@manuelvicnt

ConstraintLayout

Manuel Vicente Vivo

Introduction

ConstraintLayout

- New layout supported by Google
- A layout is a ViewGroup that is responsible for positioning its child Views. It calculates and set the position and size of those Views
- Allows you to create large and complex layouts with a flat view hierarchy (no nested view groups)

How Android Draws Views

- Measure – Layout – Draw cycle
  - Measure: Determines the view size
  - Layout: Sets position and size for the view and all its descendants
  - Draw: Draws the view on the screen

Why ConstraintLayout?

Why?

- Designed to
  - create efficient UI
  - be a top-level layout with scalability in mind
- Flattens your UI hierarchy (key for animations)
- Solution to address the performance problems of other layouts
  - Nested weighted Linear Layouts
  - Relative Layouts

Differences

- AbsoluteLayout?
  - It fixes widgets in size
- CoordinatorLayout?
  - It’s made to coordinate behaviours between views. Not a proper layout
- PercentageLayout?
  - It just solves the problem with nested, weighted LinearLayouts

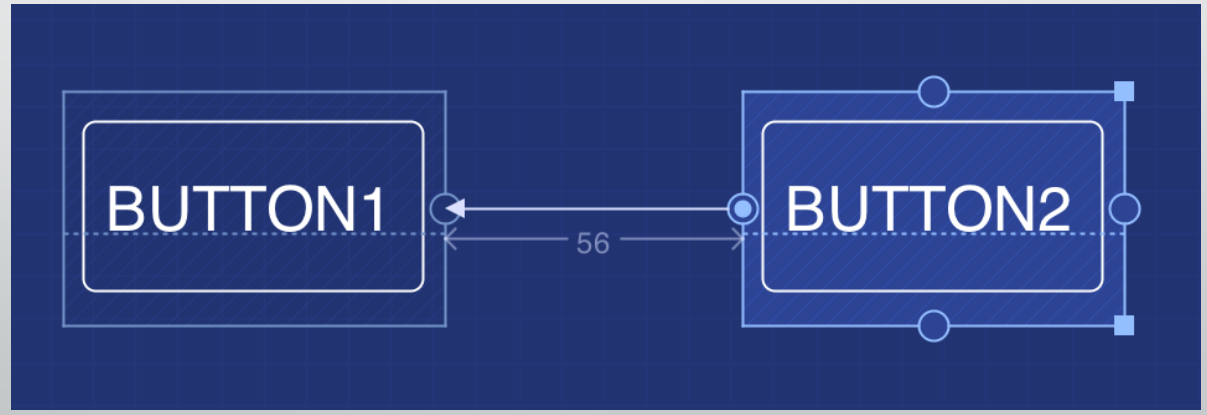
Overview

Some Facts

- It was designed with the visual editor in mind
  - To be used with it instead of hand-crafting XML
- Pretty lightweight library – 80KB
- Flat hierarchies display quick and are memory efficient

