Android @ 60fps

Phil Lopreiato
GWU Systems Hacking Club
29 January 2016

Remember this guy? ...

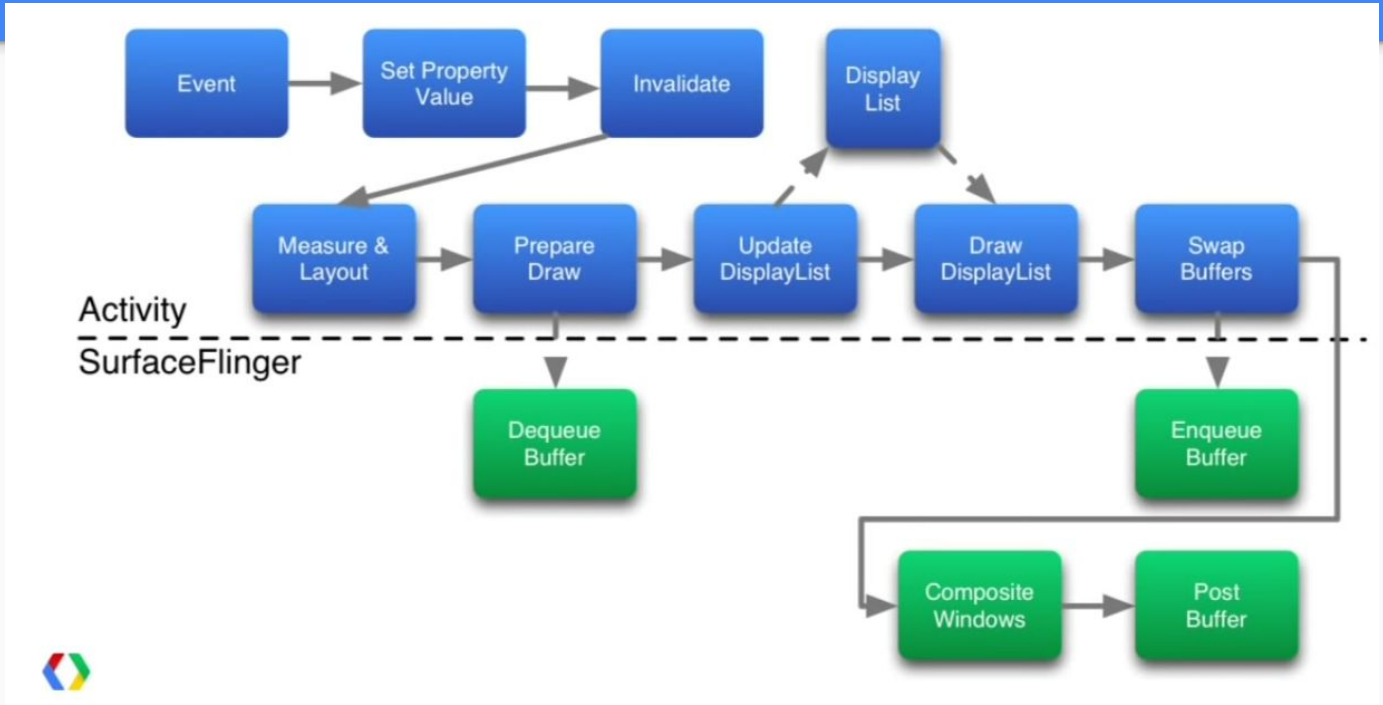
Today's Focus

Overall Theme:

Jank is bad!!!

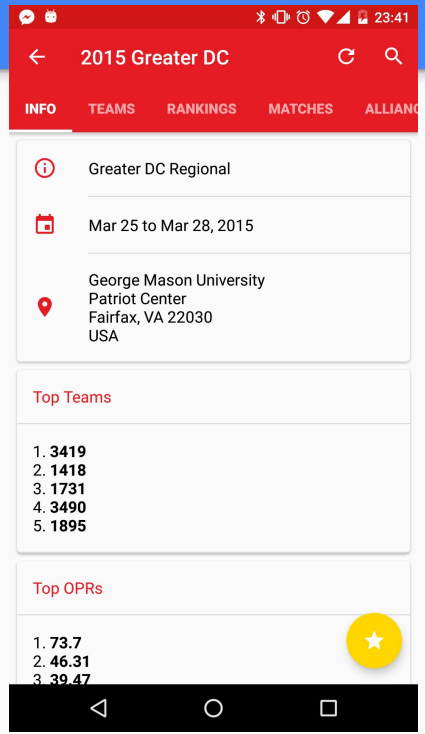
mmmm… butter

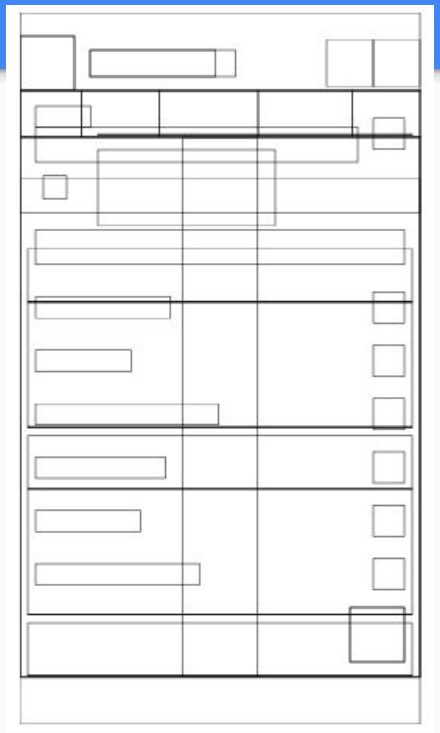
The Full Rendering Pipeline… Look complex?

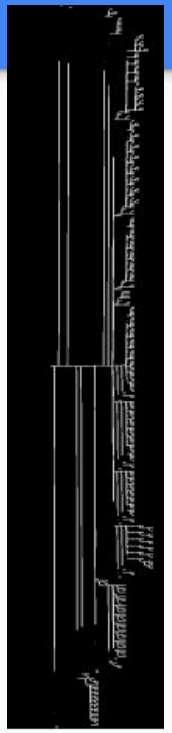


“For Butter or Worse”, Google I/O 2012

How many views* are in this screenshot?







