

[twitch.tv/dancojocar](https://twitch.tv/dancojocar)

[youtube.com/dancojocar](https://youtube.com/dancojocar)

# Lecture #9

# Animations

Mobile Applications 2020-2021

# Overview

- Add visual cues about what is going on.
- Useful when the UI changes states.
- Adding a polished look, gives a higher quality look and feel.
- Add motions to the UI.



# Property Animation

- Robust framework that allows to animate almost anything.
- Defines animation to change any object property over time.
- Characteristics of an animation:
  - Duration. Default length: 300ms.
  - Time interpolation. Defines how the values for the property are calculated.
  - Repeat count and behavior.
  - Animation sets.
  - Frame refresh delay. Default value: 10ms.

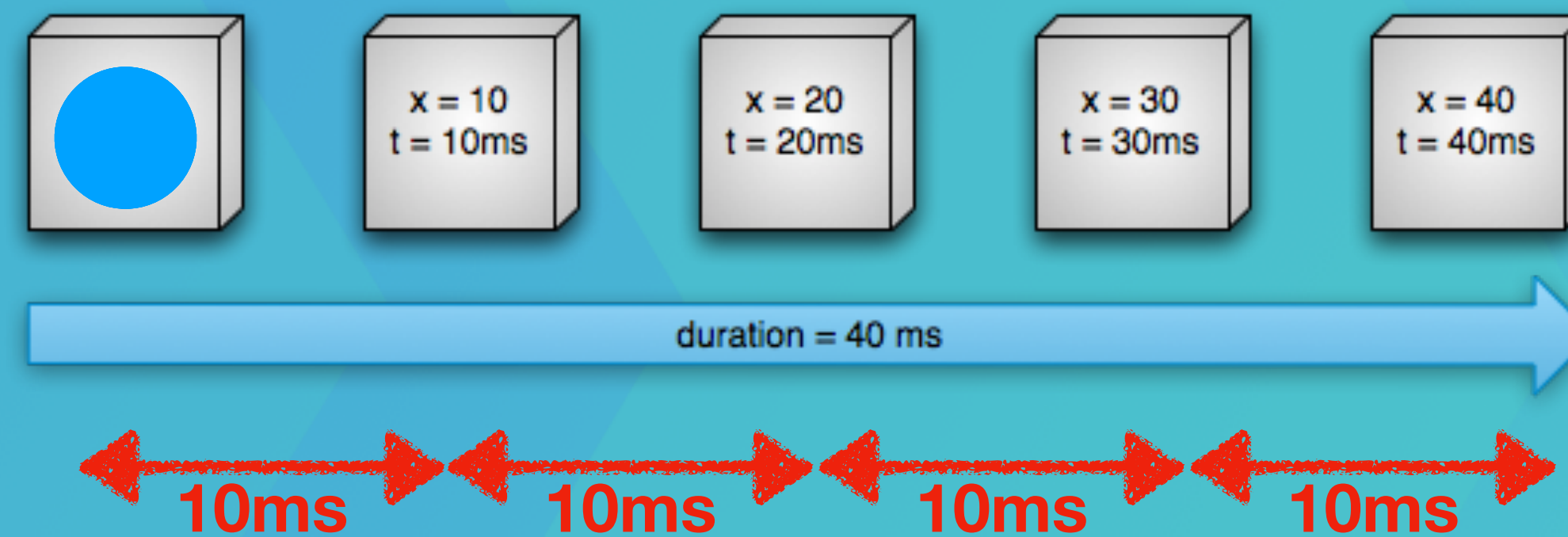


# How property animation works

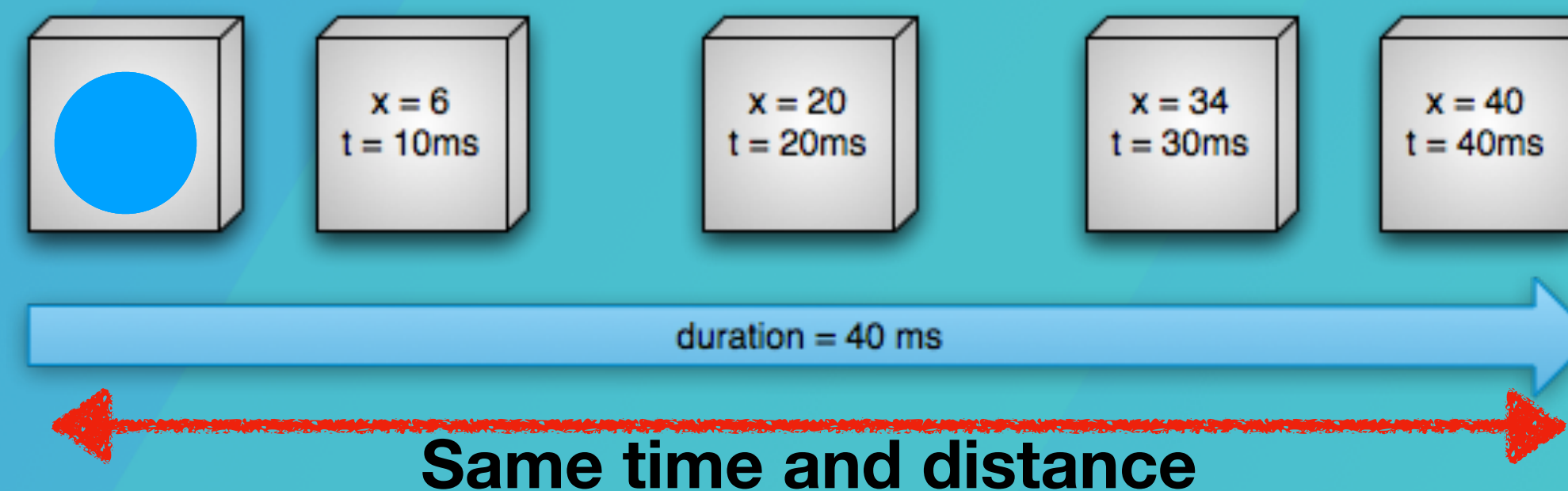
[twitch.tv/dancojocar](https://twitch.tv/dancojocar)

[youtube.com/dancojocar](https://youtube.com/dancojocar)