Constraints

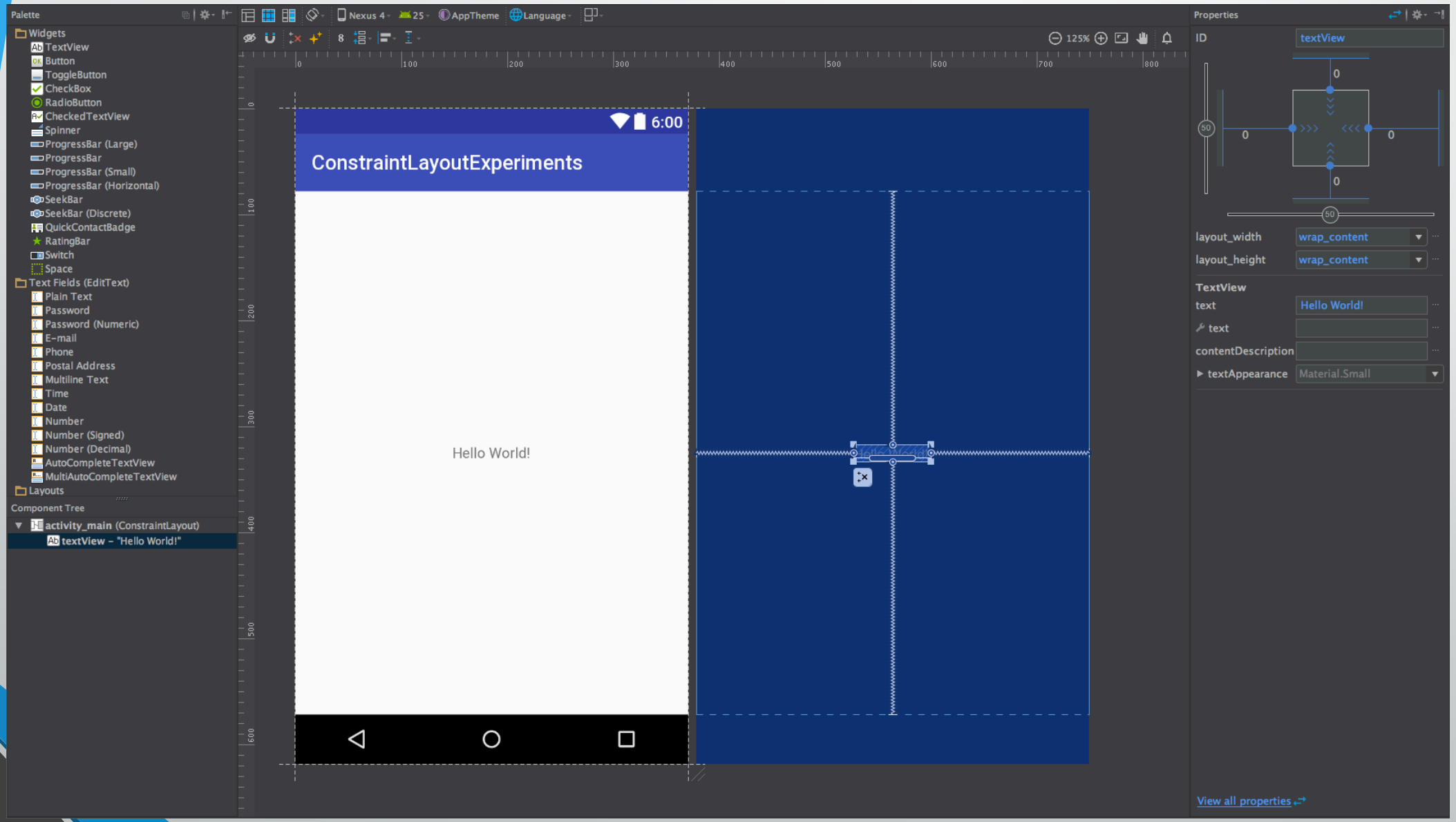
- Keep widgets aligned
- Uses anchors to determine alignment rules between various widgets



Type of Handles

- Resize handle. Resizes the widget 
- Side Constraint Handle (Anchors). Let you specify the location of the widget creating constraints 
- Baseline Constraint Handle. Helps you align the text fields of any two widgets (irrespective of the widget sizes)