- A lot.
- We see the view hierarchy is pretty complex
- A lot of “magic” done by the rendering system

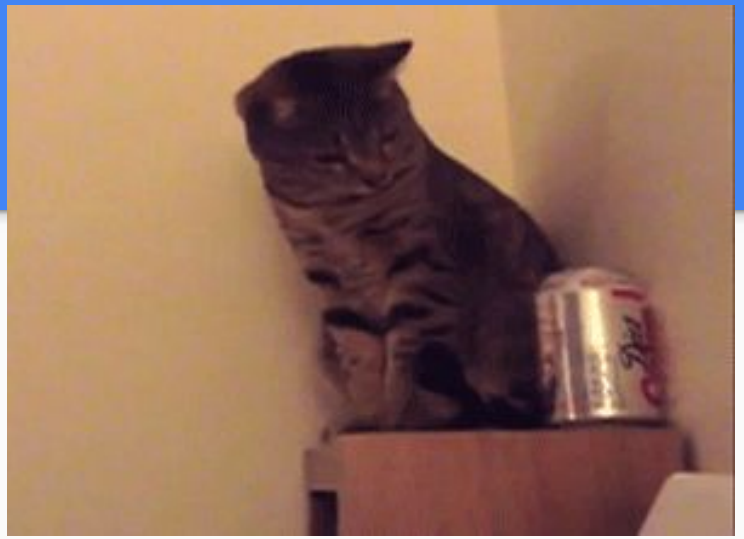
* A “view” is a basic display component, like a text box, button, or layout container.

All rendering is done on *a single thread*

- Basically, this means we can avoid concurrency hell
  - Deadlocks, race conditions, all kinds of icky stuff
  - The solution is to just not deal with it
- Plus, the screen hardware can only deal with one buffer at a time

Here’s where we’re at

- LOTS of views
- ONE thread
- 60 fps

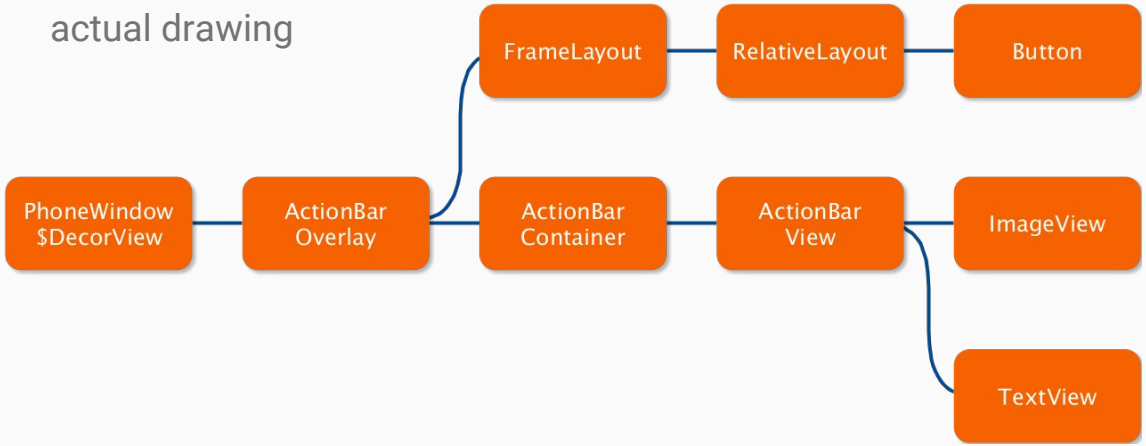


So what happens when we “do stuff”? (Set text, draw animation, change color)

Chaos.

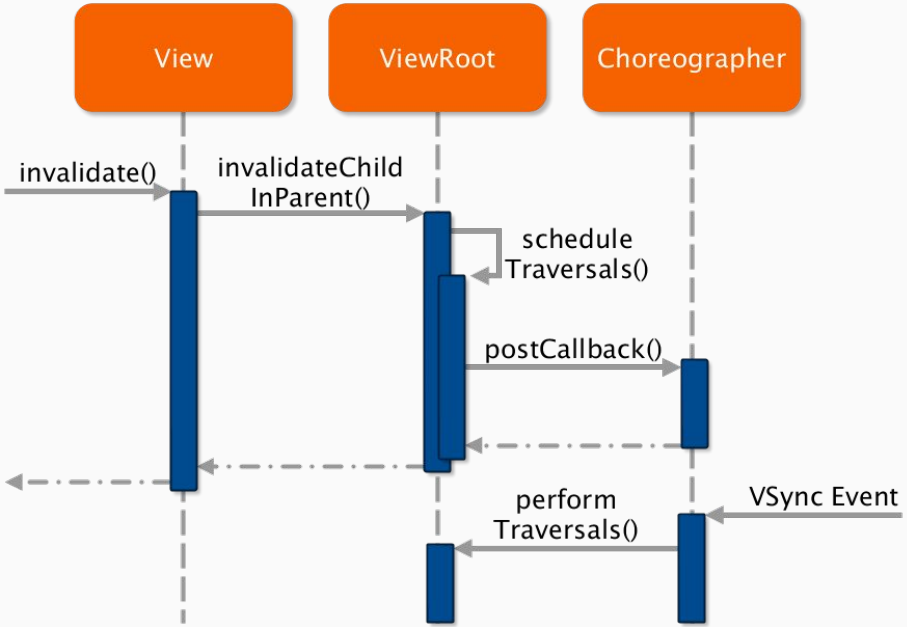
The Root View

- Every Activity has a ViewRoot - a single view at the top of the hierarchy with a single Child
  - This view is responsible for scheduling view invalidation and actual drawing

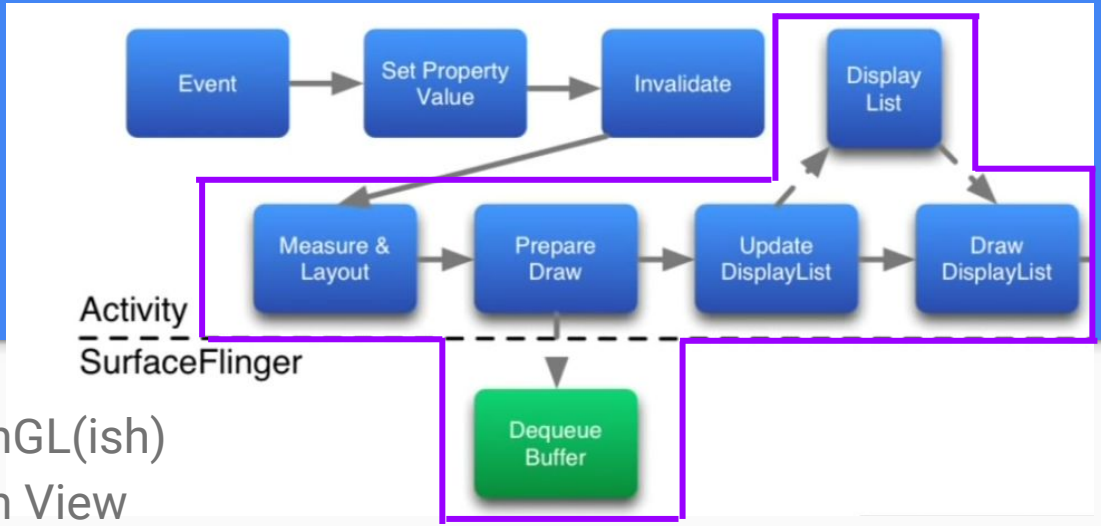


View Invalidation

- When a View property changes, it gets **invalidated**, which tells the framework it needs to be redrawn



