LOVING LEAN LAYOUTS

PRESENTED BY HUYEN TUE DAO

https://speakerdeck.com/randomlytyping/babbq-2015-loving-lean-layouts

https://github.com/queencodemonkey/Android-Loving-Lean-Layouts

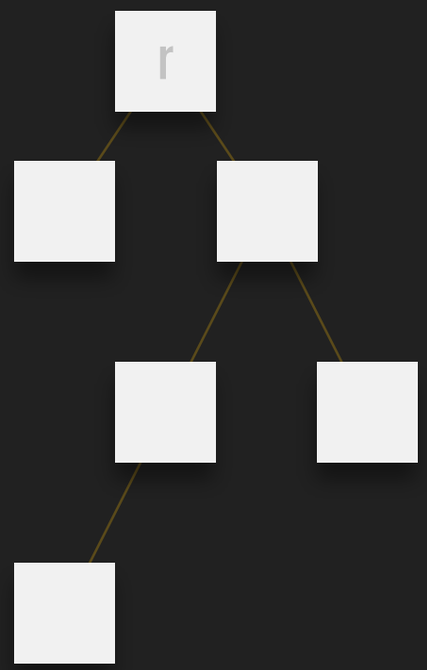
LOVING LEAN LAYOUTS

- Why does it matter?
- Analyze layouts
- Find/fix problem areas
- Good practices

WHY DOES IT MATTER?

- Performance: complexity ∝ resources (memory, time)
  - # of views, depth of view hierarchy
  - # of times measurement and layout executed
  - UI/animations: jank-ful + jittery
- Maintainability: complexity ∝ effort to maintain/refactor
  - Readability
  - Stability: a change to one part affects other part

