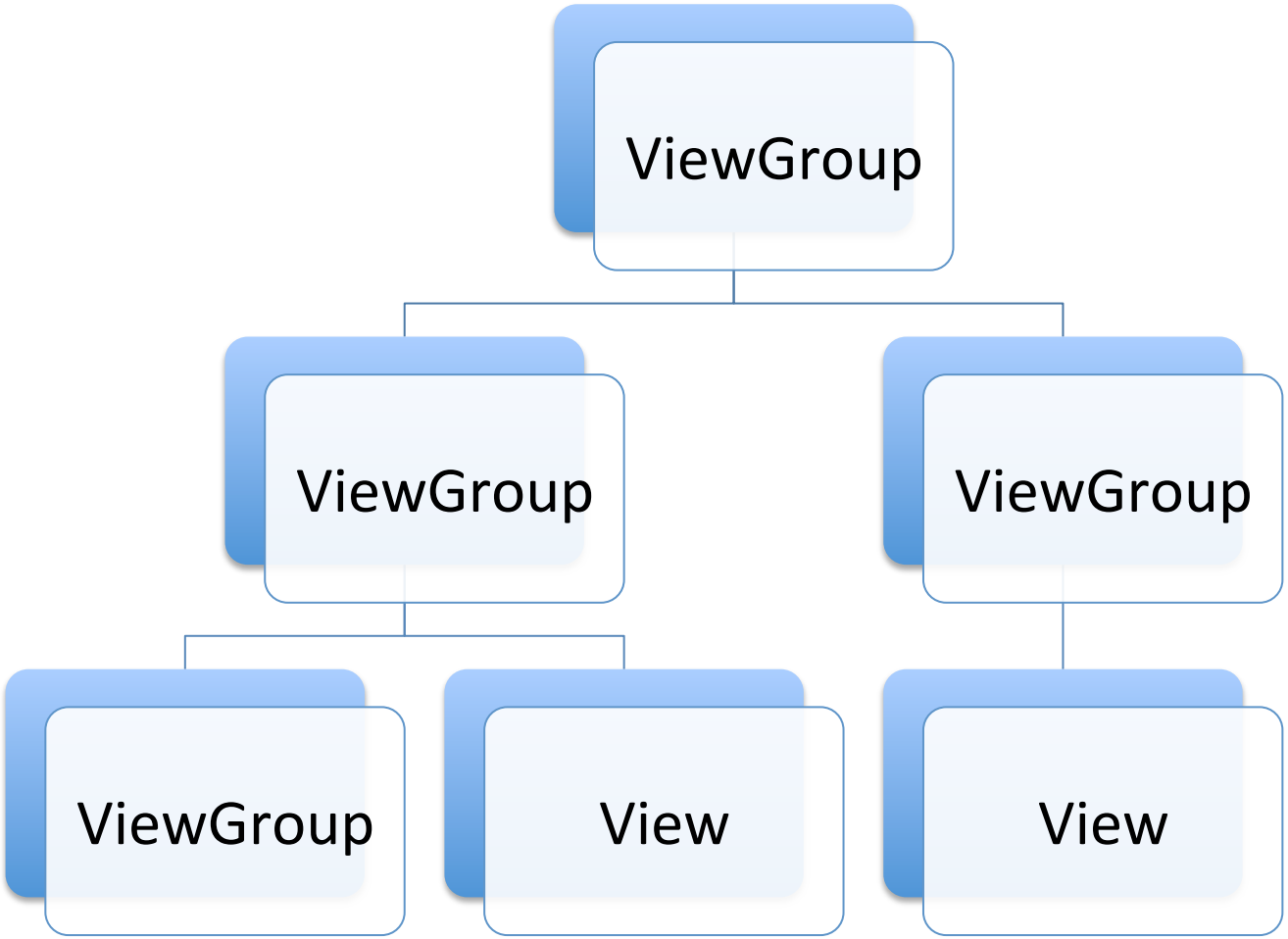
View’s Listener

- View.OnClickListener
- View.OnLongClickListener
- View.OnKeyListener
- View.OnTouchListener
- View.OnDragListener
- View. OnCreateContextMenuListener
- View.OnGenericMotionListener
- View.OnFocusChangeListener
- View.OnHoverListener
- View.OnSystemUiVisibilityChangeListener

ViewRootImpl

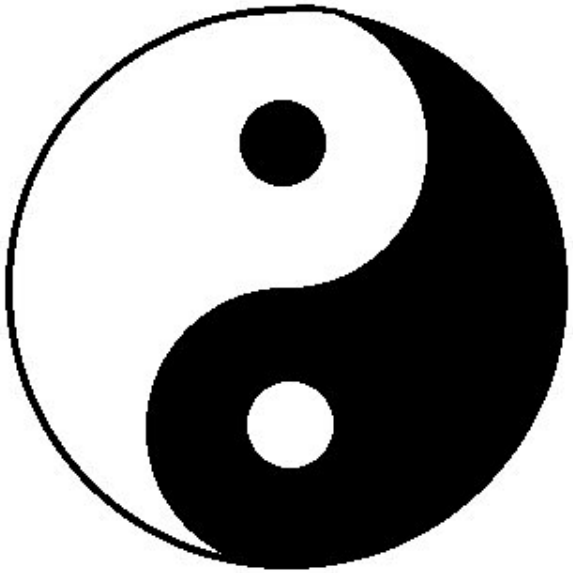
- The top of a view hierarchy, implementing the needed protocol between View and the WindowManager. This is for the most part an internal implementation detail of WindowManagerGlobal.