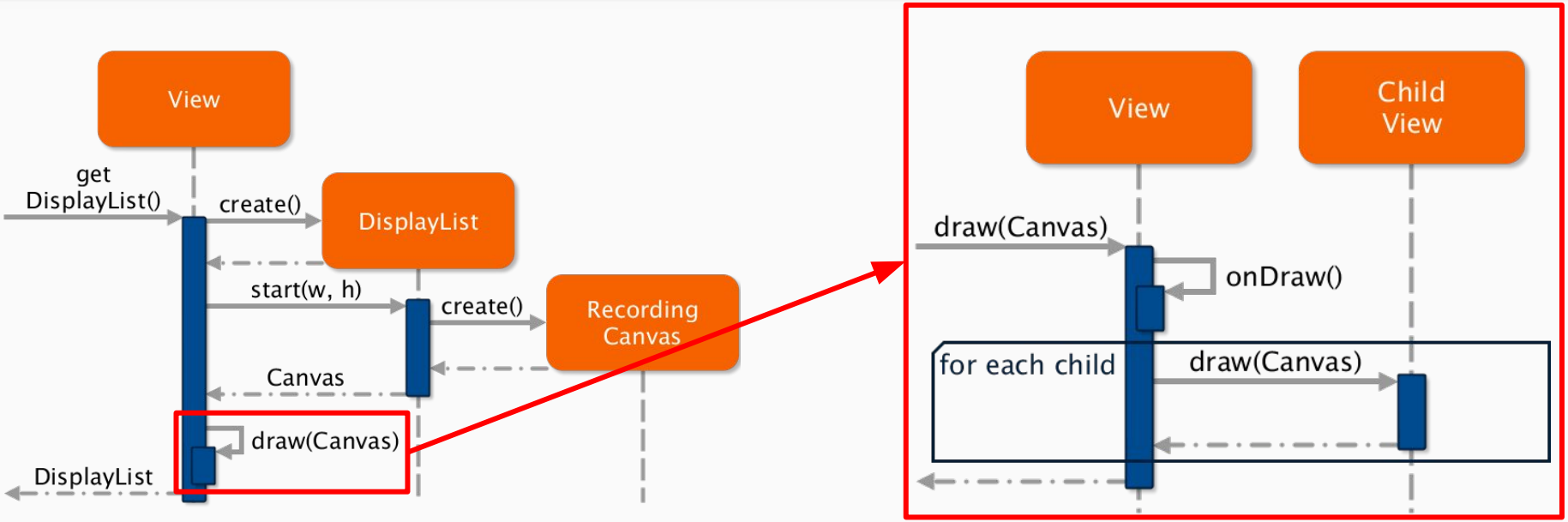
DisplayList



- A precomputed list of OpenGL(ish) Commands to draw a given View
- Each View computes its own DisplayList after it gets invalidated
  - This happens in the *onDraw* method
- DisplayLists can be nested in order to properly draw child Views

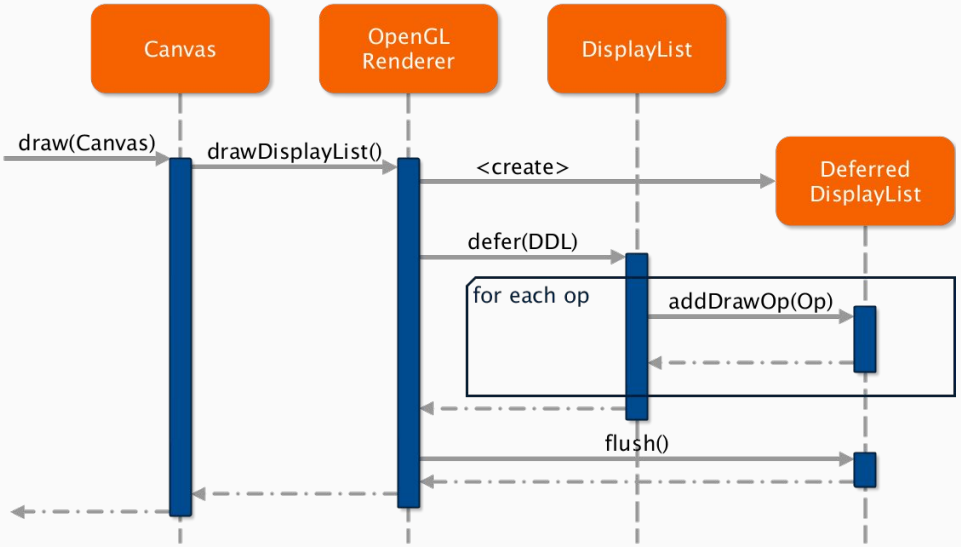
```
Save 3
DrawPatch
Save 3
ClipRect 20.00, 4.00, 99.00, 44.00, 1
Translate 20.00, 12.00
DrawText 9, 18, 9, 0.00, 19.00, 0x17e898
Restore
RestoreToCount 0
```

