MotionLayout: Animation made easy

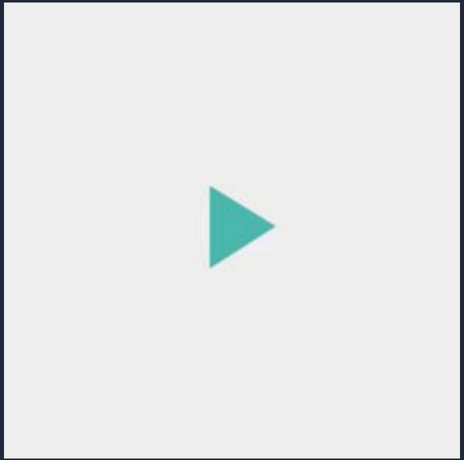
Luiz Santana
Mobile Chapter Lead @ mytaxi

Some of the current options

- Animated Vector Drawable
- Property Animation framework
- Layout Transitions
- Layout Transitions with Transition Manager
- Lottie (from airbnb)

Animated Vector Drawable

- Small user interactions
- Predefined animations

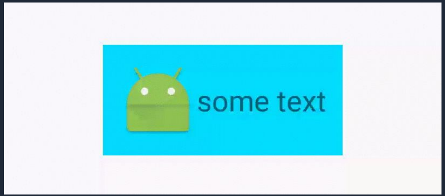


Property Animation

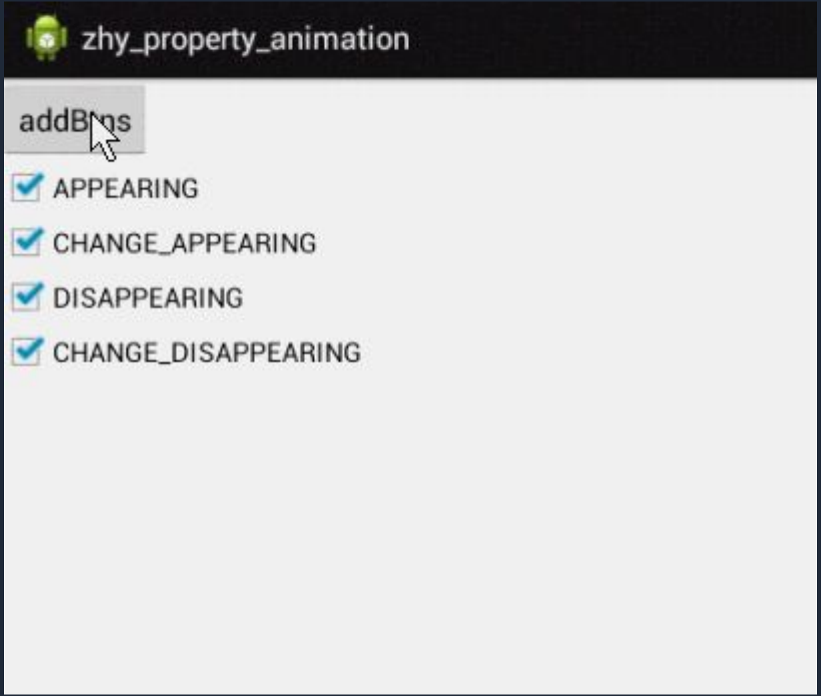
- Animate view property (alpha, translate, etc)
- Create animation set (combine animations)
- Add animations listeners
- Apply properties in the end (optional)

```
// Translate Element
ObjectAnimator.ofFloat(textView, "translationX", 100f).apply {
   duration = 1000
   start()
}

// Animator set
val bouncer = AnimatorSet().apply {
   play(bounceAnim).before(squashAnim1)
   play(squashAnim1).with(squashAnim2)
   play(squashAnim1).with(stretchAnim1)
   play(squashAnim1).with(stretchAnim2)
   play(bounceBackAnim).after(stretchAnim2)
}
val fade= ObjectAnimator.ofFloat(newBall, "alpha", 1f, 0f).apply {
   duration = 250
}

AnimatorSet().apply {
   play(bouncer).before(fade)
   start()
}
```