## Linear animation



## Non-linear animation



Same time and distance

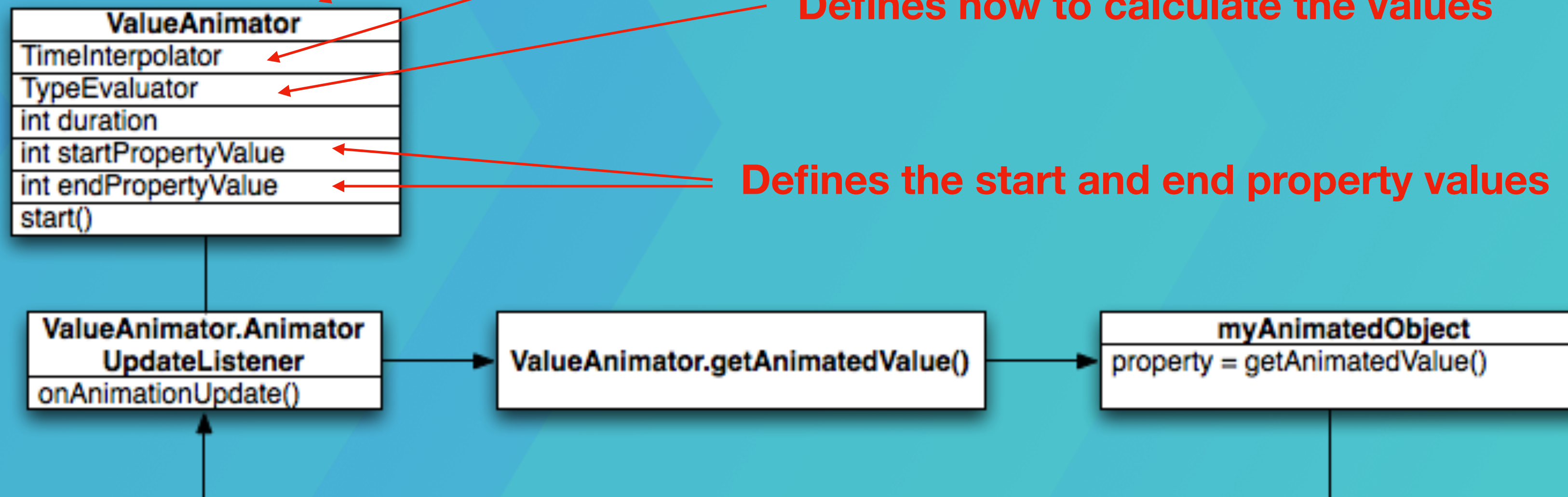
# Model

Keeps track of the animation timing

Defines the time interpolation

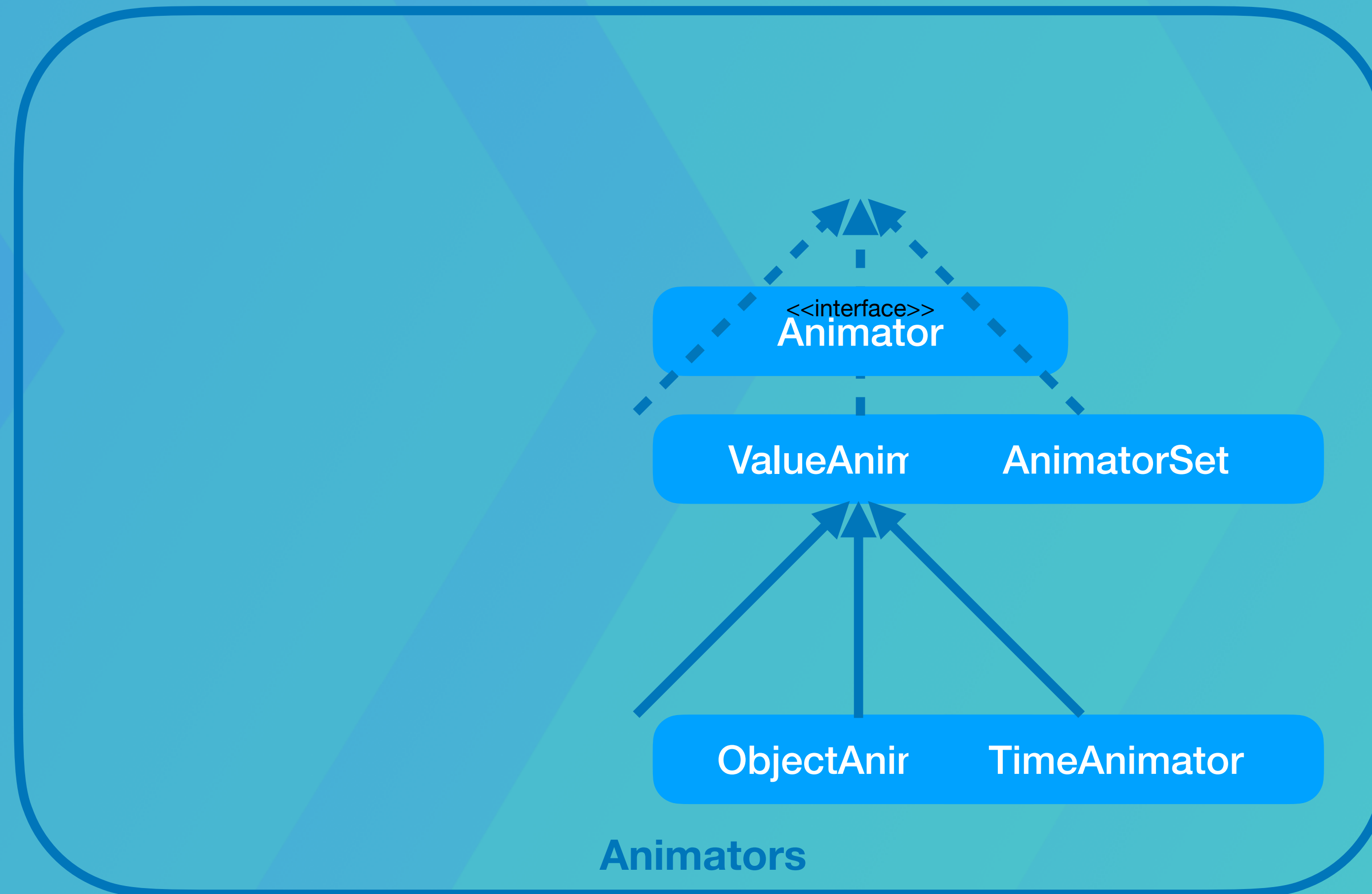
Defines how to calculate the values

Defines the start and end property values





# API



# API

Animators

IntEvaluator

FloatEvaluator

ArgbEvaluator

IntArrayEvaluator