The view hierarchy

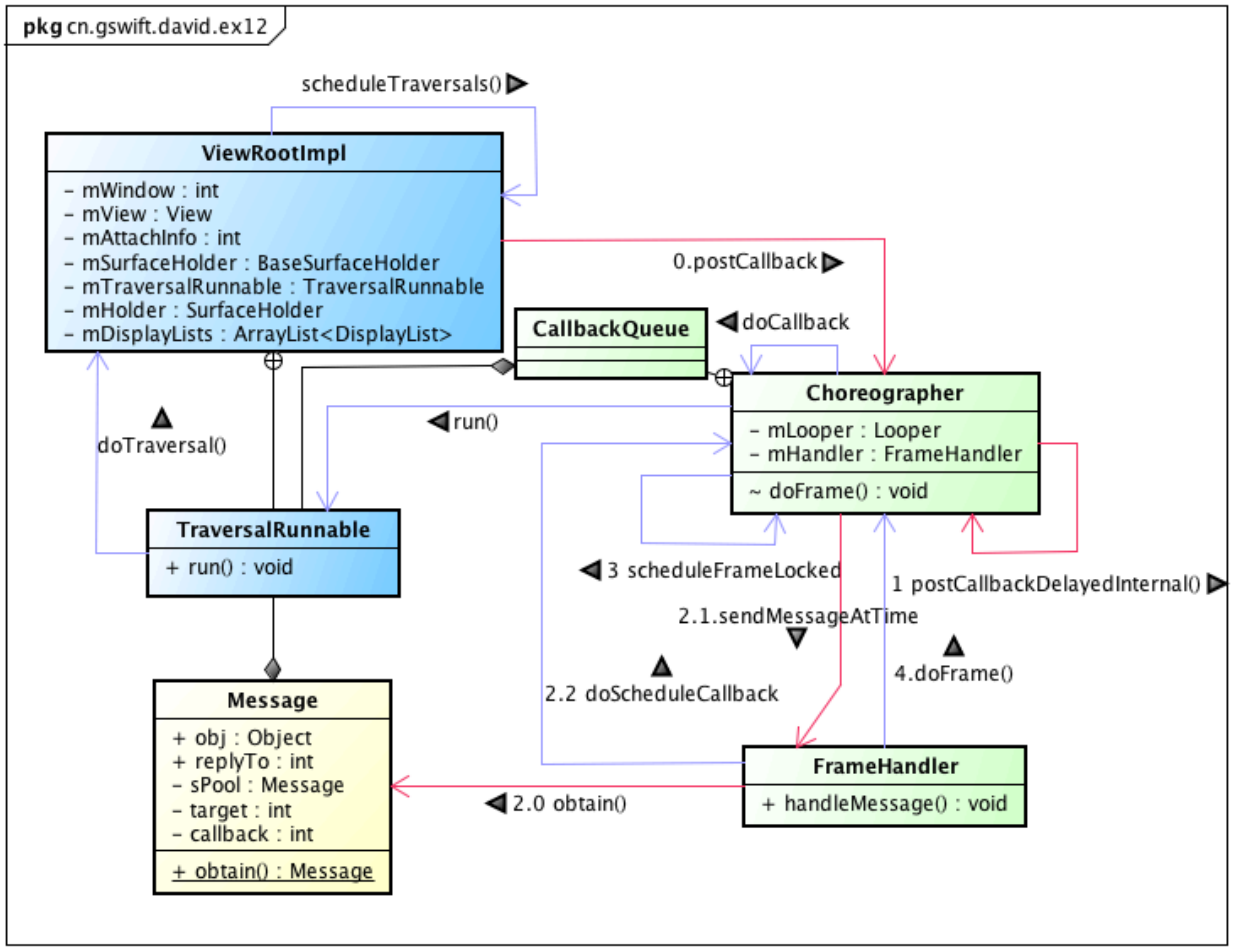


太極

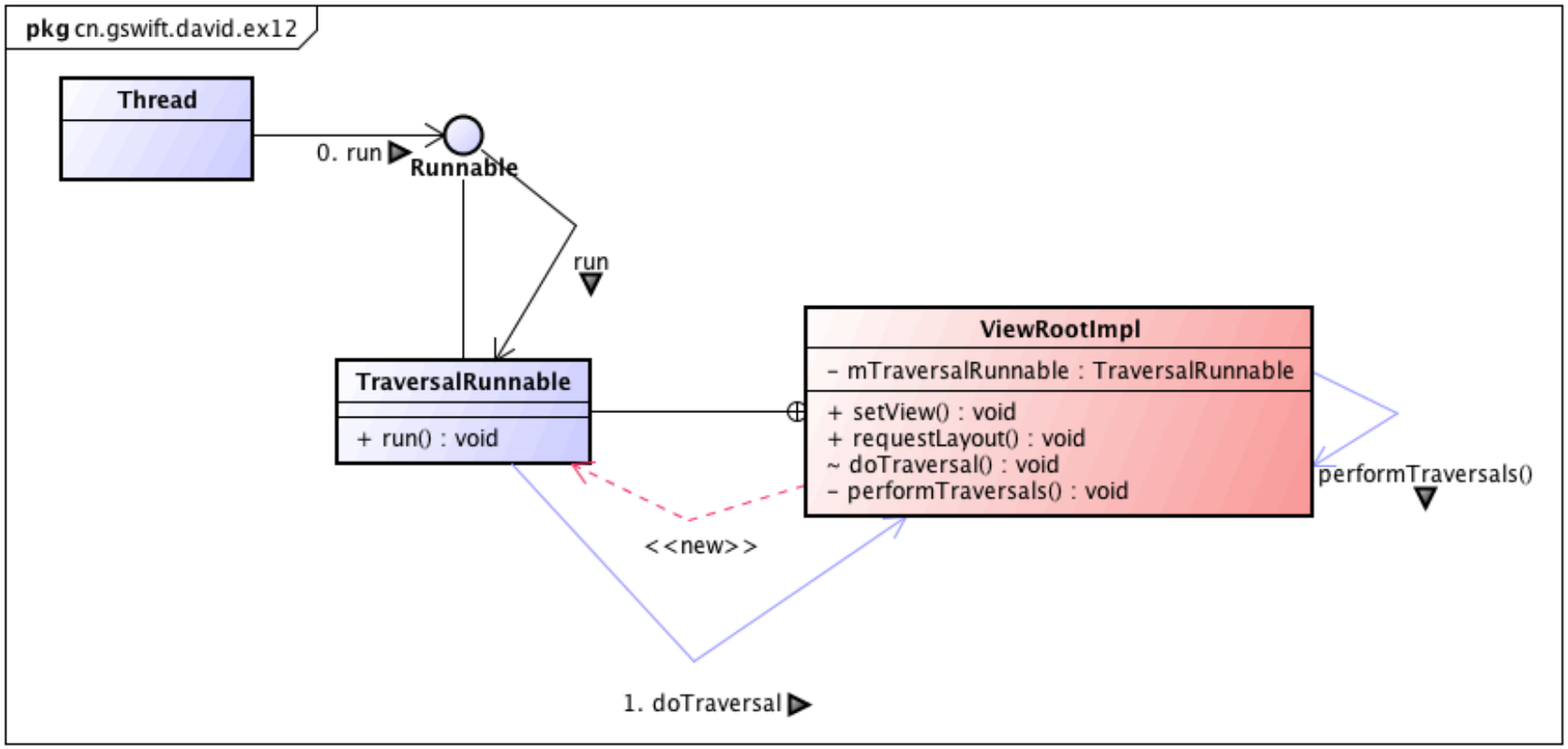
TAI CHI



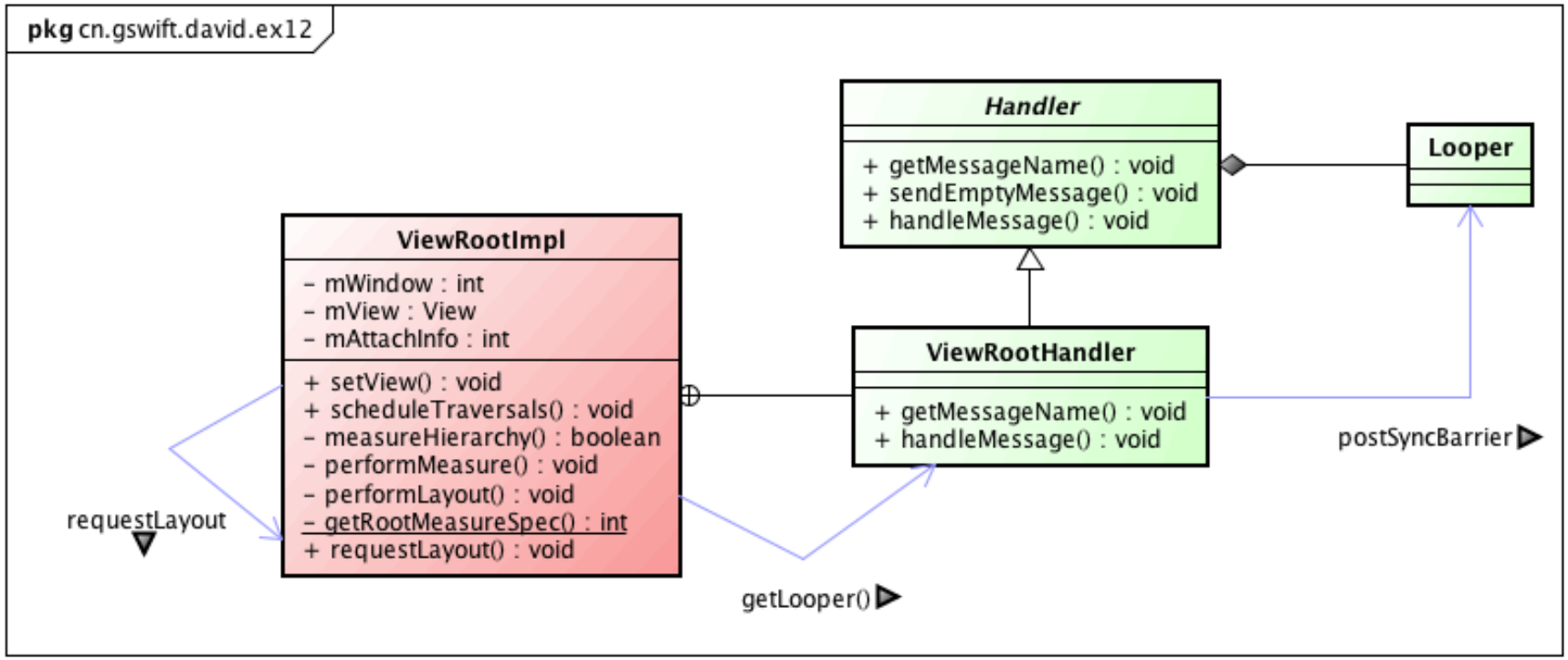
doTraveral Command



A child thread calls performTraversals



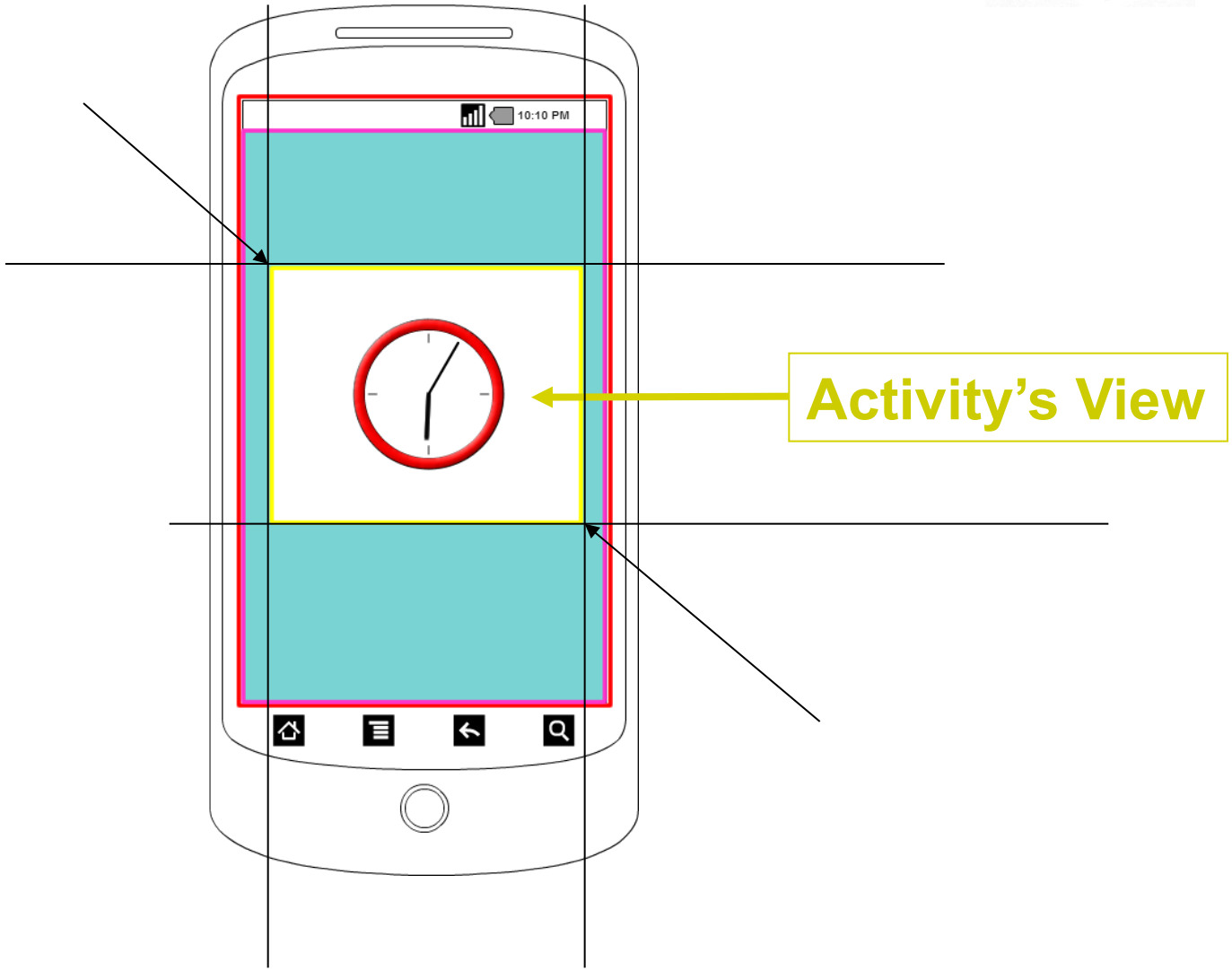
setView()->requestLayout()