Layout Transitions



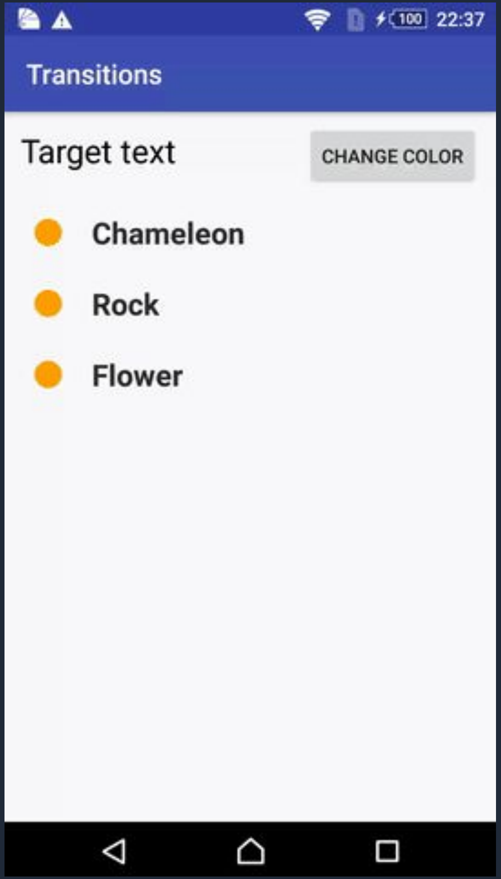
- Easy transitions on adding view
- Easy transitions on removing view
- Nice / simple effect for updating sizes



- [✓] APPEARING
- [✓] CHANGE_APPEARING
- [✓] DISAPPEARING
- [✓] CHANGE_DISAPPEARING

Layout Transitions with Transition Manager

- Able to create scenes in transitions between different layouts
- Shared elements between transitions
- Example: Change view bounds or Shared element transitions



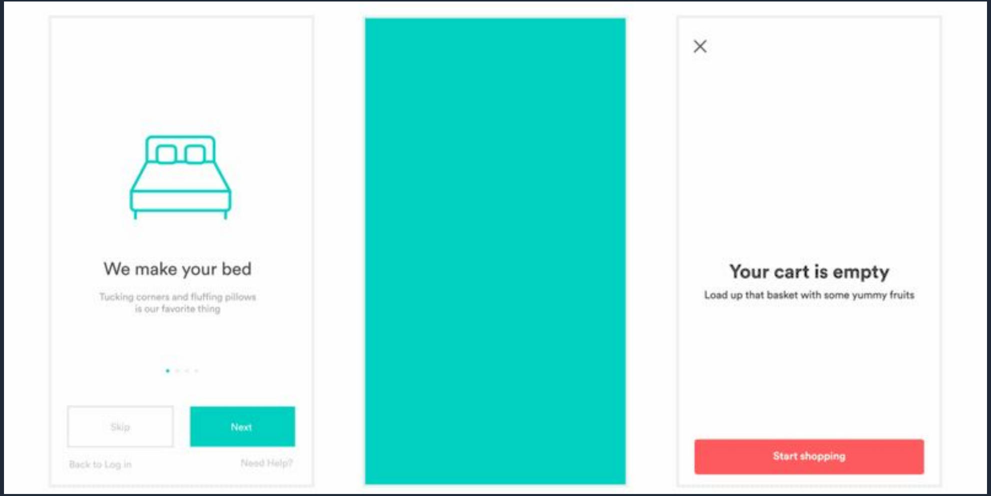
Lottie

PROS

- Animations straight from After Effects
- Easy for designers to help with it
- Nice scene animations

CONS

- Not focused on layout meaningful motions
- Not able to proper handle user interactions (swipe and click)
- Limited resources from After Effects, which make some back-and-forth between Designers and Developers