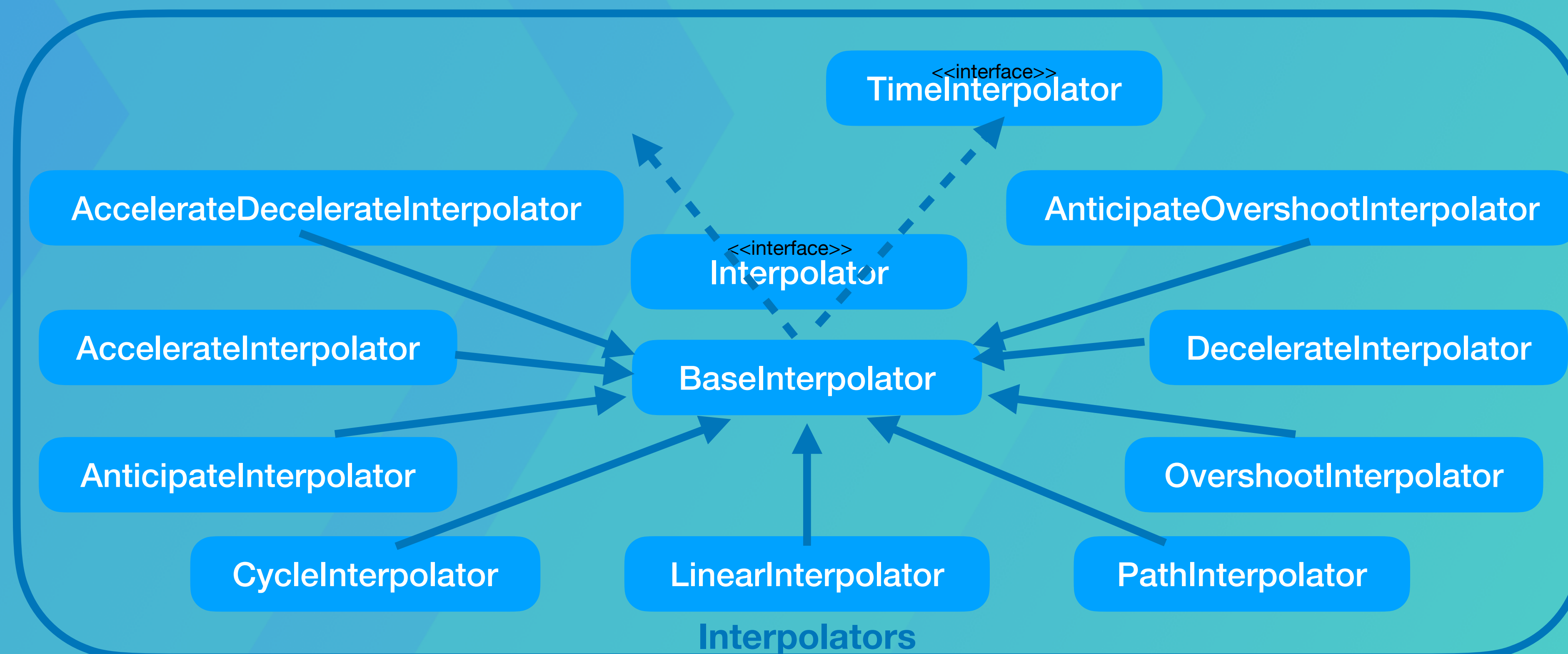
FloatArrayEvaluator

Evaluators

# API

Animators

Evaluators





# API

Animators

Evaluators

Interpolators

```
ValueAnimator.ofObject(...).apply {
    // ...
    addUpdateListener { updatedAnimation ->
        // You can use the animatedFloat(0f, 100f).apply {
        // same type as the animation. In this case, you can use the
        // float value in the translationX property.
        textView.translationX = updatedAnimation.animatedValue as Float
    }
        startPropertyValue, endPropertyValue).apply {
        // ...
        duration = 1000
    }
    start()
}

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

# Choreograph using an AnimatorSet

[twitch.tv/dancojocar](https://twitch.tv/dancojocar)

[youtube.com/dancojocar](https://youtube.com/dancojocar)

