Android Custom Controls and Canvas

Xavier Rubio Jansana

@teknik_tdr
https://xrubio.com
https://github.com/xrubioj/

What?

- Controls that don't exist in Android
- Compound Controls ("Group of `Views`")
- Customization of controls

When?

- No similar control exists
- Same groups of controls repeats
- Theming is not enough

Benefits

- Encapsulate and simplify common controls
- Create a design language
- Simplify maintenance

Compound Controls

1. Extend a `ViewGroup` (e.g. `LinearLayout`, `ConstraintLayout`...)
2. Inflate the layout and attach it
3. ...
4. Profit!

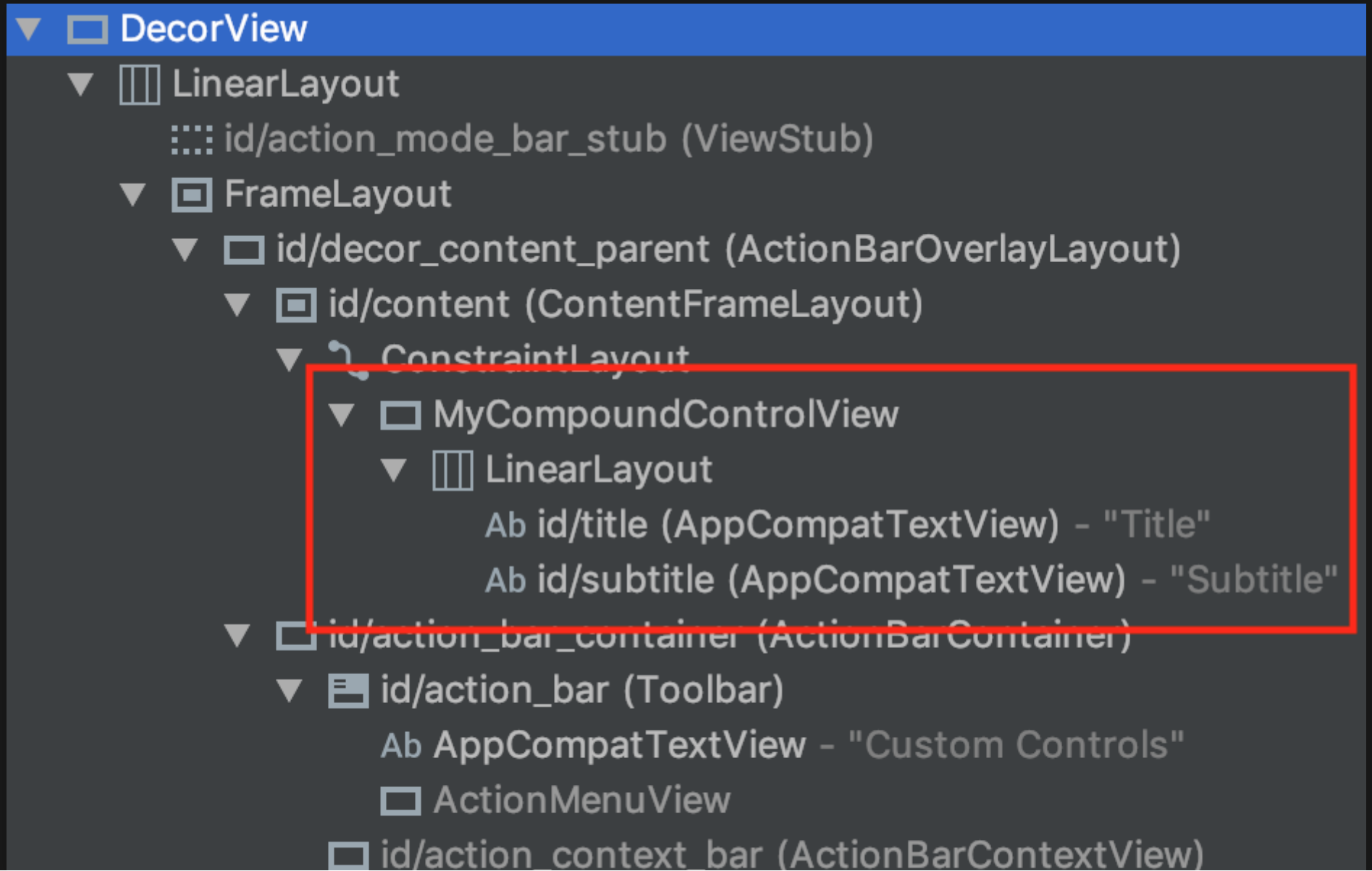
Compound Controls - Class

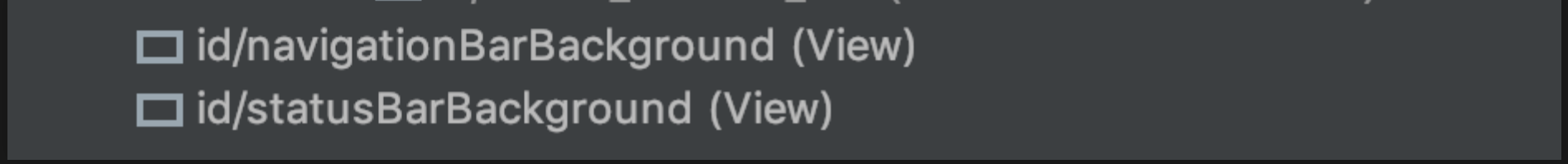
```
class MyCompoundControlView @JvmOverloads constructor(
    context: Context?,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0,
    defStyleRes: Int = 0
) : LinearLayout(context, attrs, defStyleAttr, defStyleRes) {

    init {
        View.inflate(context, R.layout.my_compount_control_layout,
                                    this)
    }
}
```

Compound Controls - Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/title"
        android:textAppearance="@style/TextAppearance.AppCompat.Title"
        android:text="Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/subtitle"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:text="Subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Compound Controls - Result




id/navigationBarBackground (View)
id/statusBarBackground (View)

Compound Controls - Layout (better)

```
<?xml version="1.0" encoding="utf-8"?>
<merge
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tool="http://schemas.android.com/tools"
    tool:parentTag="android.widget.LinearLayout"
    tool:orientation="vertical"
    tool:layout_width="wrap_content"
    tool:layout_height="wrap_content">
    <TextView .../>
    <TextView .../>
</merge>
```