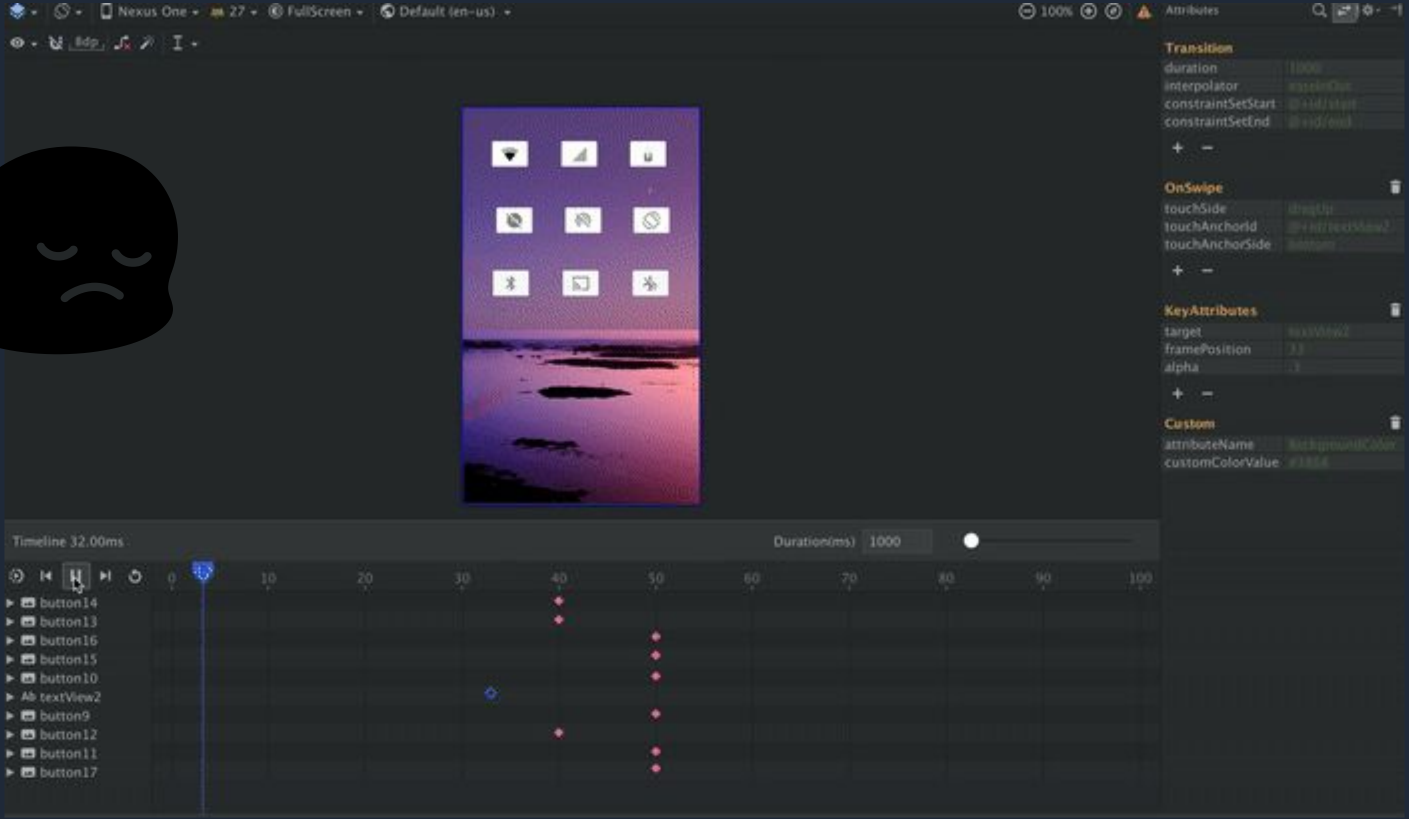

We make your bed
Tucking corners and fluffing pillows is our favorite thing
Skip
Next
Back to Log in
Need Help?
Your cart is empty
Load up that basket with some yummy fruits
Start shopping

So why MotionLayout?

So why MotionLayout?

- Mix of:
    - Transition like transition Manager (between 2 layouts)
    - Animate properties in the layout (any property)
    - Coordinator Layout
- Declarative
- Tooling

Tooling?



Transition

duration	1000
interpolator	easeInOut
constraintSetStart	@+id/start
constraintSetEnd	@+id/end

OnSwipe

touchSide	dragUp
touchAnchorId	@+id/textView2
touchAnchorSide	bottom

KeyAttributes

target	textView2
framePosition	33
alpha	.3

Custom

attributeName	backgroundColor
customColorValue	#1864

Timeline 32.00ms

Duration(ms) 1000

button14
button13
button16
button15
button10
textView2
button9
button12
button11
button17