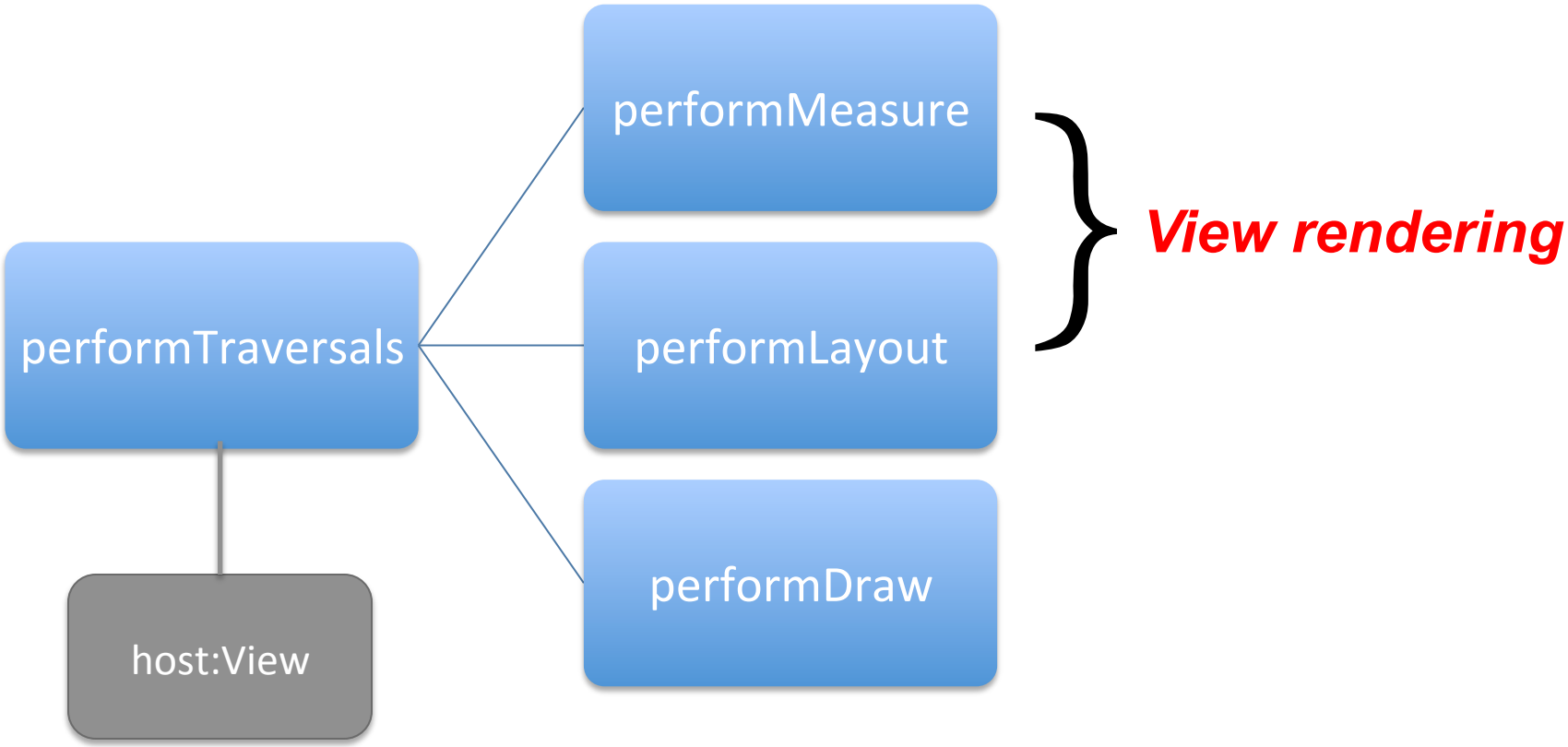
performTraversals

- how the view is laid out and positioned on the screen can only be determined by looking at it as part of the whole tree. This is because the position of every view affects the position of the next. In order to figure out where to draw a view and how big it should be, Android must do the drawing in two separate passes: a *measure pass* and a *layout pass.*

From: Android in Practice ,Charlie Collins Michael Galpin Matthias Käppler


10:10 PM
Activity's View

ViewRootImpl: performTraversals



A two-pass traversal

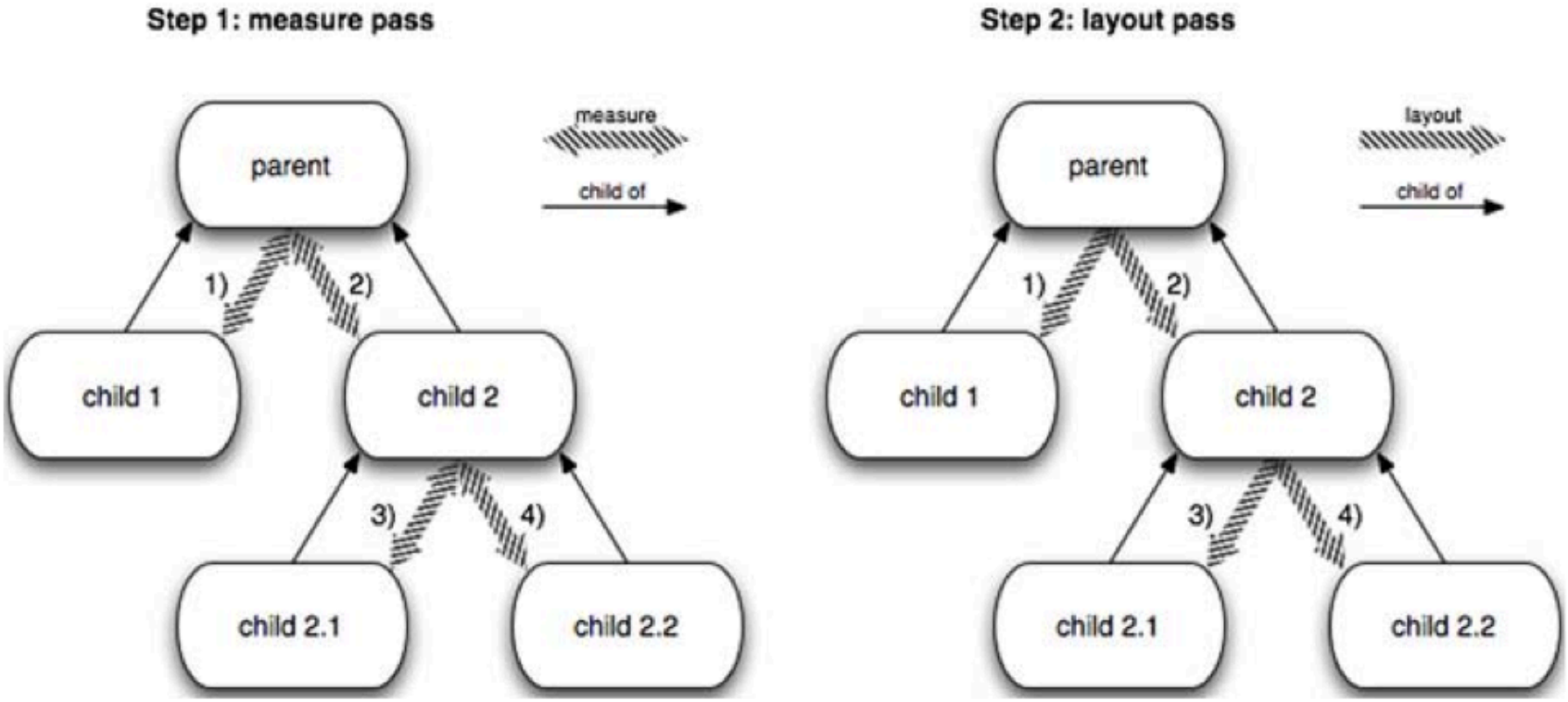


Figure 4.3 **Views are rendered using a two-pass traversal of the view tree. Pass one collects dimension specifications (left); pass two does the positioning on the screen (right).**