HOW TO LAYOUT A VIEW HIERARCHY



STEP 1

measure

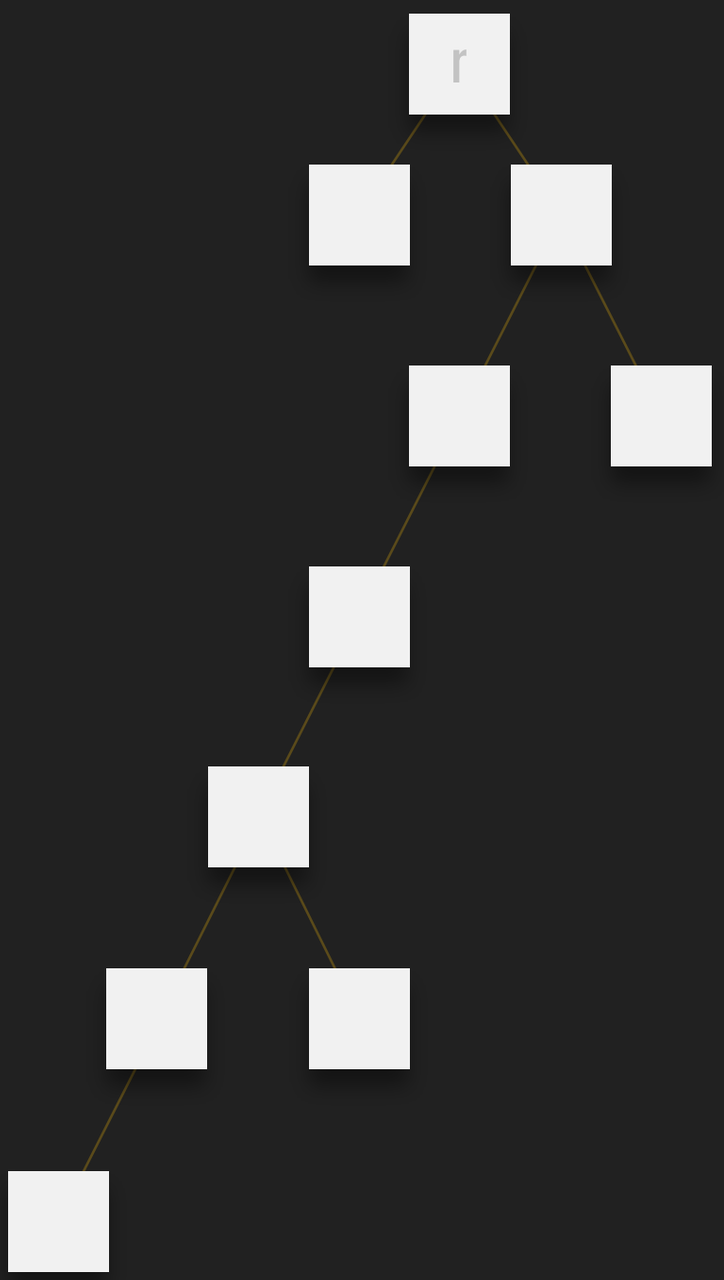
STEP 2

layout

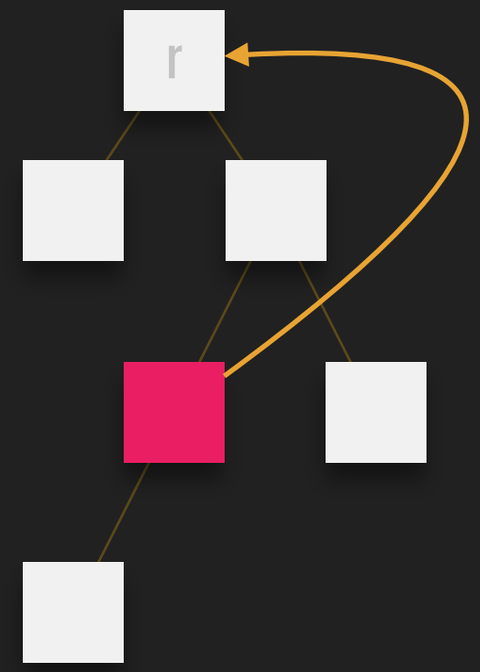
WHERE'S THE PROBLEM?

(HINT: SEVERAL PLACES)

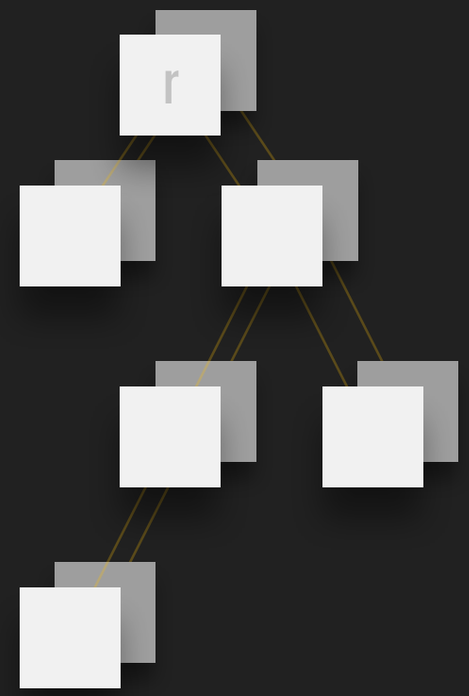
Deep hierarchies increase complexity and dependency



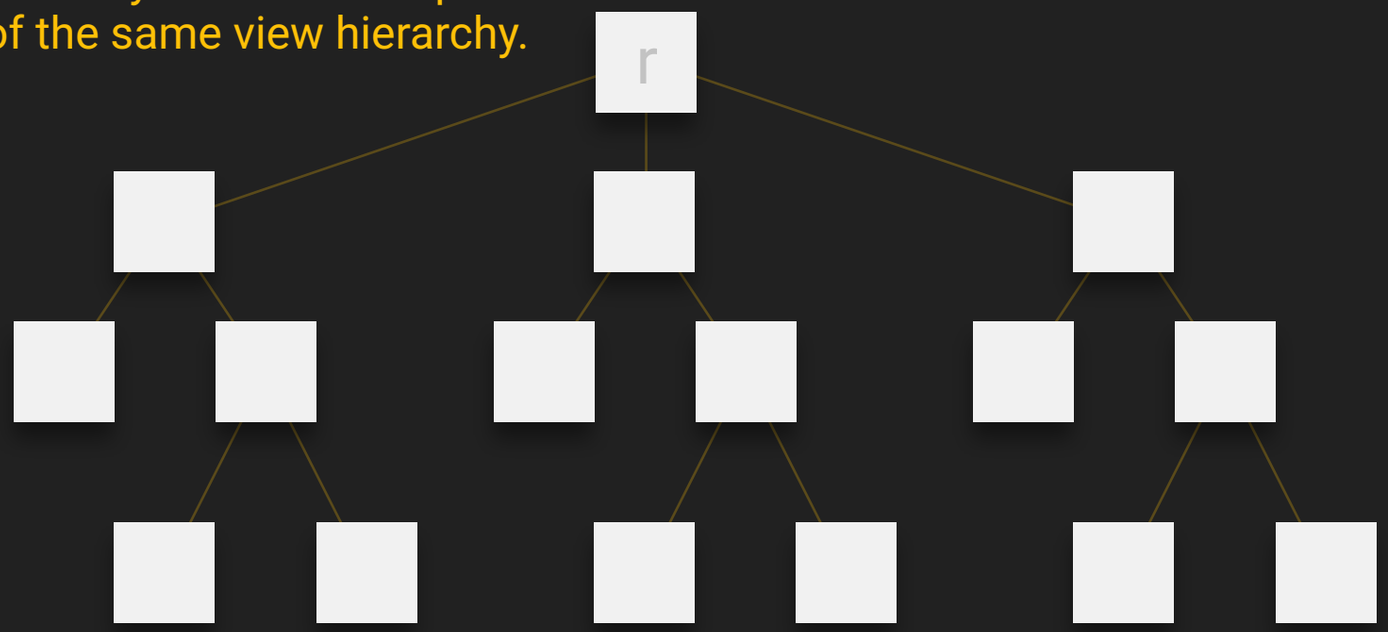
Size/position change instantiates measure/layout starting at root.