First steps with MotionLayout

First steps with MotionLayout

- Add gradle dependency

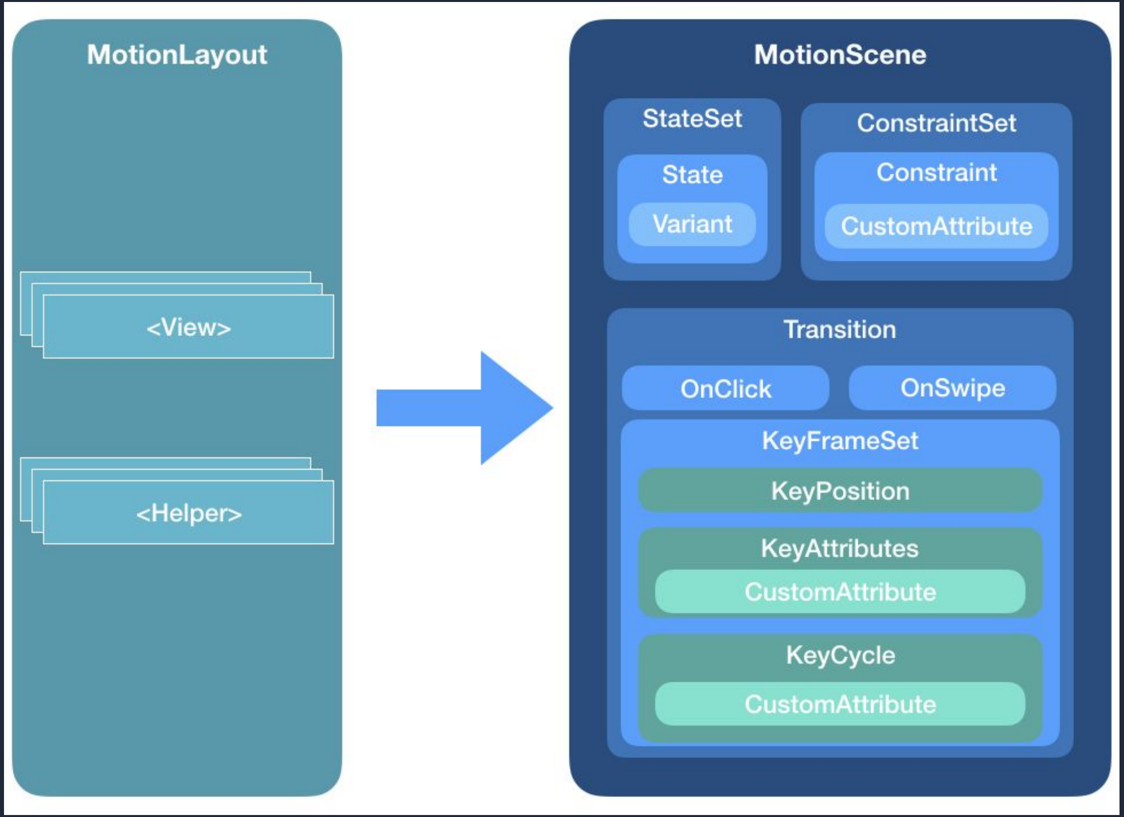
```
// build.gralde (app module)
implementation
'androidx.constraintlayout:constraintlayout:2.0.0-alpha2'
```

First steps with MotionLayout

- Add gradle dependency
- Update target layout

```
// build.gralde (app module)
implementation
'androidx.constraintlayout:constraintlayout:2.0.0-alpha2'

// layout file
// from
<android.support.constraint.ConstraintLayout .../>
// to
<android.support.constraint.motion.MotionLayout .../>
```