```
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 fadeAnim = ObjectAnimator.ofFloat(newBall, "alpha", 1f, 0f).apply {  
    duration = 250  
}  
  
AnimatorSet().apply {  
    play(bouncer).before(fadeAnim)  
    start()  
}
```

# Animation Listeners

```
ObjectAnimator.ofFloat(newBall, "alpha", 1f, 0f).apply {  
    duration = 250  
    addListener(object : AnimatorListenerAdapter() {  
        override fun onAnimationEnd(animation: Animator) {  
            balls.remove((animation as ObjectAnimator).target)  
        }  
    })  
}
```



# Animate Layout Changes

```
<LinearLayout  
    android:orientation="vertical"  
    android:layout_width="wrap_content"  
    android:layout_height="match_parent"  
    android:id="@+id/verticalContainer" />  
    android:animateLayoutChanges="true" />
```

# Animate View State Changes

[twitch.tv/dancojocar](https://twitch.tv/dancojocar)

[youtube.com/dancojocar](https://youtube.com/dancojocar)

DEMO

**Define:** res/xml/animate\_scale.xml

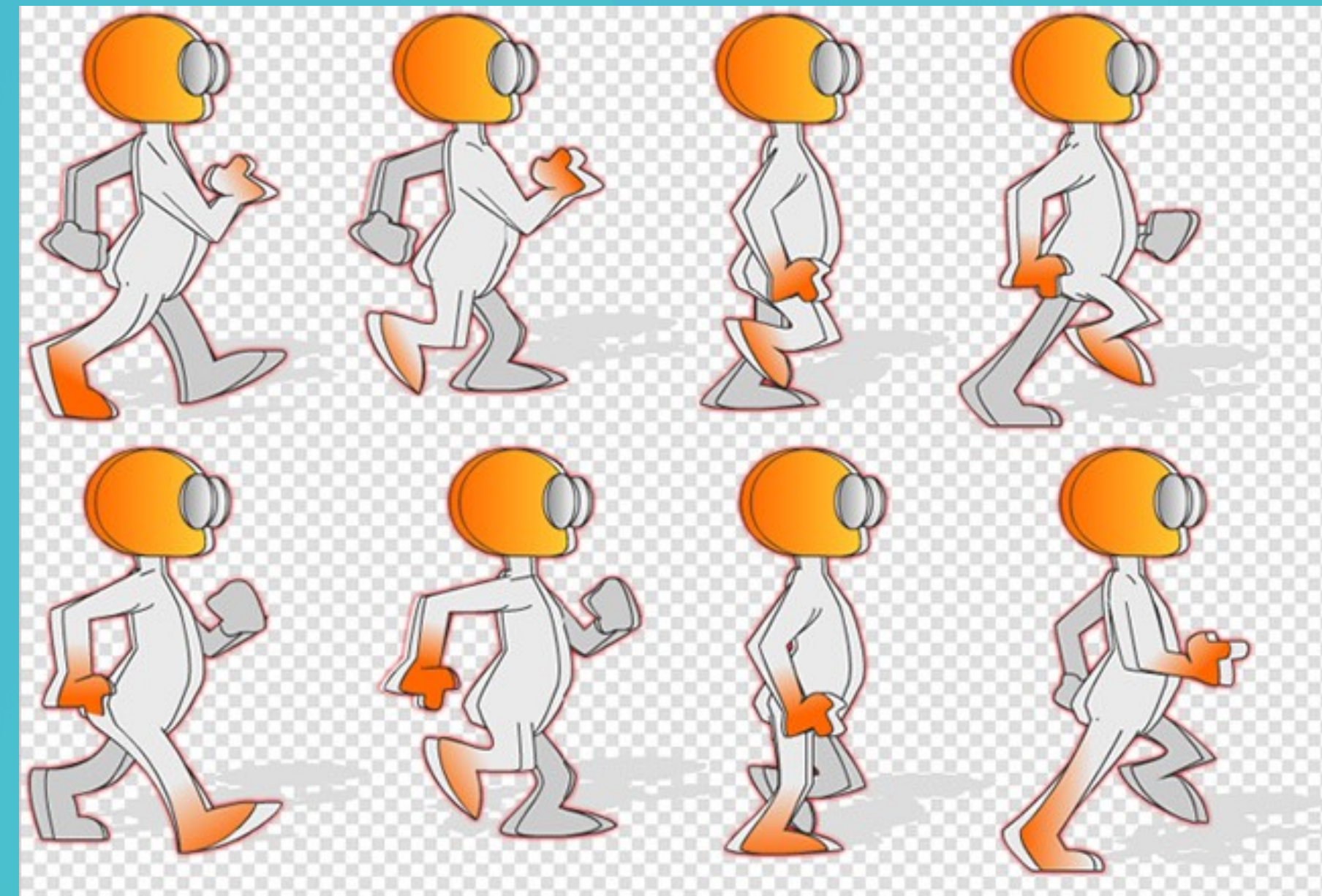
```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- the pressed state; increase x and y size to 150% -->
    <item android:state_pressed="true">
        <set>
            <objectAnimator android:propertyName="scaleX"
                android:duration="@android:integer/config_shortAnimTime"
                android:valueTo="1.5"
                android:valueType="floatType" />
            <objectAnimator android:propertyName="scaleY"
                android:duration="@android:integer/config_shortAnimTime"
                android:valueTo="1.5"
                android:valueType="floatType" />
        </set>
    </item>
    <!-- the default, non-pressed state; set x and y size to 100% -->
    <item android:state_pressed="false">
        <set>
            <objectAnimator android:propertyName="scaleX"
                android:duration="@android:integer/config_shortAnimTime"
                android:valueTo="1"
                android:valueType="floatType" />
            <objectAnimator android:propertyName="scaleY"
                android:duration="@android:integer/config_shortAnimTime"
                android:valueTo="1"
                android:valueType="floatType" />
        </set>
    </item>
</selector>