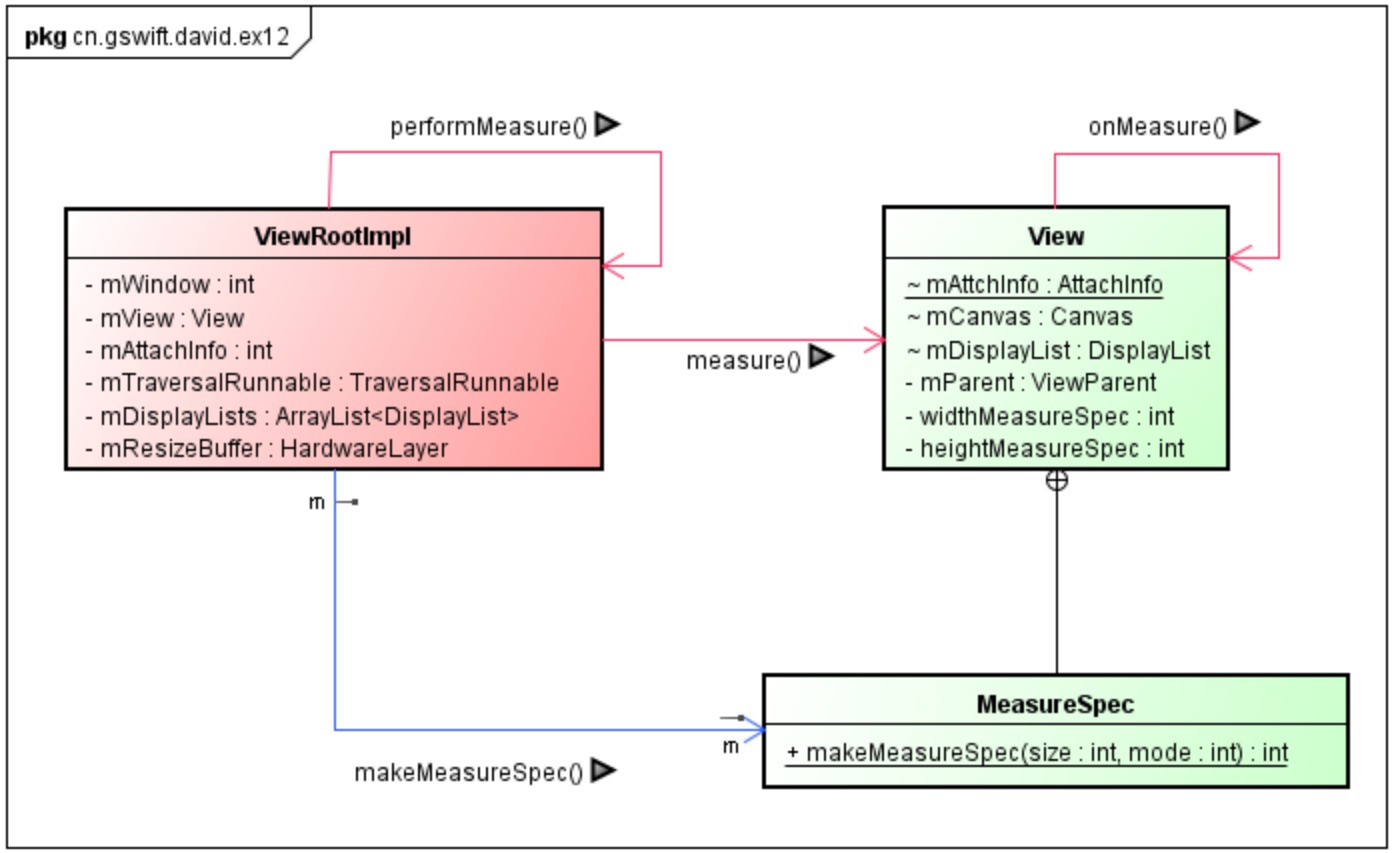
From: Android in Practice ,Charlie Collins Michael Galpin Matthias Käppler