Basic Drawing



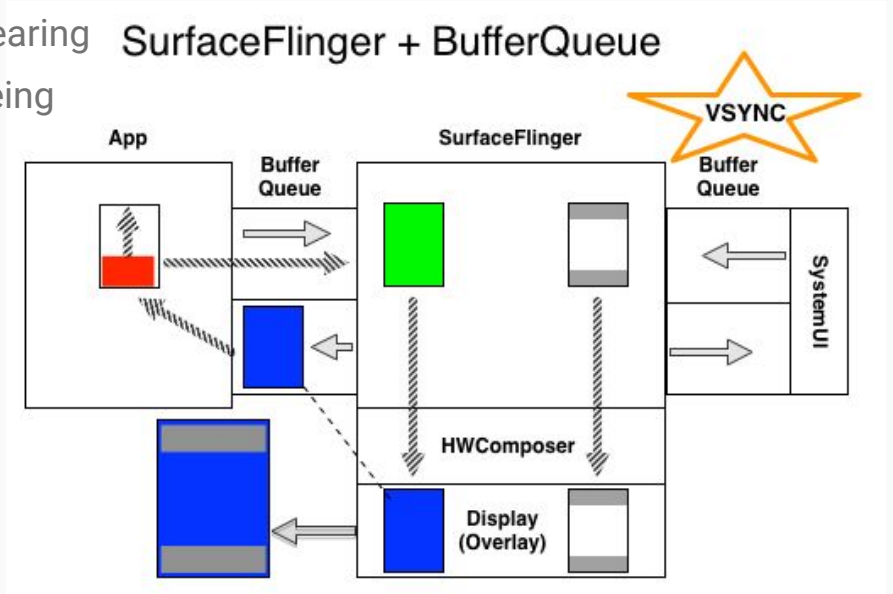
Inside draw()

- Call out to OpenGLRenderer to actually make pixels
  - This is the first native C++ class
- Creates a buffer that will get sent to the display hardware



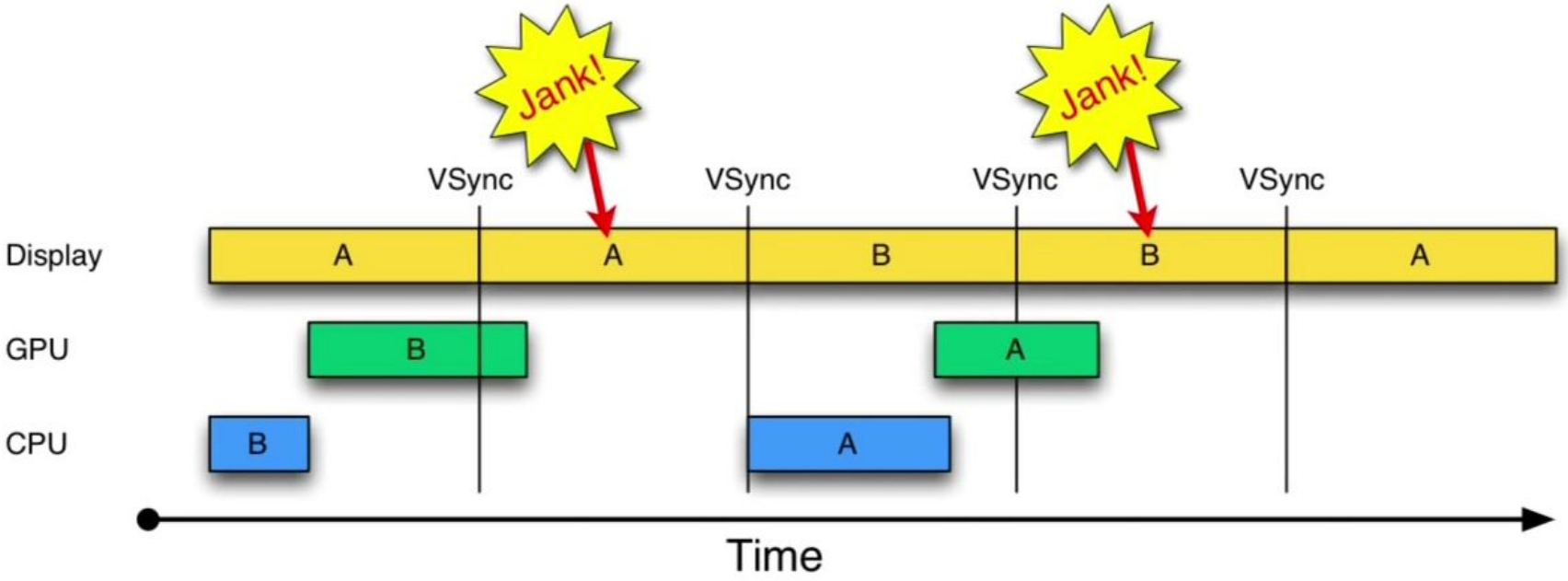
So many buffers

- Android display pipeline is triple-buffered
  - Two buffers are needed to avoid display tearing
  - One is currently shown, and the other is being prepared in the background
    - They get swapped every VSYNC



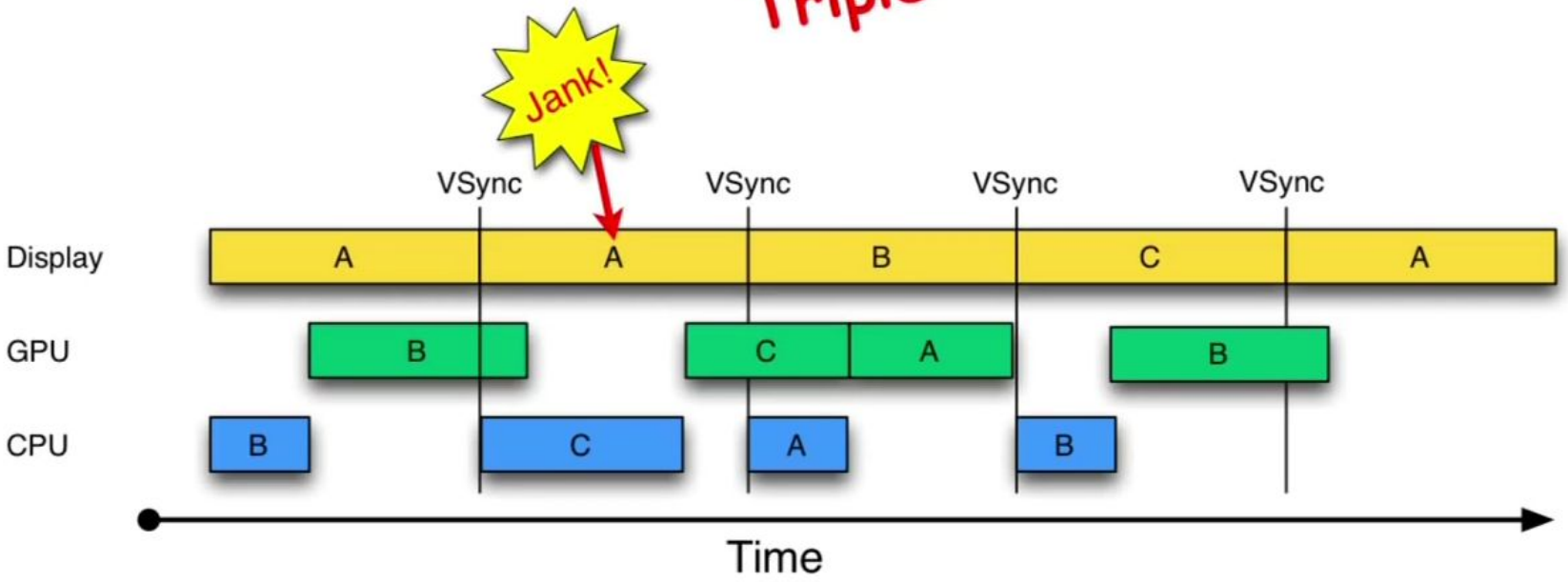
http://source.android.com/devices/graphics/architecture.html