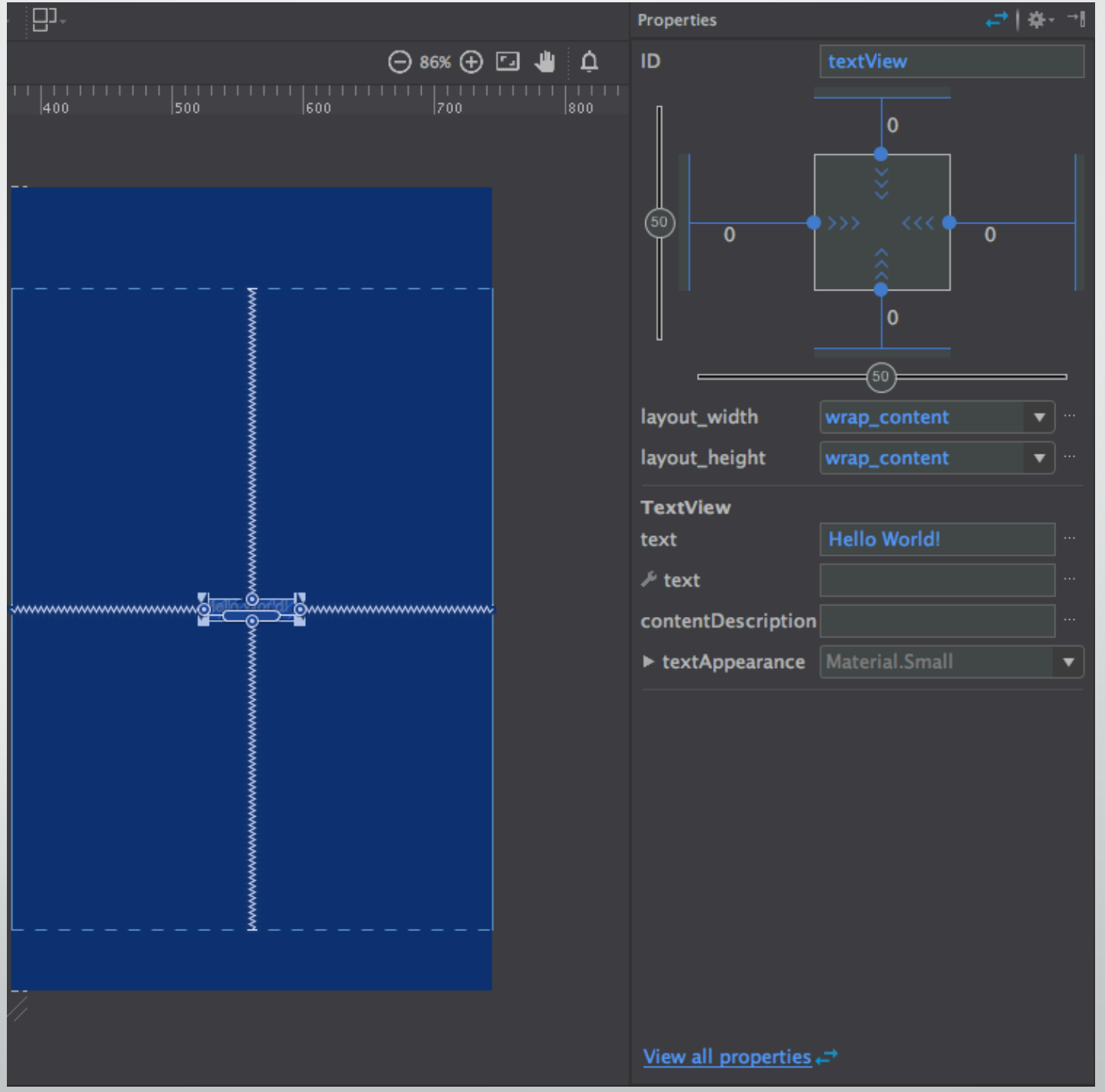
New layout editor



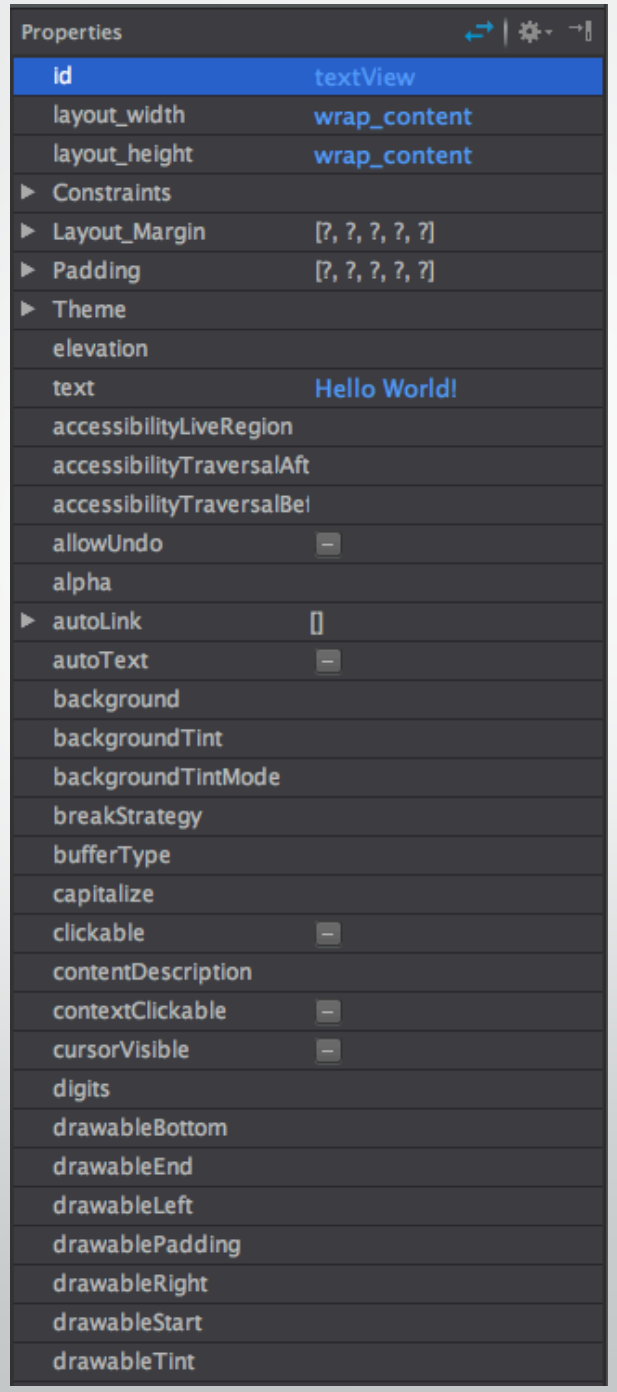
Tools

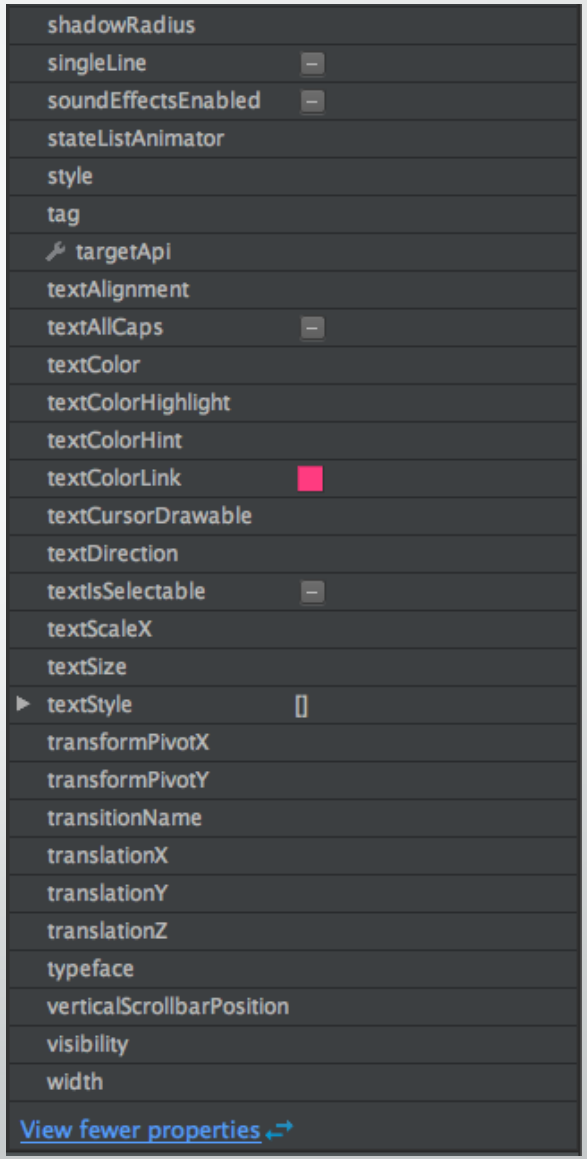
- Show/Hide constraints
- Autoconnect. Creates default constraints based upon where we dropped the widget. Watch this in action
- Clear constraints
- Launch Inference engine. Automatically infer constraints that are not set
- Default margin when dropping a widget