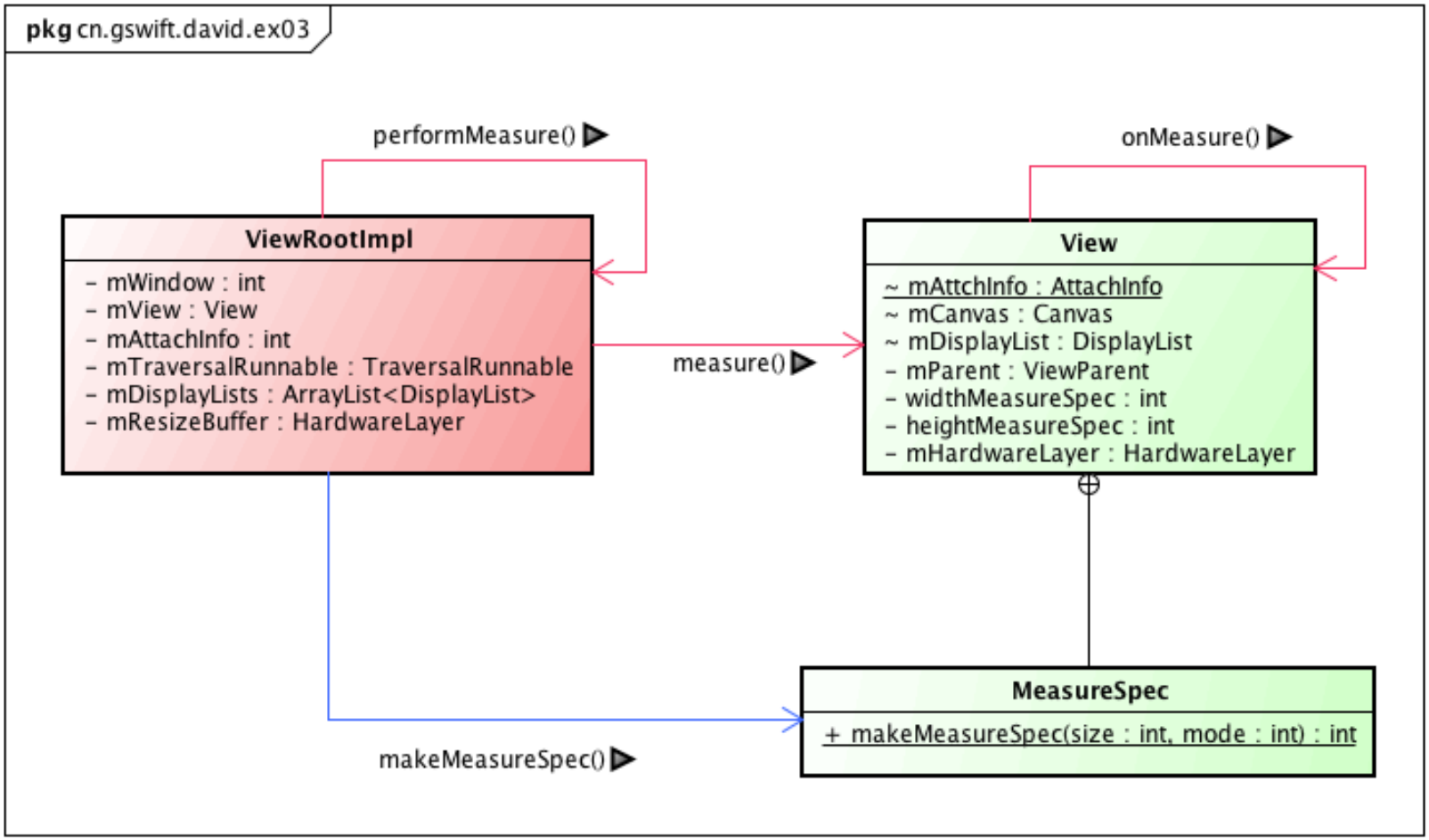
Measure pass

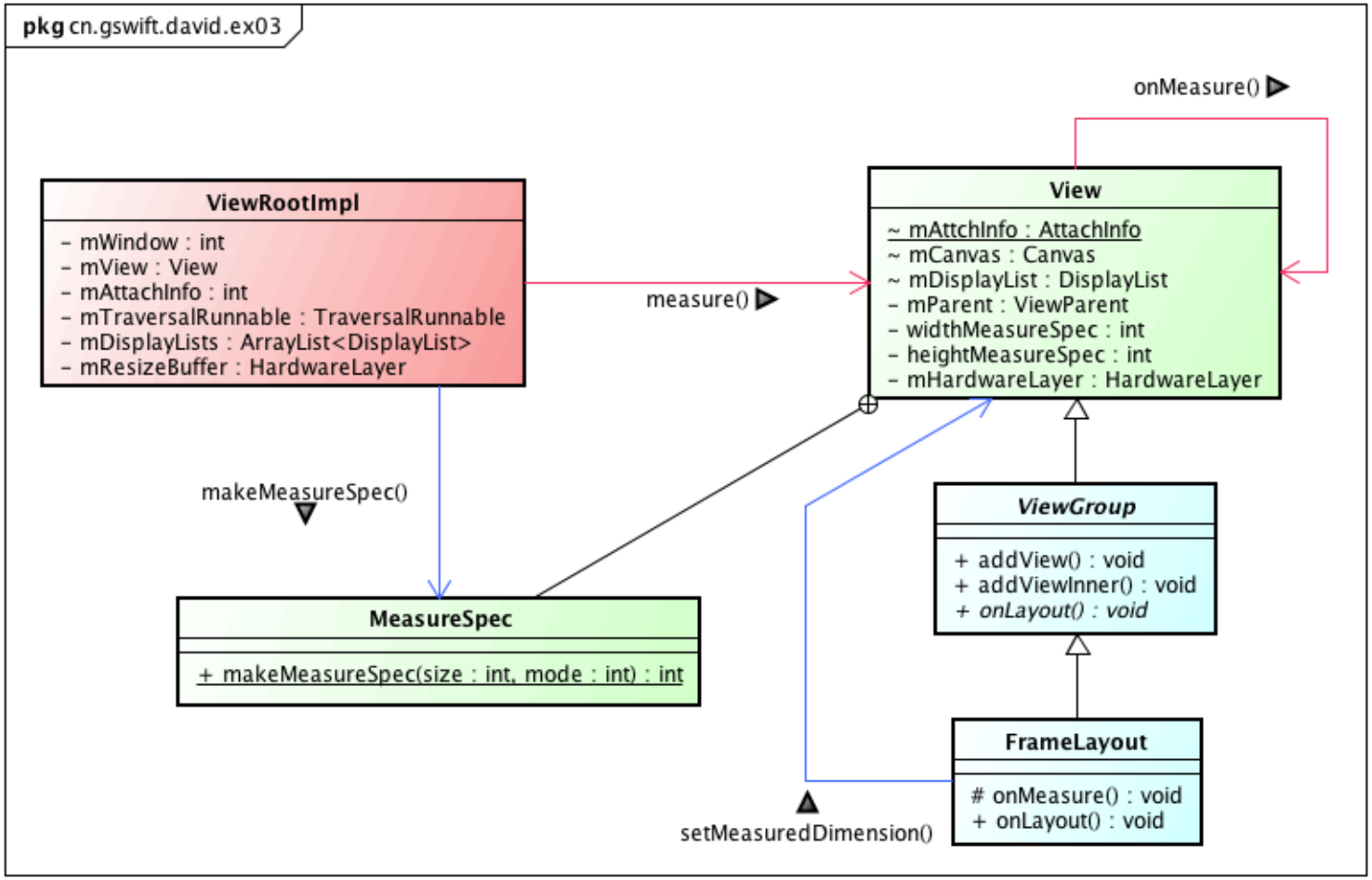
- During the measure pass, each parent view must find out how big their child views want to be by calling their measure method. This includes pushing a measure specification object down the tree that contains the size restrictions imposed by a parent view on a child. Every child must then find out how big it wants to be, while still obeying these restrictions.

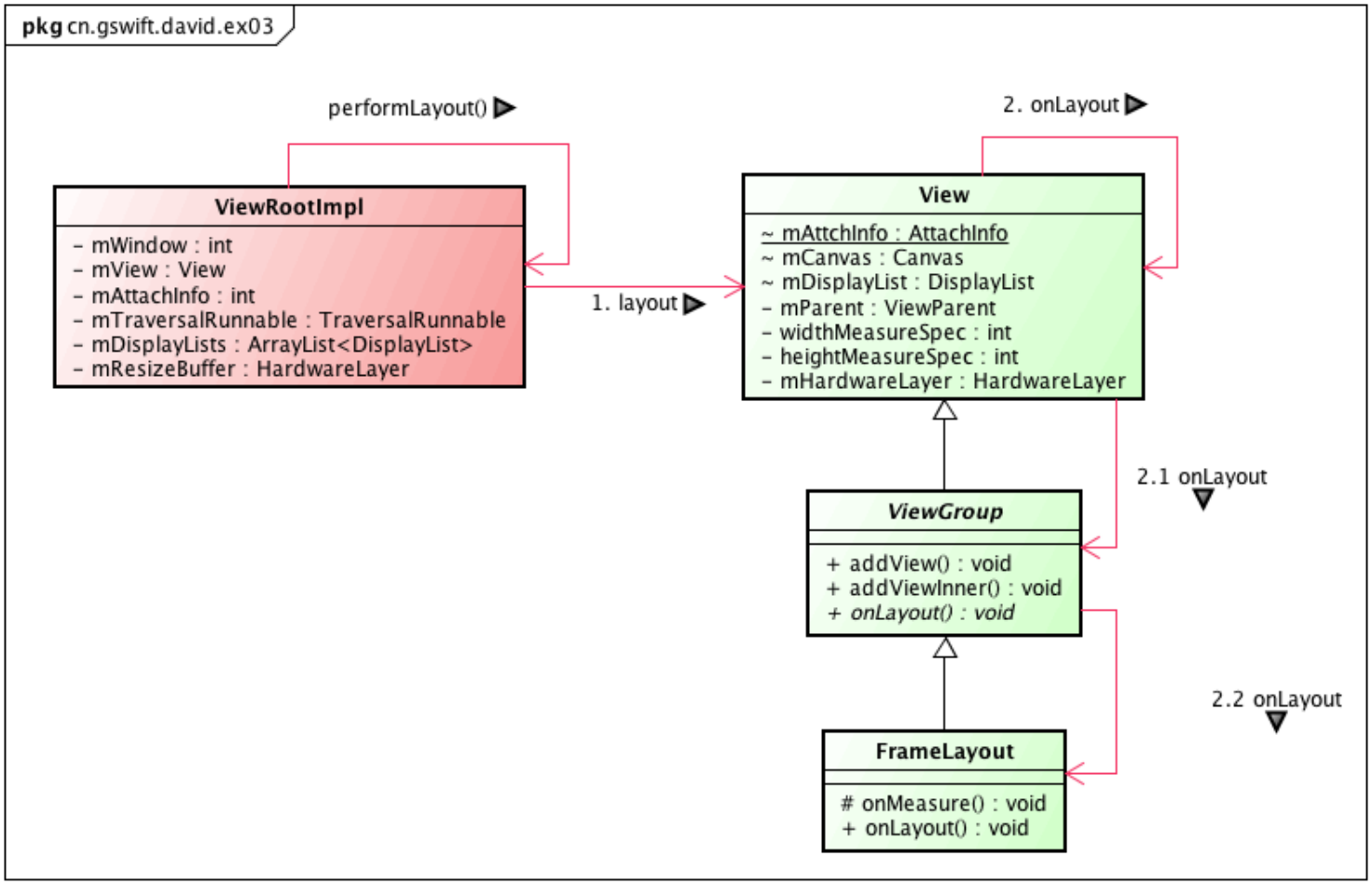
From: Android in Practice ,Charlie Collins Michael Galpin Matthias Käppler

Measure pass




pkg cn.gswift.david.ex03
performMeasure()
onMeasure()
ViewRootImpl
- mWindow : int
- mView : View
- mAttachInfo : int
- mTraversalRunnable : TraversalRunnable
- mDisplayLists : ArrayList<DisplayList>
- mResizeBuffer : HardwareLayer
measure()
View
~ mAttchInfo : AttachInfo
~ mCanvas : Canvas
~ mDisplayList : DisplayList
- mParent : ViewParent
- widthMeasureSpec : int
- heightMeasureSpec : int
- mHardwareLayer : HardwareLayer
MeasureSpec
+ makeMeasureSpec(size : int, mode : int) : int
makeMeasureSpec()


pkg cn.gswift.david.ex03
onMeasure()
ViewRootImpl
- mWindow : int
- mView : View
- mAttachInfo : int
- mTraversalRunnable : TraversalRunnable
- mDisplayLists : ArrayList<DisplayList>
- mResizeBuffer : HardwareLayer
measure()
View
~ mAttchInfo : AttachInfo
~ mCanvas : Canvas
~ mDisplayList : DisplayList
- mParent : ViewParent
- widthMeasureSpec : int
- heightMeasureSpec : int
- mHardwareLayer : HardwareLayer
makeMeasureSpec()
ViewGroup
+ addView() : void
+ addViewInner() : void
+ onLayout() : void
MeasureSpec
+ makeMeasureSpec(size : int, mode : int) : int
FrameLayout
onMeasure() : void
+ onLayout() : void
setMeasuredDimension()