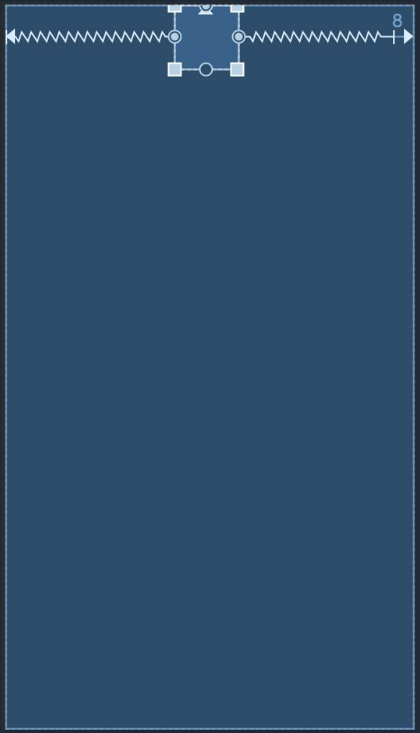
Understanding MotionLayout animations



First Sample:
Reference existing layout

Referencing existing layout

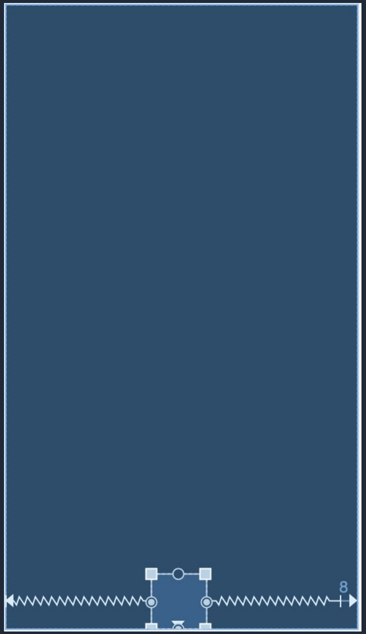
- Create layout for start position



```
// layout start XML
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">

  <View
     android:id="@+id/button"
     android:background="@color/colorAccent"
     android:layout_width="64dp"
     android:layout_height="64dp"
     android:layout_marginTop="8dp"
     android:text="Button"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toTopOf="parent" />

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

Referencing existing layout

- Create layout for start position
- Create layout for end position



```
// layout end XML
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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">

  <View
     android:id="@+id/button"
     android:background="@color/colorAccent"
     android:layout_width="64dp"
     android:layout_height="64dp"
     android:layout_marginBottom="8dp"
     android:text="Button"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintBottom_toBottomOf="parent" />

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

Referencing existing layout

- Create layout for start position
- Create layout for end position
- Create actual motion layout

```
// activity layout
<androidx.constraintlayout.motion.widget.MotionLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  app:layoutDescription="@xml/motion_scene_01"
  tools:showPaths="true">

  <View
     android:id="@+id/button"
     android:background="@color/colorAccent"
     android:layout_width="64dp"
     android:layout_height="64dp"
     android:text="Button" />

</androidx.constraintlayout.motion.widget.MotionLayout>
```

Referencing existing layout

- Create layout for start position
- Create layout for end position
- Create actual motion layout
- Create motion scene

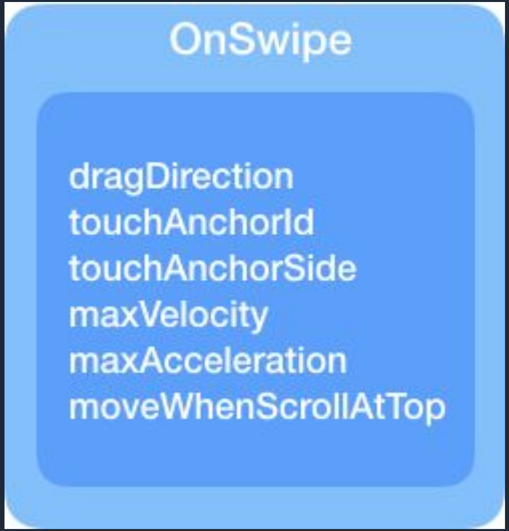
```
// motion scene xml
<MotionScene
  xmlns:motion="http://schemas.android.com/apk/res-auto">

  <Transition
     motion:constraintSetStart="@layout/activity_reference_start"
     motion:constraintSetEnd="@layout/activity_reference_end"
     motion:duration="1000">
     <OnSwipe
        motion:touchAnchorId="@+id/button"
        motion:touchAnchorSide="bottom"
        motion:dragDirection="dragDown" />
  </Transition>

</MotionScene>
```

Referencing existing layout

- Create layout for start position
- Create layout for end position
- Create actual motion layout
- Create motion scene
- RUN the app

OnSwipe handler

- dragDirection
  - Direction we are tracking (dragDown, dragLeft, …)
- touchAnchorId
  - Object id to be tracked
- touchAnchorSide
  - Side of object to be tracked
- moveWhenScrollAtTop
  - do scroll and transition happen at the same time?



Second Sample:
Self-contained scenes

Self-contained scenes

- Just create the main view with MotionLayout

```
// activity layout
<androidx.constraintlayout.motion.widget.MotionLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  app:layoutDescription="@xml/motion_scene_02"
  tools:showPaths="true">

  <View
     android:id="@+id/button"
     android:background="@color/colorAccent"
     android:layout_width="64dp"
     android:layout_height="64dp"
     android:text="Button" />

</androidx.constraintlayout.motion.widget.MotionLayout>
```

Self-contained scenes

- Just create the main view with MotionLayout
- Create the scene based on ContraintSet

```
// motions scene
<MotionScene ...>
  <Transition
     motion:constraintSetStart="@+id/start"
     motion:constraintSetEnd="@+id/end"
     motion:duration="1000">
     <OnSwipe
        motion:touchAnchorId="@+id/button"
        motion:touchAnchorSide="right"
        motion:dragDirection="dragRight" />
  </Transition>

  <ConstraintSet android:id="@+id/start">
     <Constraint
        android:id="@+id/button"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginStart="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintStart_toStartOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
  </ConstraintSet>

  <ConstraintSet android:id="@+id/end">
     <Constraint
        android:id="@+id/button"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:layout_marginEnd="8dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintEnd_toEndOf="parent"
        motion:layout_constraintTop_toTopOf="parent" />
  </ConstraintSet>
</MotionScene>
```