Parallel Processing and Double Buffering



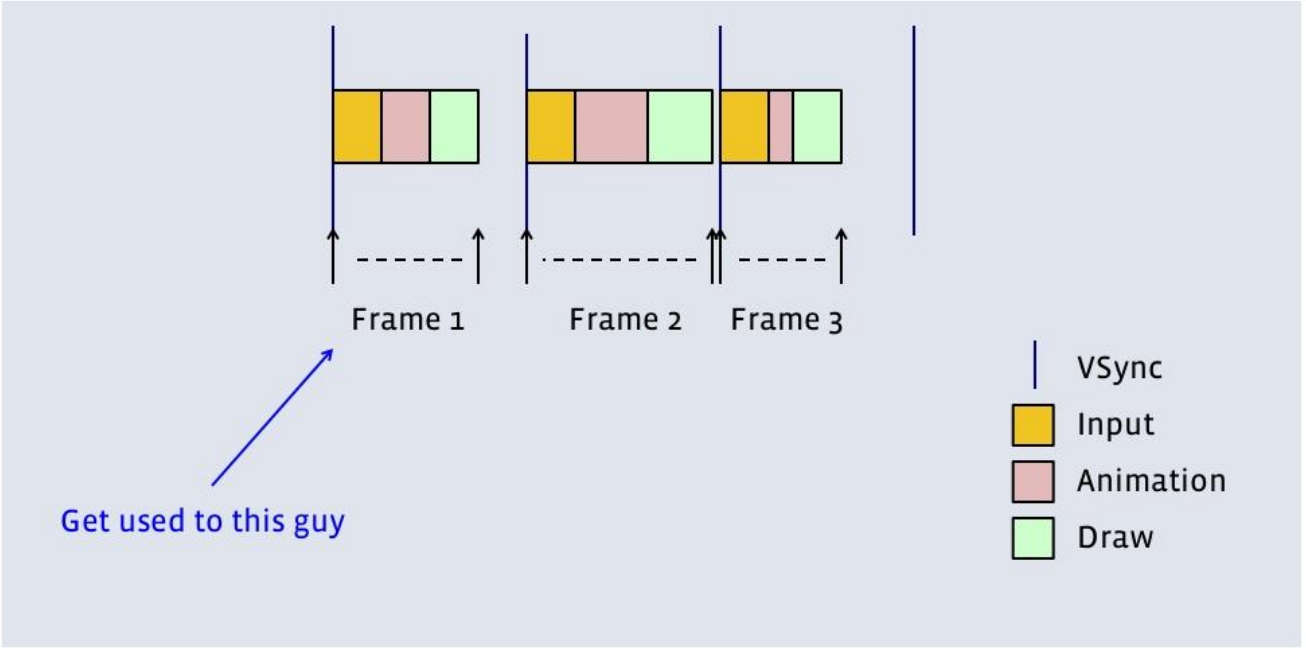
“For Butter or Worse”

Parallel Processing and ~~Double~~ Triple Buffering



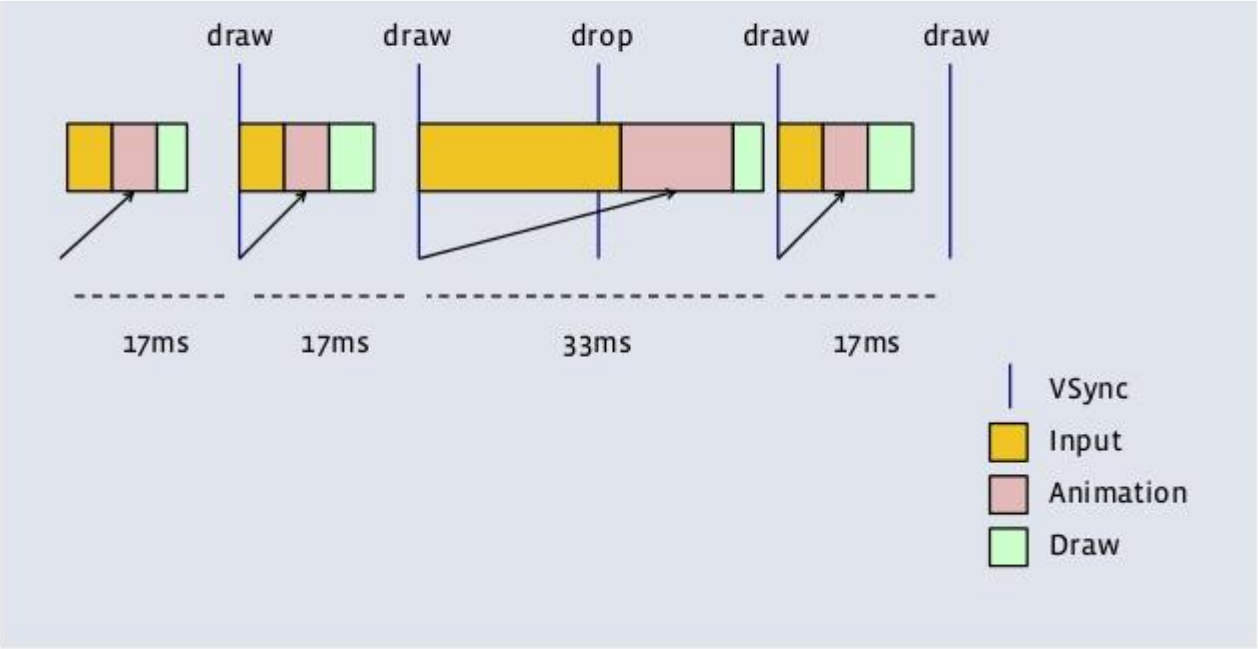
“For Butter or Worse”

So what’s happening on the UI thread?