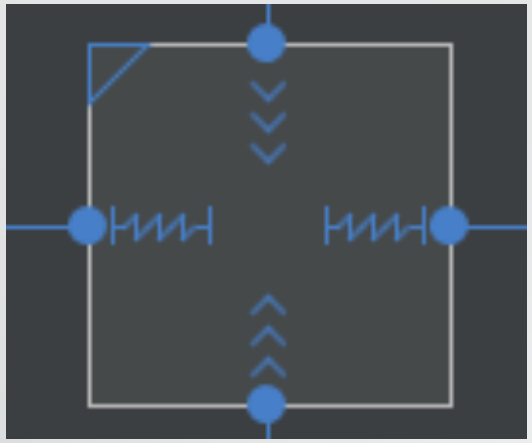
Properties View



Properties Panel







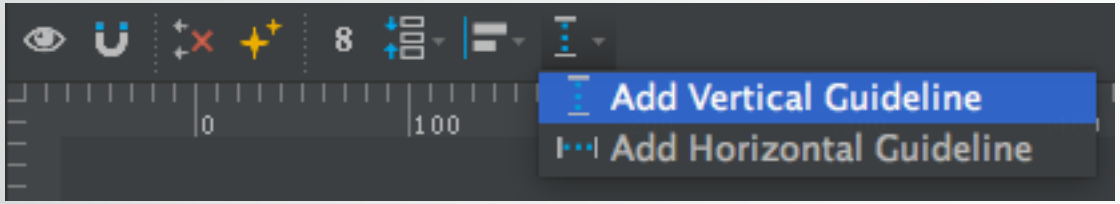
Dimensions