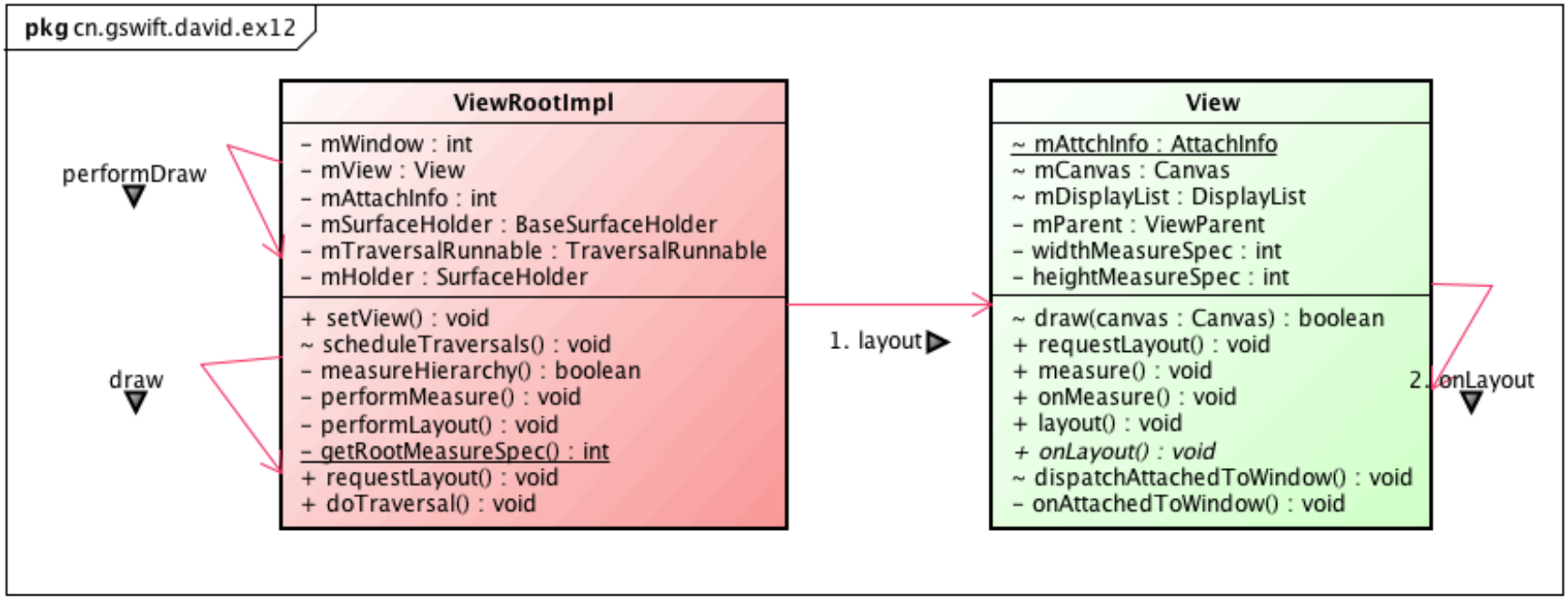

pkg cn.gswift.david.ex03
performLayout()
2. onLayout
ViewRootImpl
- mWindow : int
- mView : View
- mAttachInfo : int
- mTraversalRunnable : TraversalRunnable
- mDisplayLists : ArrayList<DisplayList>
- mResizeBuffer : HardwareLayer
1. layout
View
~ mAttchInfo : AttachInfo
~ mCanvas : Canvas
~ mDisplayList : DisplayList
- mParent : ViewParent
- widthMeasureSpec : int
- heightMeasureSpec : int
- mHardwareLayer : HardwareLayer
2.1 onLayout
ViewGroup
+ addView() : void
+ addViewInner() : void
+ onLayout() : void
2.2 onLayout
FrameLayout
onMeasure() : void
+ onLayout() : void

Layout pass

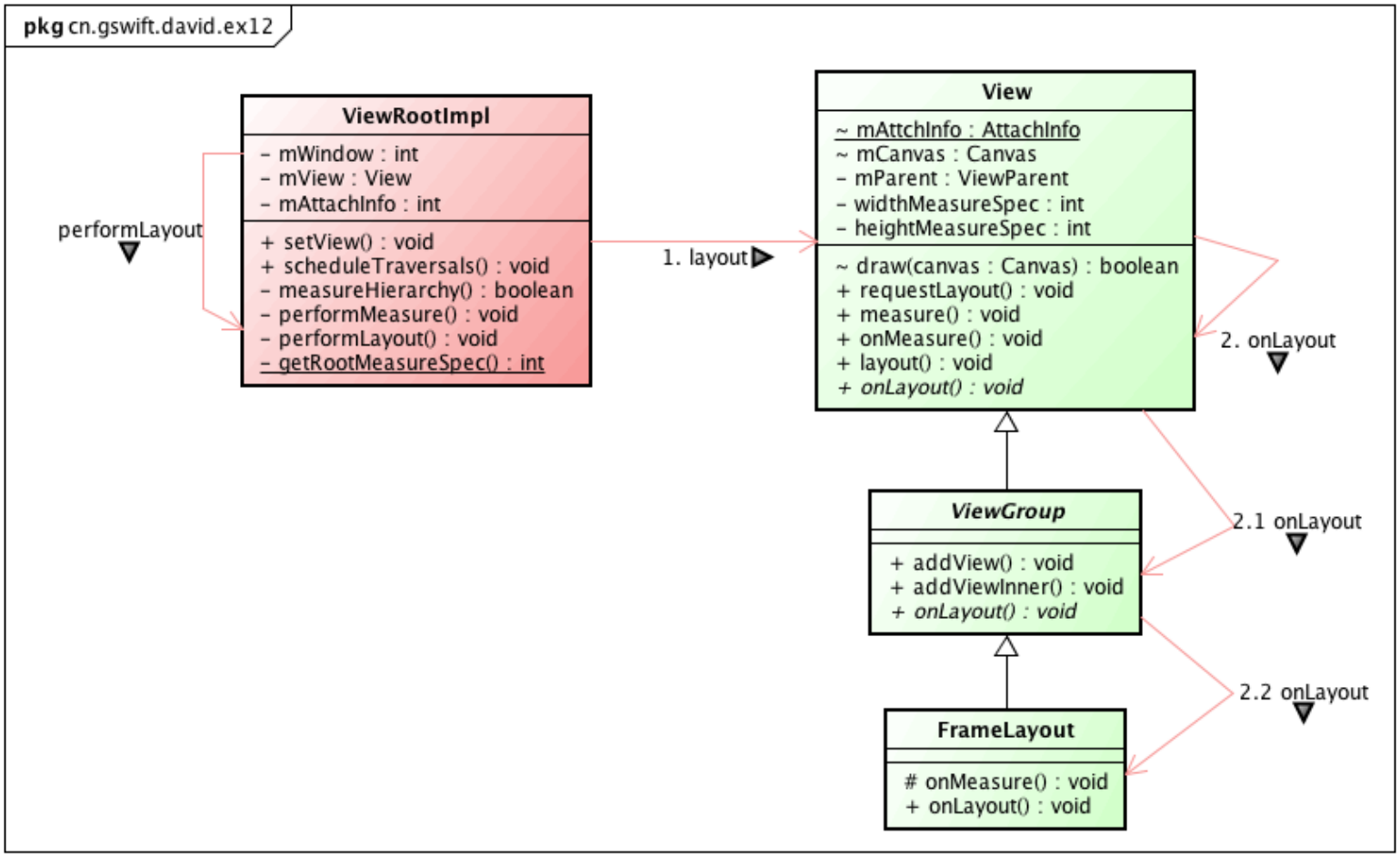
- Once all views have been measured, the layout pass is entered. This time, each parent must position every child on the screen using the respective measurements obtained from the measure pass by calling their layout method.

——《Android in Practice》 written by Charlie Collins Michael Galpin Matthias Käppler