Sendros

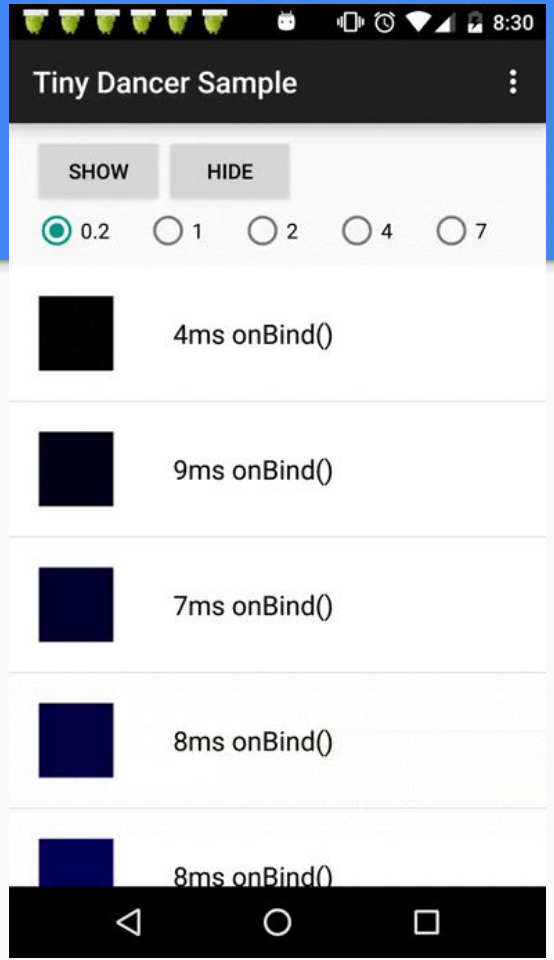
This is what lag looks like



Sendros

Check Out This App

- https://goo.gl/MAzK7i
- Live-Updating FPS Display
- See how longer times to generate ListView items affects frame rate



Case Study:
Facebook News Feed Performance

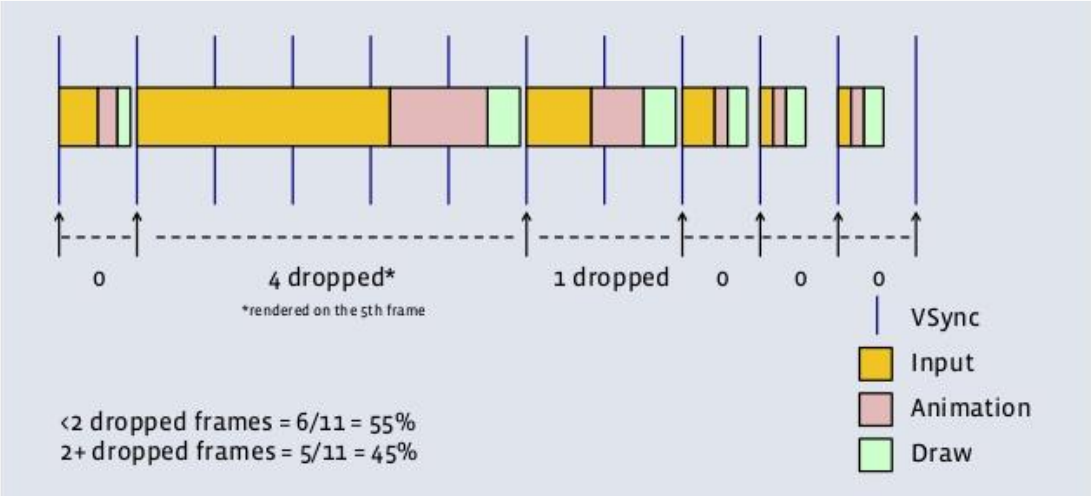
How do we know what’s wrong?

- Use Choreographer callbacks to determine how long each frame takes
  - Use FrameCallback interface and compute the time between callbacks to get fps

```
public interface FrameCallback {
    /**
     * @param frameTimeNanos The time in nanoseconds when the frame started being rendered
     */
    void doFrame(long frameTimeNanos);
}
```

Facebook’s Metric

- Percent time spent dropping 2+ frames



Sendros

Automagically Blame Jank

- Categories of events (GC, get new view, notify dataset changed)
  - When an event happens, drop a “marker”
  - When the app notices frames dropped, find the nearest marker and attribute blame
- Can account for 60% of slow frame time

Year Class	Slow time	GC	getView	notifyDSC
2012	16%	6%	50%	16%
2014	22%	12%	44%	12%

Sendros

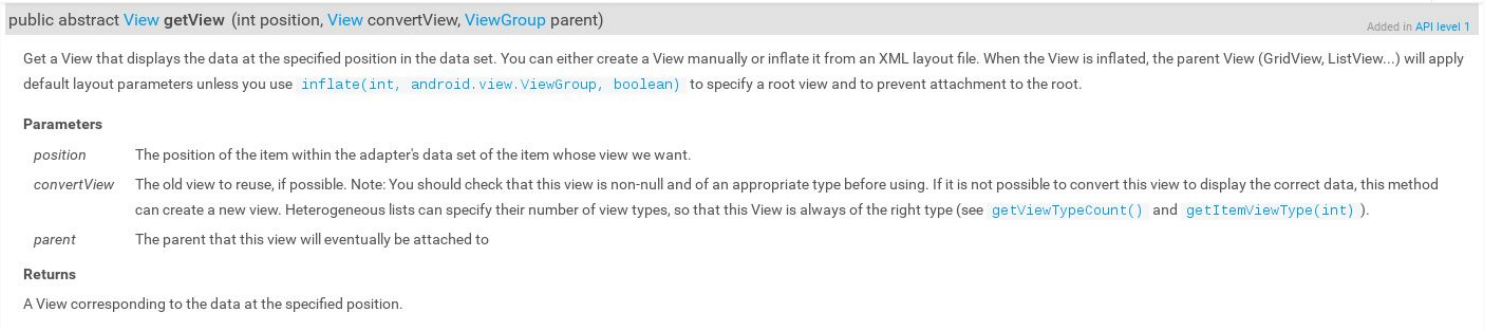
Automatically Profile CI Builds

- Run a test on CI infrastructure that scrolls through Newsfeed
  - Use Traceview sample-based profiling of frame data
  - Find where the weak points are and dig to the root of the problem (by method!)