- Fixed. Specify the width/height of the widget

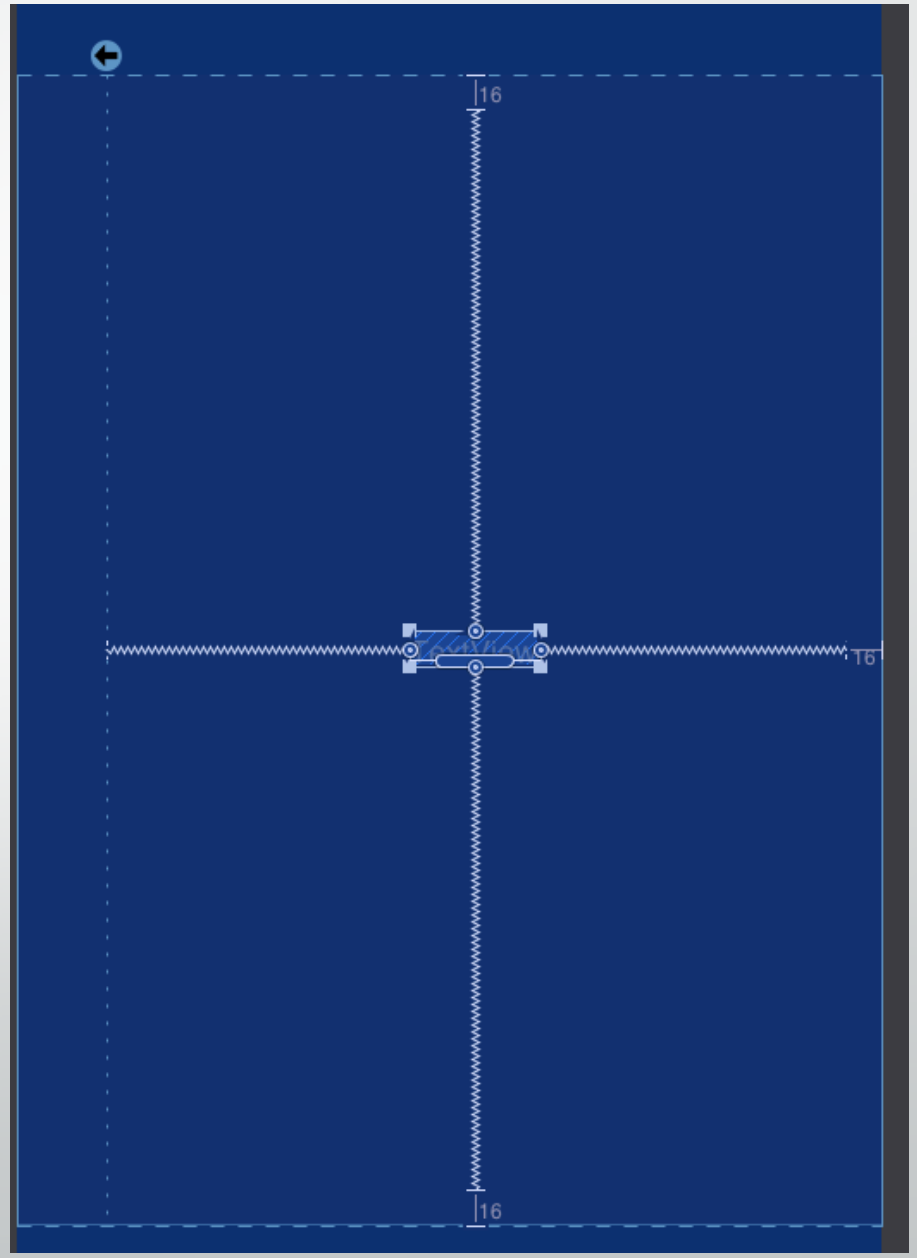
- AnySize. Occupy all available space to satisfy the constraint