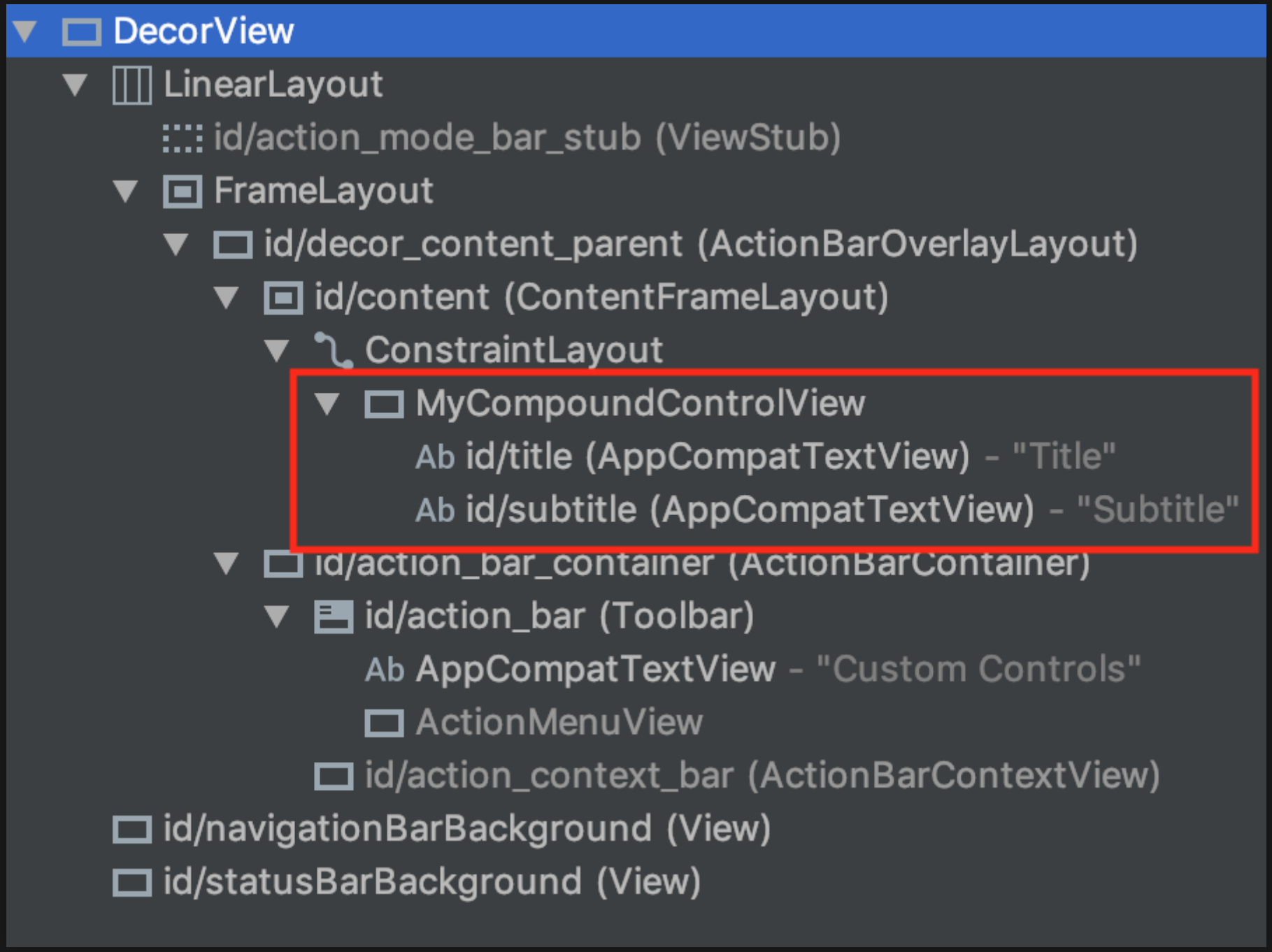
Notice `<merge>` and `tool`

Compound Controls - Updated class

```
class MyCompoundControlView @JvmOverloads constructor(
    context: Context?,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0,
    defStyleRes: Int = 0
) : LinearLayout(context, attrs, defStyleAttr, defStyleRes) {

    init {
        View.inflate(context, R.layout.my_compount_control_layout,
                     this)
        this.orientation = VERTICAL
    }
}
```

Initialization of root tag moved here

Compound Controls - Result


DecorView
LinearLayout
id/action_mode_bar_stub (ViewStub)
FrameLayout
id/decor_content_parent (ActionBarOverlayLayout)
id/content (ContentFrameLayout)
ConstraintLayout
MyCompoundControlView
Ab id/title (AppCompatTextView) - "Title"
Ab id/subtitle (AppCompatTextView) - "Subtitle"
id/action_bar_container (ActionBarContainer)
id/action_bar (Toolbar)
Ab AppCompatTextView - "Custom Controls"
ActionMenuView
id/action_context_bar (ActionBarContextView)
id/navigationBarBackground (View)
id/statusBarBackground (View)

Compound Controls - Using

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sch
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.xrubio.customcontrols.MyCompoundControlView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Notice we're using the fully qualified class name

UI Drawing Steps

1. Measure: calculate dimensions based in constraints ✔
2. Layout: layout children → we don't need it
3. Draw: use `Canvas` to draw ✔

Measure

```
val specMode: Int = MeasureSpec.getMode(measureSpec)
val specSize: Int = MeasureSpec.getSize(measureSpec)
```

We have a measureSpec per axis (X & Y)

- UNSPECIFIED
- EXACTLY
- AT_MOST

Measure

```
override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpe
    val minW: Int = (paddingStart + paddingEnd + _radius * 2.0
    val w: Int = resolveSizeAndState(minW, widthMeasureSpec, 0

    val minH: Int = (paddingTop + paddingBottom + _radius * 2.
    val h: Int = resolveSizeAndState(minH, heightMeasureSpec,