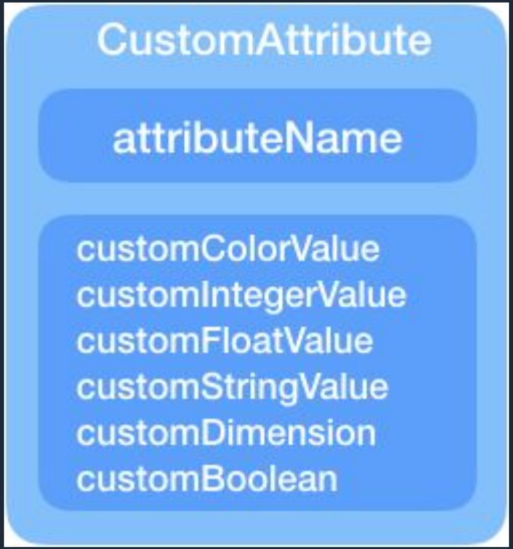
Self-contained scenes

- Just create the main view with MotionLayout
- Create the scene based on ContraintSet
- RUN the app!!

Third sample:
Custom attributes

Custom attributes

- Vary dynamically between different values
- E.g Float: 0.0 ~ 1.0



Custom attributes

- Create MotionLayout with ImageFilterView

```
// activity layout
<androidx.constraintlayout.motion.widget.MotionLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  app:layoutDescription="@xml/motion_scene_03"
  tools:showPaths="true">

  <androidx.constraintlayout.utils.widget.ImageFilterView
    android:id="@+id/image"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:src="@drawable/gdgberlin"/>

</androidx.constraintlayout.motion.widget.MotionLayout>
```

Custom attributes

- Create MotionLayout with ImageFilterView
- Add Motion Scene with Custom attributes

```
// motion scene
<MotionScene ...>
  <Transition  ...>
     <OnSwipe ... />
  </Transition>

  <ConstraintSet android:id="@+id/start">
     <Constraint
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        motion:layout_constraintStart_toStartOf="parent"
        motion:layout_constraintTop_toTopOf="parent">
        <CustomAttribute
           motion:attributeName="saturation"
           motion:customFloatValue="1" />
     </Constraint>
  </ConstraintSet>

  <ConstraintSet android:id="@+id/end">
     <Constraint
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        motion:layout_constraintBottom_toBottomOf="parent"
        motion:layout_constraintEnd_toEndOf="parent">
        <CustomAttribute
           motion:attributeName="saturation"
           motion:customFloatValue="0" />
     </Constraint>
  </ConstraintSet>
</MotionScene>
```

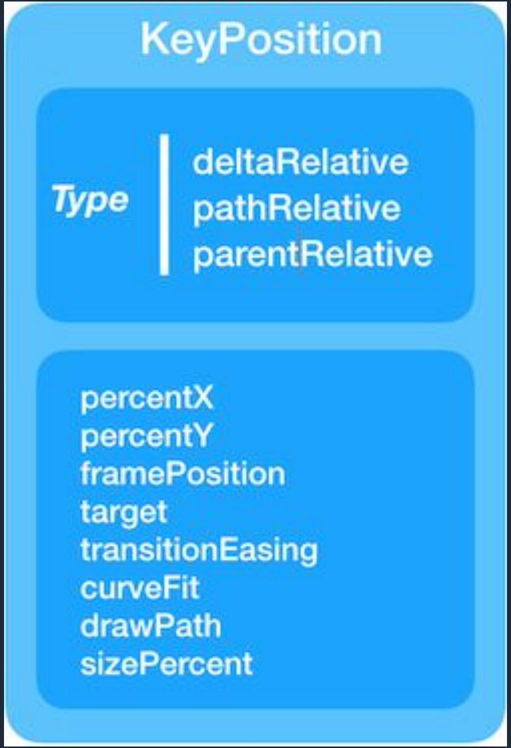
Custom attributes

- Create MotionLayout with ImageFilterView
- Add Motion Scene with Custom attributes
- RUN the app!!