- Wrap Content. Size is big enough to enclose its content

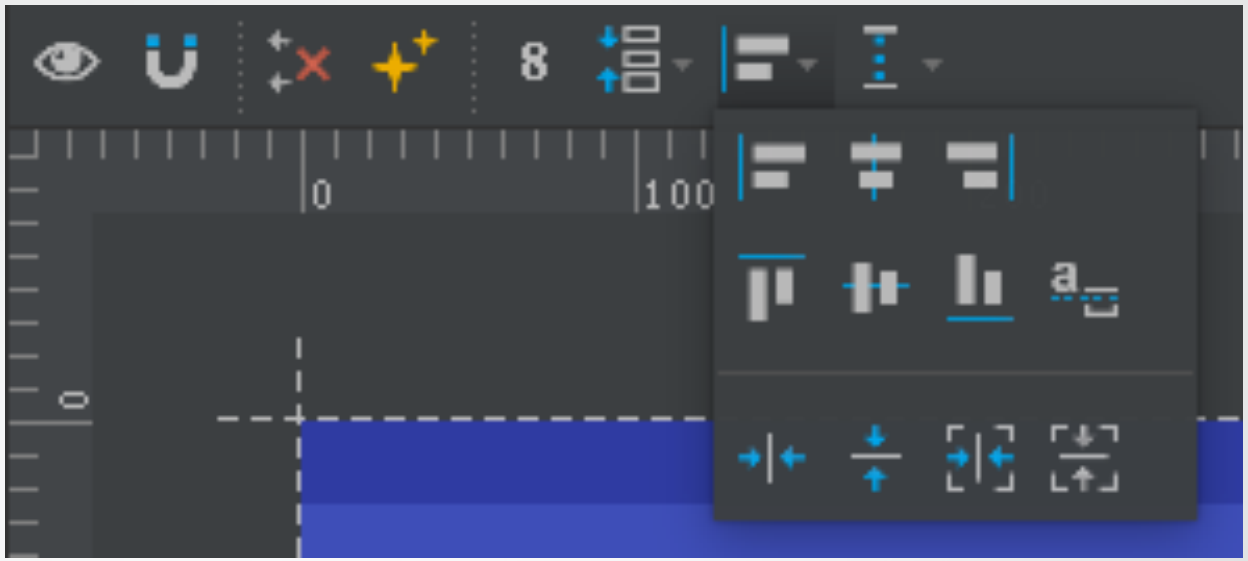
Guidelines



- Lines that can be set on the canvas at edit-time.
- Can be used to align/constraint widgets against them.

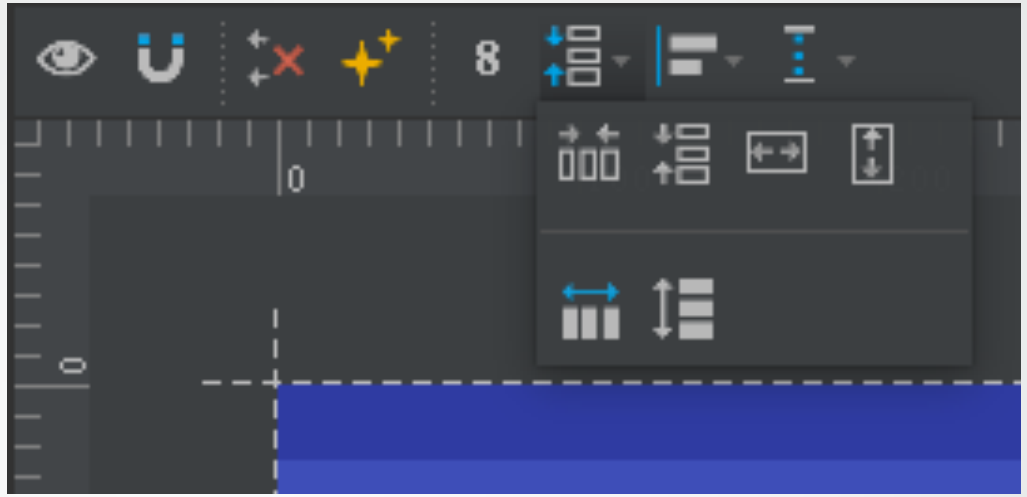


Alignment tools



- First line: Align all selected widgets to the left right or center of each other
- Second line: same as first one but vertically
- Third line: centring items.
  - First two: centre within the remaining space
  - Second two: centre within the parent layout

Pack Tools



- First pair are to pack the items within the available space either horizontally or vertically
- Second pair of items will expand the selected items to fill the parent (it harcodes dp values in the attributes ☹)
- The final pair distribute items horizontally or vertically (uses constraints ☺)

Chains

- Group behaviour on a single axis.