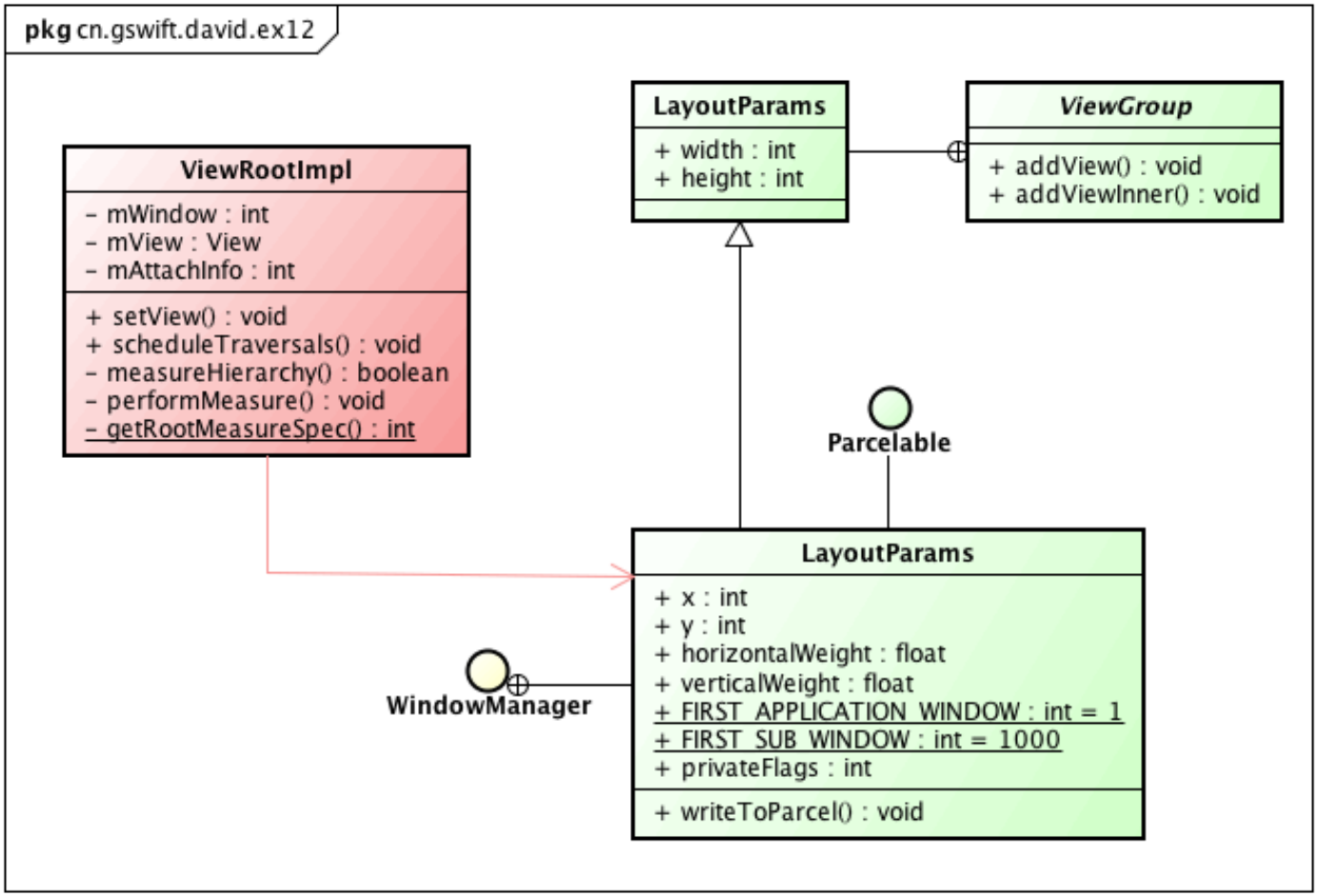
Layout Pass


pkg cn.gswift.david.ex12
performDraw
draw
ViewRootImpl
- mWindow : int
- mView : View
- mAttachInfo : int
- mSurfaceHolder : BaseSurfaceHolder
- mTraversalRunnable : TraversalRunnable
- mHolder : SurfaceHolder
+ setView() : void
~ scheduleTraversals() : void
- measureHierarchy() : boolean
- performMeasure() : void
- performLayout() : void
- getRootMeasureSpec() : int
+ requestLayout() : void
+ doTraversal() : void
1. layout
View
~ mAttchInfo : AttachInfo
~ mCanvas : Canvas
~ mDisplayList : DisplayList
- mParent : ViewParent
- widthMeasureSpec : int
- heightMeasureSpec : int
~ draw(canvas : Canvas) : boolean
+ requestLayout() : void
+ measure() : void
+ onMeasure() : void
+ layout() : void
+ onLayout() : void
~ dispatchAttachedToWindow() : void
- onAttachedToWindow() : void
2. onLayout