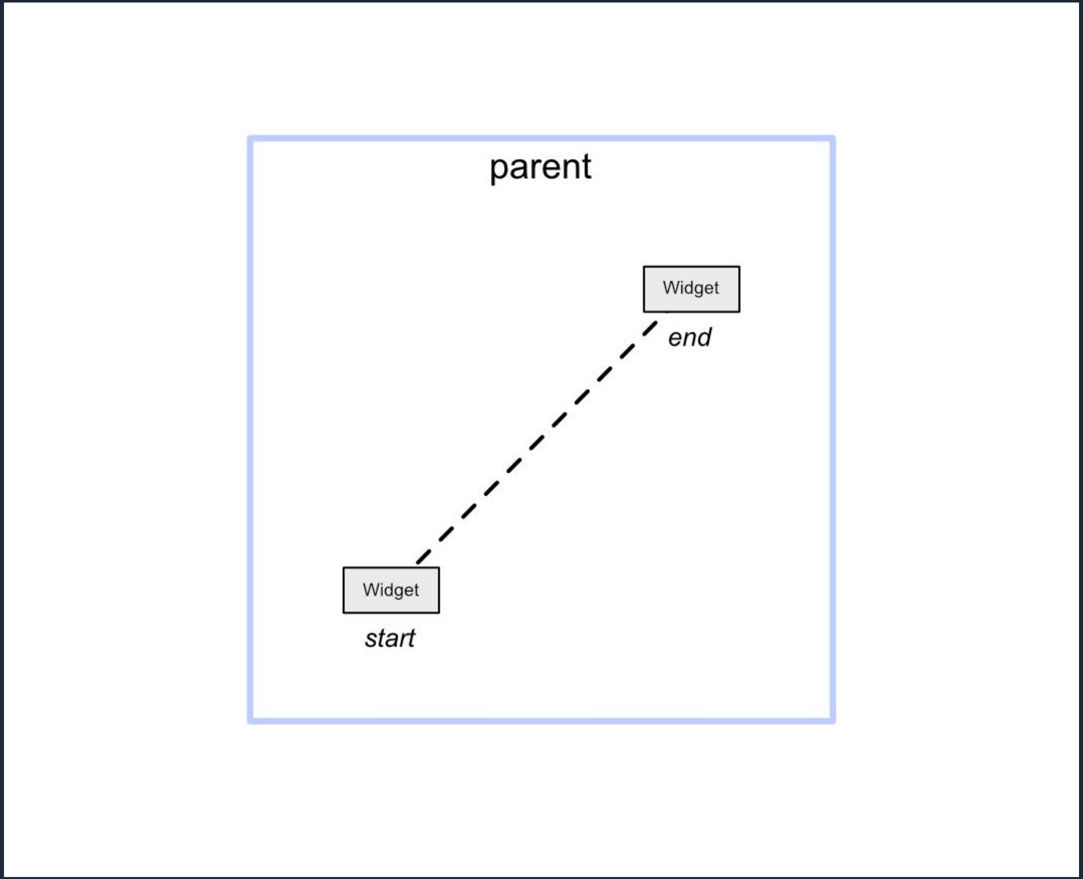
Fourth sample:
Keyframes are the key

Keyframes are the key

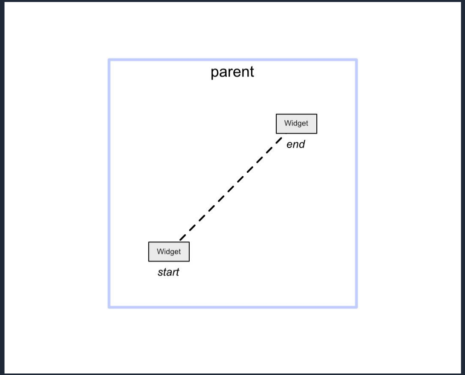
- Define animation based on it's progress
- Relative to position
- Start horizontal or vertical
- Rotate
- Scale