    setMeasuredDimension(w, h)
}
```

Measure

- Units are **pixels**
- **Must** call `setMeasuredDimension()`
- Use `resolveSizeAndState(size, measureSpec, childMeasuredState)` as helper

Canvas

- Units are **pixels**
- Top-left is (0, 0)
- Draws in order ("on top of")

Canvas vs. Paint

- Canvas is where things are drawn
- Paint is how things are drawn
- Canvas can get a backing Bitmap

Canvas commands

- drawBitmap()
- drawRect(), drawCircle(), etc.
- drawColor()
- drawText()
- clip*()

Properties

After changind a property, we need to either invalidate or relayout:

- `invalidate()`: triggers a redraw. Use when property doesn't changes size.
- `requestLayout()`: triggers the whole cycle (measure, layout, draw). Use when property changes size.

Canvas and KTX

```
implementation 'androidx.core:core-ktx:1.1.0'
```

- Helps cleanup code (avoiding save() and restore(), for instance)
- `withClip()`, `withMatrix()`, `withRotation`, etc.
- See: https://developer.android.com/reference/kotlin/androidx/core/graphics/package-summary

References

- Custom View Components at developer.android.com https://developer.android.com/guide/topics/ui/custom-components
- "Tips for Building Custom Views on Android with Canvas APIs" slides, Rebecca Franks https://speakerdeck.com/riggaroo/tips-for-building-custom-views-on-android-with-canvas-apis
- Android Canvas APIs with Kotlin and KTX https://riggaroo.co.za/android-canvas-apis-with-ko-and-ktx/

Other References

- Source code of `View#resolveSizeAndState(int, int, int)` in AOSP https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/java/android/view/View.java#24704
- What's the utility of the third argument of View.resolveSizeAndState()? https://stackoverflow.com/questions/13650903/whats-the-utility-of-the-third-argument-of-view-resolvesizeandstate/<13651513#13651513

Questions? 

Thanks! 

Xavier Rubio Jansana

@teknik_tdr
https://xrubio.com
https://github.com/xrubioj/

This talk is available at:
https://xrubio.com/talks/talk-android-custom-controls-and-canvas/