Some layouts require multiple measure/layout passes, e.g. RelativeLayout



Lists naturally result in multiple copies of the same view hierarchy.



HIERARCHY VIEWER

- SDK tool for visualizing view hierarchy
  - Used to be standalone; deprecated
  - Now in Android Device Monitor
- Tree representation of view hierarchy
- Evaluates performance of measure, layout, and draw phases
- Best: use on physical device running 4.1 up

OTHER MEASURING TOOLS

- **Systrace**: system + application process execution data → interactive reports
  - Rendering frame → how much time measure/layout takes up.
- **dumpsys**: collects and dumps “interesting information” about system services status
  - `adb shell dumpsys gfxinfo <PACKAGE_NAME>` → perf info related to frames of animation
  - `adb shell dumpsys gfxinfo <PACKAGE_NAME> framestats` → detailed frame timing info; on M

LEARNING FROM LINT

- Points out things that are generally not good ideas.
- Good candidates for things you should fix first.
- Generally, straightforward points:
  - Don’t nest weights
  - Remove useless views
  - Don’t nest too hard

SIMPLIFY AND REDUCE

- Usually many different ways to do one thing: opt for simplest
- Often there are view attributes/features that take the place of multiple views.
  - `TextView` compound drawables
  - `Spannable`
  - `LinearLayout`, `android:divider`
- If you need a placeholder/divider, just use a plain ol' `View`
- Sometimes it just takes picking the right view/layout.

SOMETIMES CUSTOM IS THE WAY

- Sometimes the answer is a custom `ViewGroup`.
- Total control over measure/layout contents.
- Mitigate double-layout phases from platform layouts.
- Starting out? Try using with straightforward layouts.
- Balance performance benefits with development effort.
- Also: totally custom `Views`.

GENERAL GAME PLAN

- Anticipate and develop good habits
- Know how to identify possible problem areas
- Use simplest solutions where possible
  - Fewer Views/ViewGroups
  - Flatter hierarchies
- Don’t let problems accumulate
- Balance performance benefit with effort required

THANKS FOR COMING!

https://github.com/queencodemonkey/Android-Loving-Lean-Layouts

Huyen Tue Dao

@queencodemonkey
+HuyenTueDao

huyen@randomlytyping.com
randomlytyping.com

speakerdeck.com/queencodemonkey

REFERENCES

- Hierarchy Viewer: http://developer.android.com/tools/help/hierarchy-viewer.html
- Hierarchy Viewer Walkthrough: http://developer.android.com/tools/performance/hierarchy-viewer/index.html
- Testing Display Performance: http://developer.android.com/training/testing/performance.html
- Analyzing UI Performance with Systrace: http://developer.android.com/tools/debugging/systrace.html

REFERENCES

- Custom ViewGroups: https://sriramramani.wordpress.com/2015/05/06/custom-viewgroups/
- Optimizing Layout Hierarchies: http://developer.android.com/training/improving-layouts/optimizing-layout.html
- Android Performance Patterns, Double Layout Taxation: https://www.youtube.com/watch?v=dB3_vgS-Uqo
- Android Performance Pattern, Invalidations, Layouts, and Performance: https://youtu.be/we6poP0kw6E