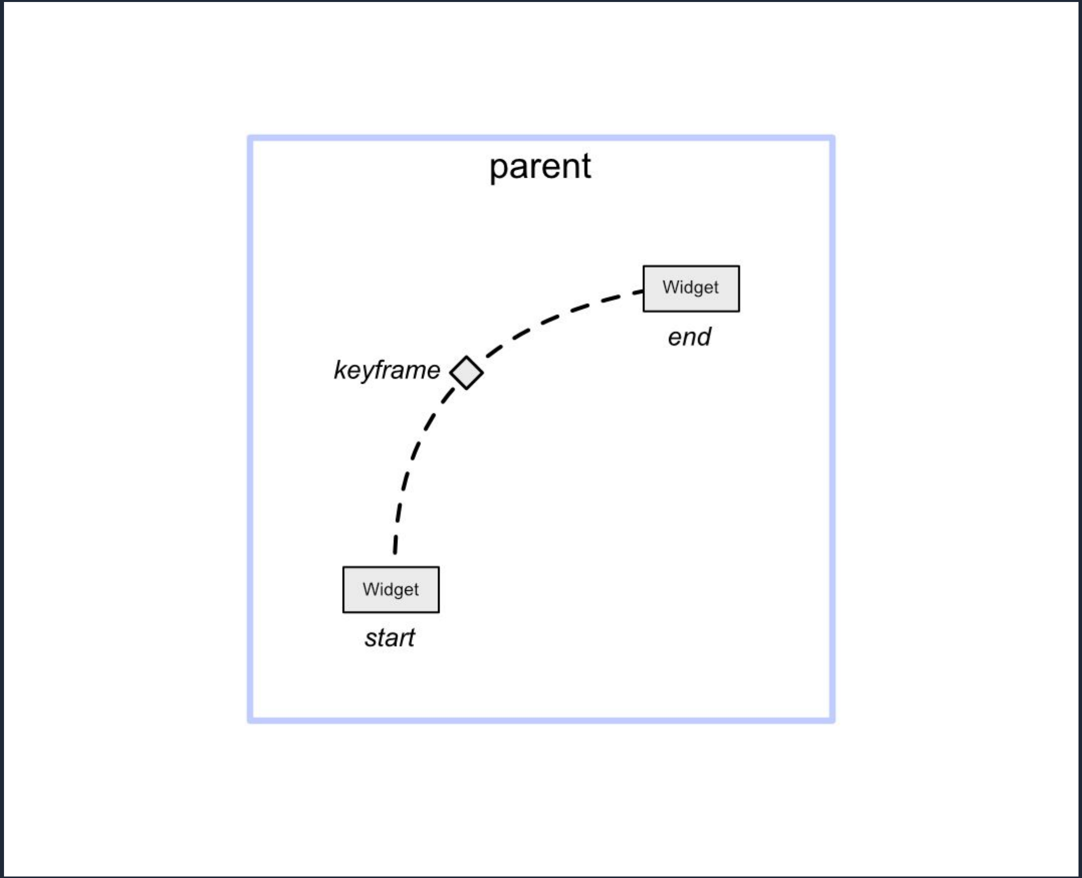


Keyframes are the key



Keyframes are the key





Keyframes are the key

- Once again just add the view with the simple button

```
// activity layout
<androidx.constraintlayout.motion.widget.MotionLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  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"
  app:layoutDescription="@xml/motion_scene_04"
  tools:showPaths="true">

  <View
     android:id="@+id/button"
     android:background="@color/colorAccent"
     android:layout_width="64dp"
     android:layout_height="64dp"
     android:text="Button" />

</androidx.constraintlayout.motion.widget.MotionLayout>
```

Keyframes are the key

- Once again just add the view with the simple button
- Add the motion scene with keyframe

```
// motion scene
<MotionScene ...>
  <Transition ...>
     <OnSwipe .../>

     <KeyFrameSet>
        <KeyAttribute
           android:scaleX="2"
           android:scaleY="2"
           android:rotation="-45"
           motion:framePosition="50"
           motion:target="@id/button" />
        <KeyPosition
           motion:keyPositionType="parentRelative"
           motion:percentX="0.2"
           motion:framePosition="50"
           motion:target="@id/button"/>
     </KeyFrameSet>

  </Transition>
  <ConstraintSet android:id="@+id/start">
     <Constraint ...>

        <CustomAttribute
           motion:attributeName="backgroundColor"
           motion:customColorValue="#D81B60"/>
     </Constraint>
  </ConstraintSet>

  <ConstraintSet android:id="@+id/end">
     <Constraint ...>

        <CustomAttribute
           motion:attributeName="backgroundColor"
           motion:customColorValue="#9999FF"/>
     </Constraint>
  </ConstraintSet>
</MotionScene>
```

Keyframes are the key

- Once again just add the view with the simple button
- Add the motion scene with keyframe
- RUN the app!!

Extra sample

Highlight

An Extra and more complex sample

References

- Introducing MotionLayout - *Nicolas Roard* - LINK
- The Motion Knight - *Arman Chatikyan* - LINK

Thank you