Method	Time
com/android/widget/RecyclerView.onScrolled	16%
› com/facebook/logging/FeedLogger.onScroll	10%
› com/facebook/feed/NewsFeedFragment.maybeFetch	3.3%
›› com/facebook/feed/NewsFeedRecyclerView.notifyDataSetChange	3%
›› com/facebook/feed/FetchController.shouldFetch	.3%
› com/facebook/prefetch/ImagePrefetcher.onScroll	2.7%

What is Newsfeed

- A *really* complicated ListView
  - When a new becomes visible, ListView calls adapter's *getView* method
    - Needs to inflate (or convert) and populate a view in <<16.7 ms
- Each story looks different, has different sub-components
  - Headers, footers, content (images, videos), shared stories, aggregate stories, etc.



public abstract View **getView** (int position, View convertView, ViewGroup parent)

Added in API level 1

Get a View that displays the data at the specified position in the data set. You can either create a View manually or inflate it from an XML layout file. When the View is inflated, the parent View (GridView, ListView...) will apply default layout parameters unless you use `inflate(int, android.view.ViewGroup, boolean)` to specify a root view and to prevent attachment to the root.

Parameters

position	The position of the item within the adapter's data set of the item whose view we want.
convertView	The old view to reuse, if possible. Note: You should check that this view is non-null and of an appropriate type before using. If it is not possible to convert this view to display the correct data, this method can create a new view. Heterogeneous lists can specify their number of view types, so that this View is always of the right type (see `getViewTypeCount()` and `getItemViewType(int)`).
parent	The parent that this view will eventually be attached to

Returns

A View corresponding to the data at the specified position.

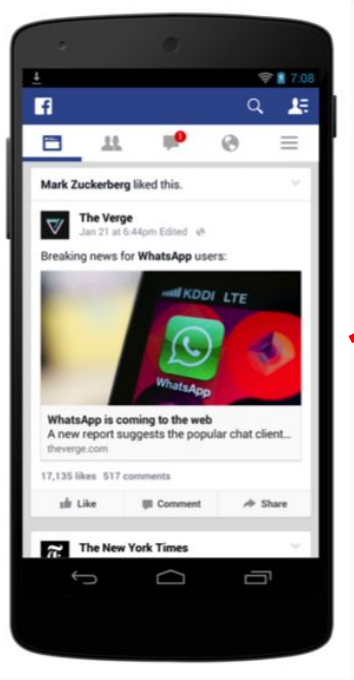
The Old Way

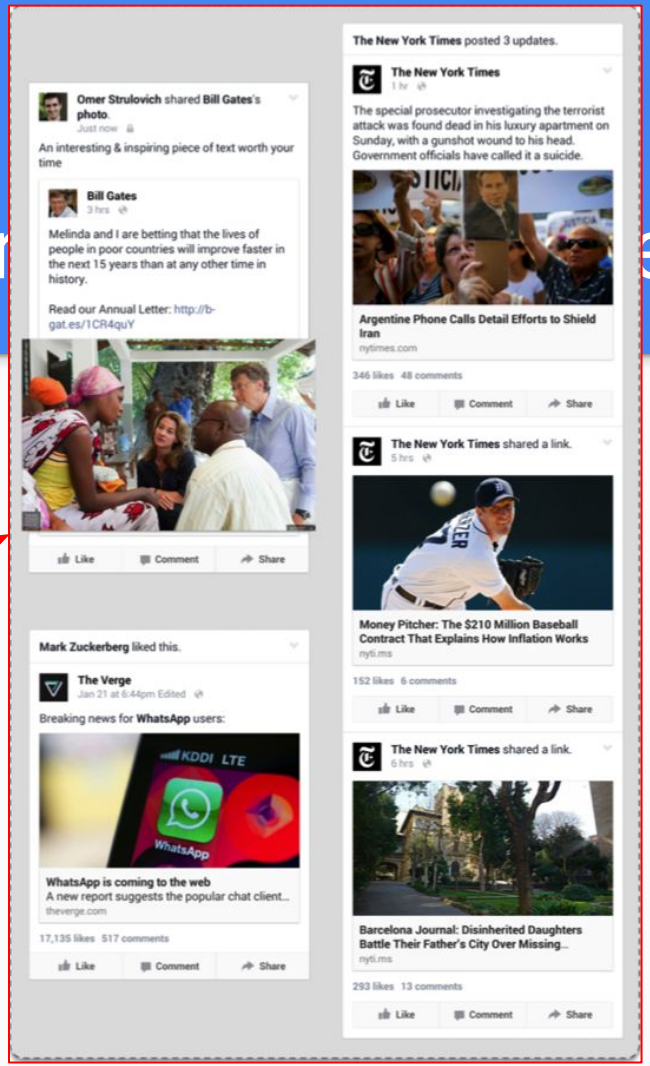
- Lots of custom view classes
  - “Bound” data models so that they could be rendered
- Did not work well with View recycling
  - Two instances of *StoryView* would be very different from each other
- Complex view hierarchy
- Computationally intense
- Had to keep lots in memory

```
public class NewsFeedStoryView extends LinearLayout {

  // ...

  public void bindModel(NewsFeedStory story) {
    // ...
    mHeaderView.bindModel(story);
    mMessageView.bindModel(story)
    mAttachmentsView.bindModel(story);
    // ... and so on ...
  }

  // ...
}
```

Strulovich

How many stories ar e?