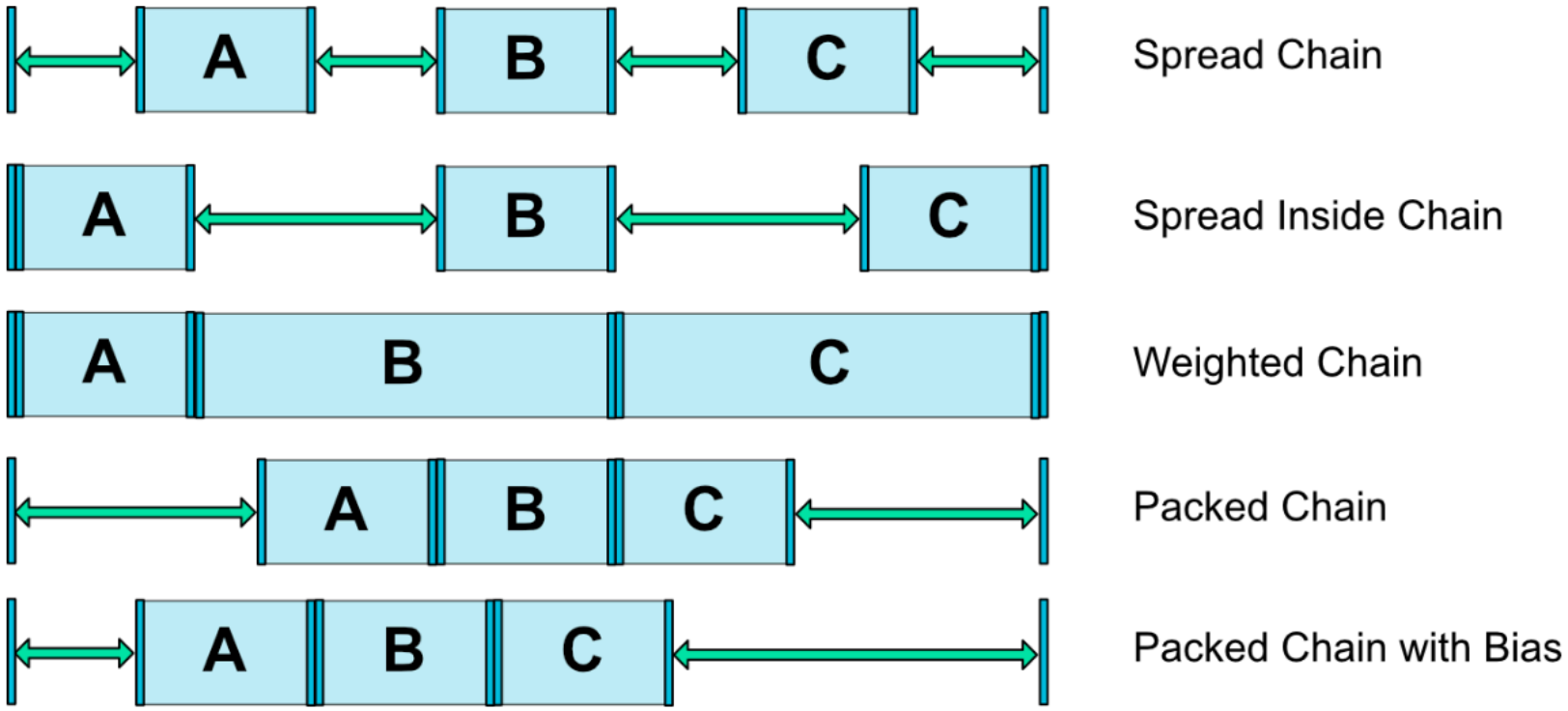


Fig. 10 - Chains Styles

Critics

- It’s going to be difficult to code review because developers will not be as familiar with the XML format
- Not production ready yet (beta version). They still have to solve some performance issues

Get Started

- Android Studio 2.3

```
dependencies {

    ...
    compile 'com.android.support.constraint:constraint-layout:1.0.0-beta4'
}
```

```
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

</android.support.constraint.ConstraintLayout
```

Hands on!

Future

- Guidelines will be inferred
- Import an image and it’ll get everything out of it

@manuelvicnt

Q&A

Thanks for Listening