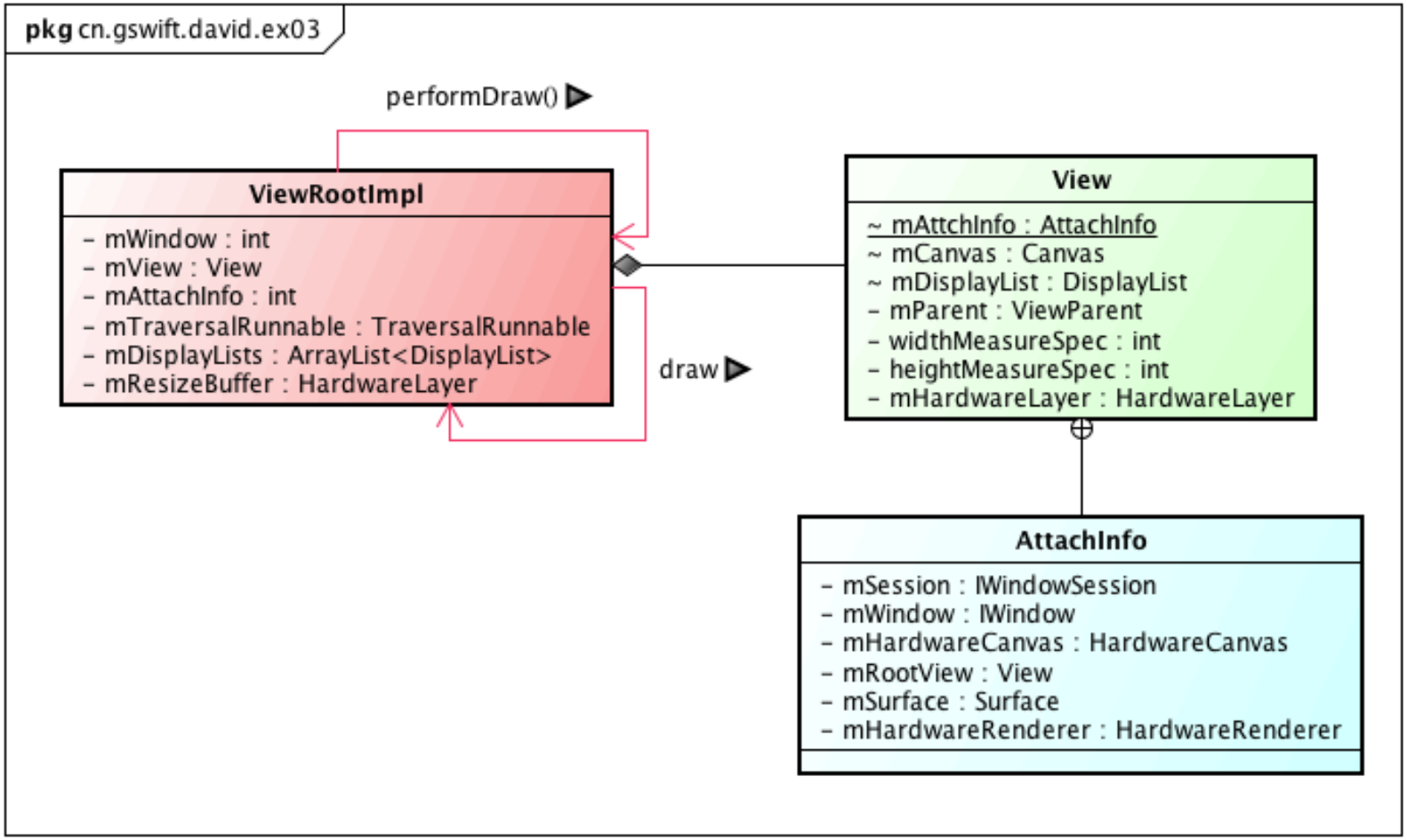
Layout pass eg:FrameLayout



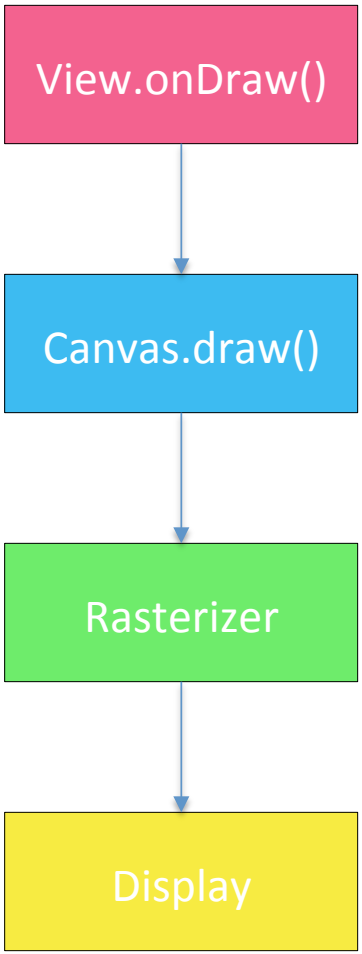
LayoutParams



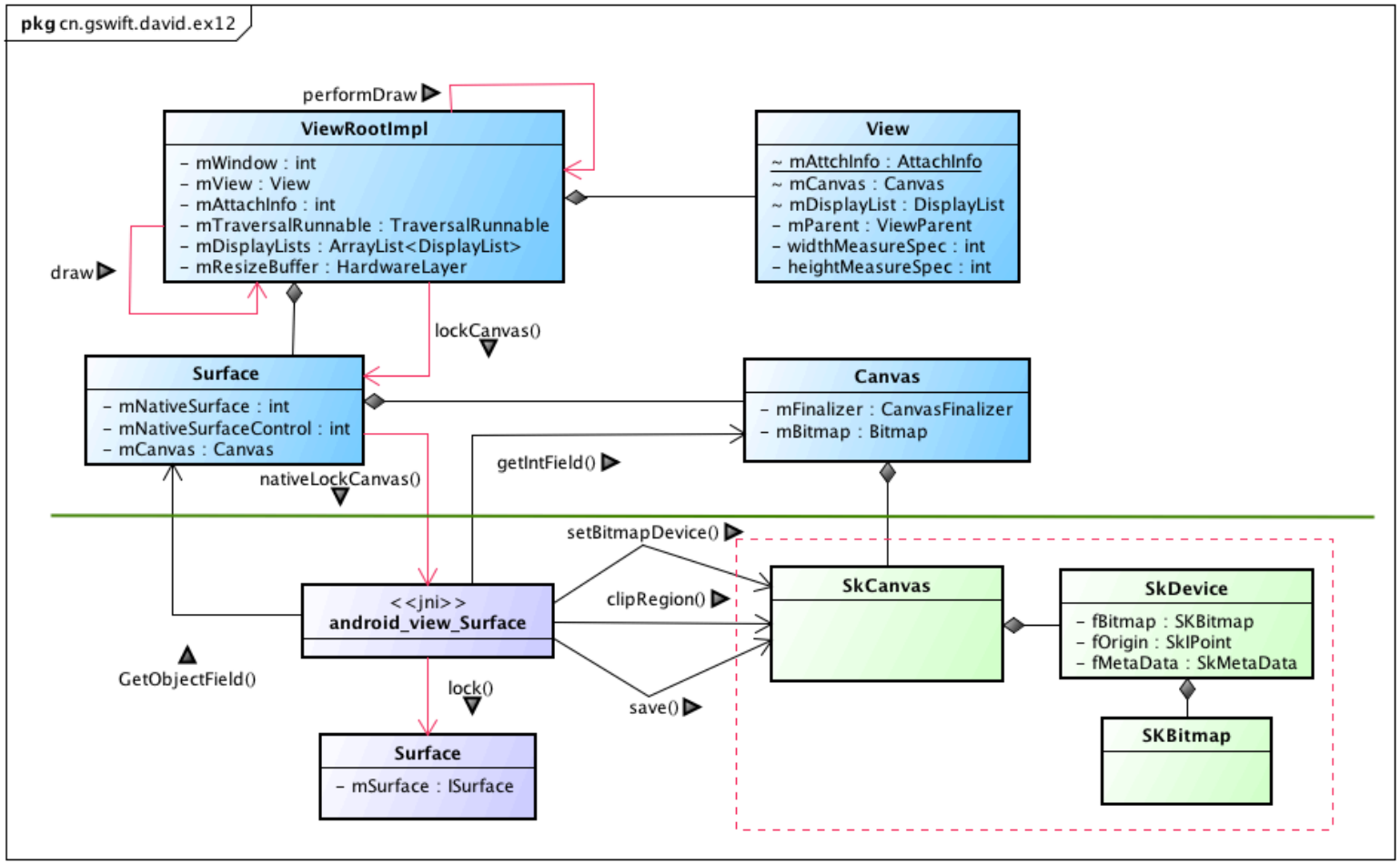
Draw Pass

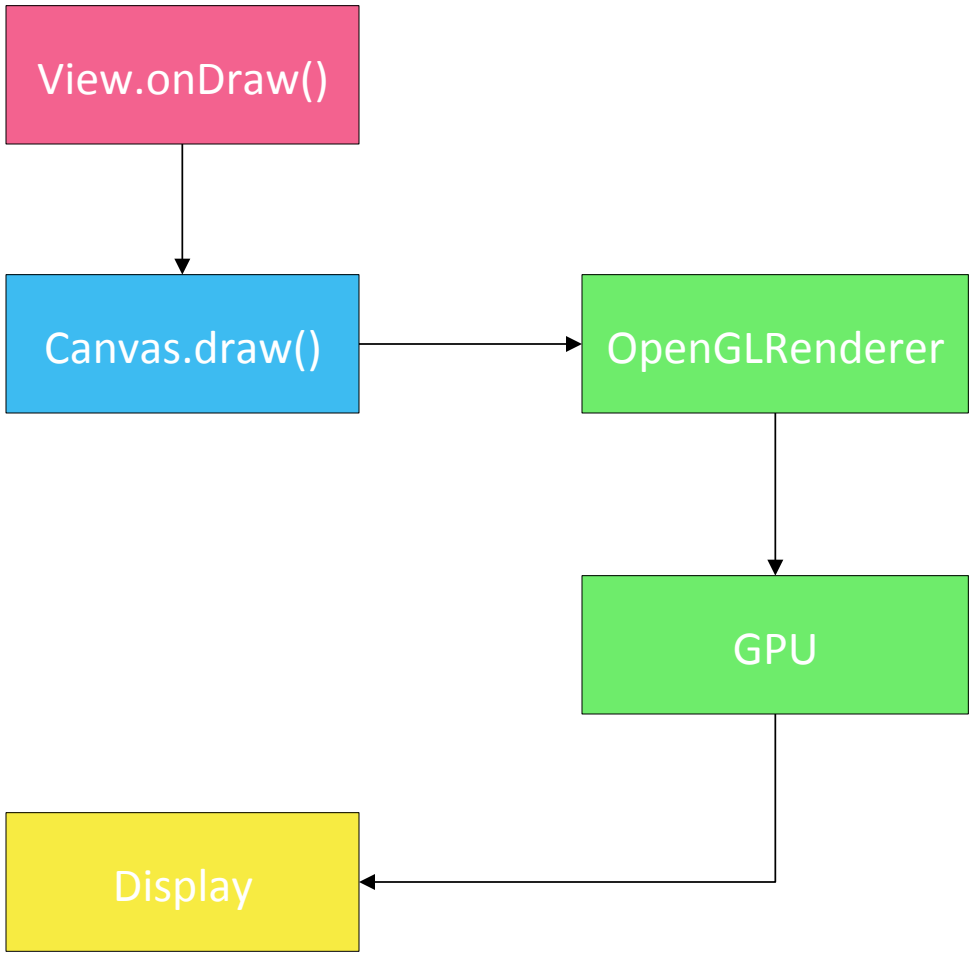

pkg cn.gswift.david.ex03
performDraw()
ViewRootImpl
- mWindow : int
- mView : View
- mAttachInfo : int
- mTraversalRunnable : TraversalRunnable
- mDisplayLists : ArrayList<DisplayList>
- mResizeBuffer : HardwareLayer
draw
View
~ mAttchInfo : AttachInfo
~ mCanvas : Canvas
~ mDisplayList : DisplayList
- mParent : ViewParent
- widthMeasureSpec : int
- heightMeasureSpec : int
- mHardwareLayer : HardwareLayer
AttachInfo
- mSession : IWindowSession
- mWindow : IWindow
- mHardwareCanvas : HardwareCanvas
- mRootView : View
- mSurface : Surface
- mHardwareRenderer : HardwareRenderer

Software Rendering

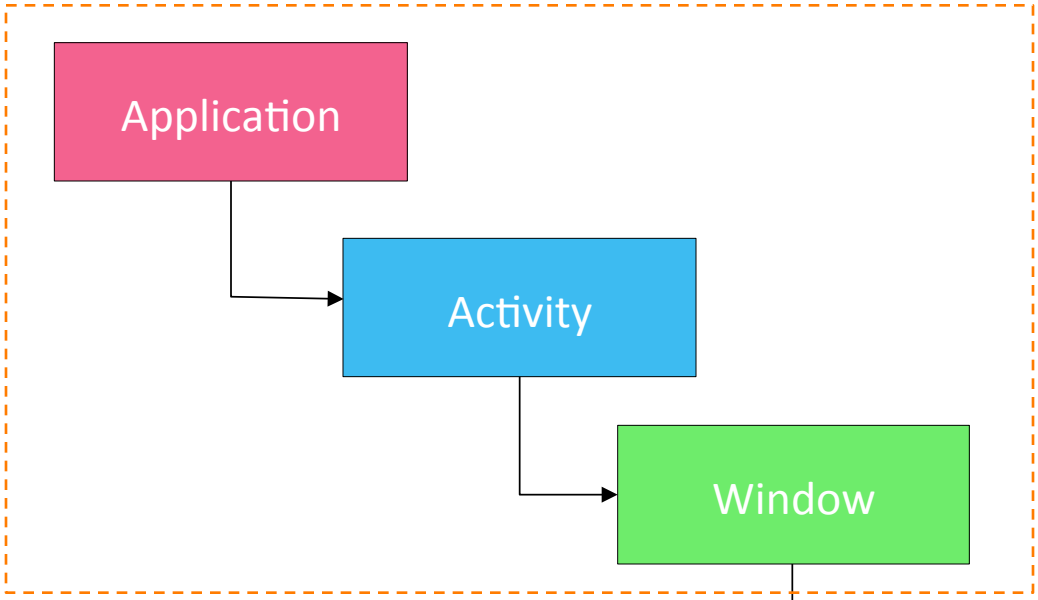


Drawsoftware()




View.onDraw()
Canvas.draw()
OpenGLRenderer
GPU
Display

HW Acceleration Control



Applicationlevel
<application android:hardwareAccelerated="true">
– Default value
• False in Android 3.x, True in Android 4.x
Activity
Window

WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED
View setLayerType(View.LAYER_TYPE_SOFTWARE,null)

Next->
GO to Android’s Renderer

参考资料：

Android4.2 http://code.metager.de/source/xref/android/4.2/

Android in Practice ,Charlie Collins Michael Galpin Matthias Käppler

Android Accelerated Rendering, Romain Guy Chet Haase, May 11, 2011, http://www.google.com/events/io/2011/sessions/accelerated-android-rendering.html

About Me

- I have been working as a product-designer specializing in software/Web application design and development. I am passionate about mobile application development and became interested in Android programming when the platform was launched by Google. Thus I was not programming on Android projects, I spent spare time reading technical blogs, researching, analyzing, and testing mobile applications, as a software consultancy specialized in android technologies.
- In my product-design time, in the developing, I've encountered too many program manage troubles that suffer due to poor communication and code design, I know that help them to understand the system framework is very important. I amd experienced in system and application layers, my goal is simple: help someone who wishes to better understand the **Android framework** in java、JNI and C/C++ libraries.

- Please also check my article and slides on this http://blog.sina.com.cn/gswift

Contact: Zhiyong.liu@aliyun.com

http://weibo.com/gswift