Strulovich

MultiRow

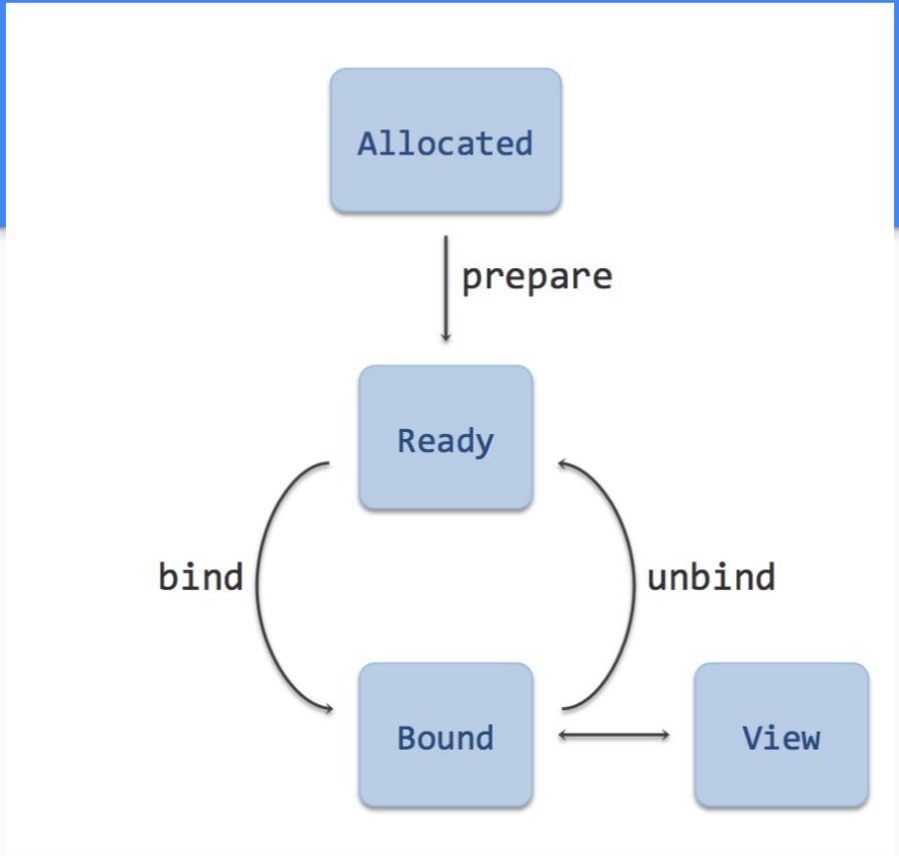
- Take a new approach
  - Each story in Feed consists of multiple items in the ListView
- This makes Android's built-in recycling effective
  - Recycling can happen on a sub-story level
- Fewer views need to be stored in memory (see last example)
  - Also reduces binding time (only have to process story parts)
  - Splits full story binding times over multiple frames (when certain parts of story enter view)
- Decoupled design: split custom view into two classes
  - A simpler custom view and a "binder" class that handles business logic

Look Better?

Strulovich

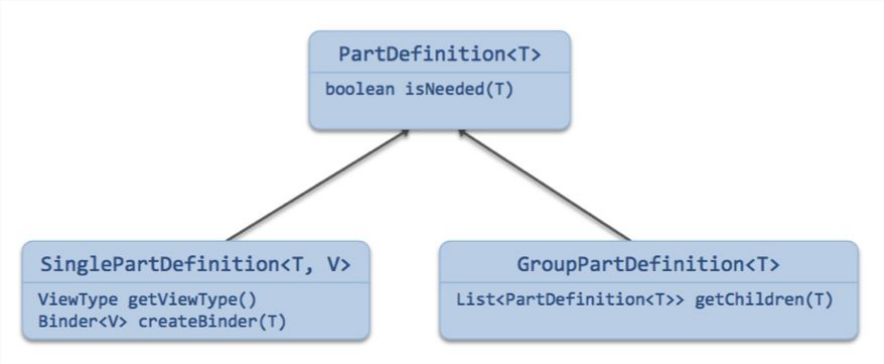
The lifecycle of a story

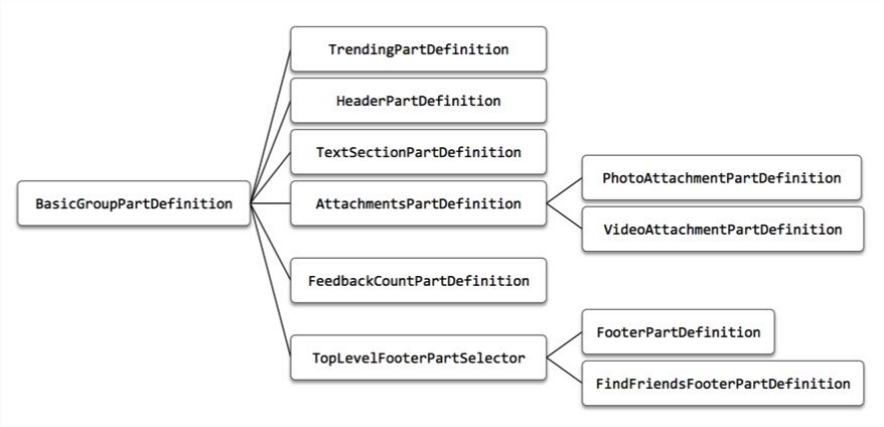
- Custom view has only “setters”
  - *setTitle*, *setProfilePic*, etc.
  - Decoupled from specific story
- Binder class replaces *bindModel*
  - *prepare*, *bind*, and *unbind* steps



Which parts are needed?

- Build hierarchy of story parts - start from the root and expand





Why is this better?

- Reduced Out Of Memory errors by 17%
  - Total errors reduced by 8%
  - Stack Overflows in view hierarchy have disappeared
- Reduced max time to render a frame by 10%
  - Removed lag from loading a big, complex story
- Increase code reuse between different layouts and stories
- Improved code quality
  - Increased test coverage
  - 70% MultiRow line coverage (as compared to 17% before)

Questions?

Sources

- Chet Haase & Romain Guy, “For Butter or Worse” @ Google I/O 2012
  - Video: https://www.youtube.com/watch?v=Q8m9sHdyXnE
- Mathias Garbe, Android Graphics Pipeline: From Button to Framebuffer
  - Part 1: https://blog.inovex.de/android-graphics-pipeline-from-button-to-framebuffer-part-1/
  - Part 2: https://blog.inovex.de/android-graphics-pipeline-from-button-to-framebuffer-part-2/
- Jason Sendros, “The Road to 60 fps” @ Droidcon NYC 2015
  - Slides: www.slideshare.net/JasonSendros/the-road-to-60-fps
  - Video: https://www.youtube.com/watch?v=RFzhXbZu_N8
- Omer Strulovich, “Facebook Performance: Designing News Feed Rendering” @ Droidcon NYC 2014
  - Video: https://www.youtube.com/watch?v=CPbzxK_s41s
  - Blog Post: https://code.facebook.com/posts/879498888759525/fast-rendering-news-feed-on-android/
- Android docs:
  - http://source.android.com/devices/graphics/architecture.html
  - http://developer.android.com/guide/components/processes-and-threads.html#Threads
  - http://developer.android.com/reference/android/widget/Adapter.html
  - http://developer.android.com/guide/topics/ui/how-android-draws.html