<Button android:id="@+id/button"
    android:stateListAnimator="@xml/animate_scale"
    ... />
```



# Animate bitmaps

- Used to animate a graphic such as:
  - An icon.
  - Illustration.
- Drawable animation API.
- Defined statically with a drawable resource or at runtime.





# Using an AnimationDrawable

twitch.tv/dancojocar

youtube.com/dancojocar



```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    private lateinit var rocketAnimation: AnimationDrawable
    android:oneshot="true"
    <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.main)

    val rocketImage = findViewById<ImageView>(R.id.rocket_image).apply {
        setBackgroundResource(R.drawable.rocket_thrust)
        rocketAnimation = background as AnimationDrawable
    }

    rocketImage.setOnClickListener({ rocketAnimation.start() })
}
```

<https://developer.android.com/guide/topics/graphics/drawable-animation>





# Move a View with Animation

twitch.tv/dancojocar

youtube.com/dancojocar

DEMO

Add curved motion

```
// arcTo() and PathInterpolator only available on API 21+
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    val path = Path().apply {
        arcTo(0f, 0f, 1000f, 1000f, 270f, 180f, false)
    }
    ObjectAnimator.ofFloat(view, "translationX", 100f).apply {
        duration = 2000
        val pathInterpolator = PathInterpolator(path)
        start()
    }
}

<pathAnimation xmlns:android="http://schemas.android.com/apk/res/android"
    interpolator="@android:interpolator
    start()
    android:controlY1="0"
    android:controlX2="1"
    android:controlY2="1"/>
```

<https://developer.android.com/training/animation/reposition-view>



# Animate Movement using Spring Physics

twitch.tv/dancojocar

youtube.com/dancojocar

```
findViewById<View>(R.id.imageView).also { img ->
    SpringAnimation(img, DynamicAnimation.TRANSLATION_Y).apply {
        ...
dependencies//{Setting up the spring with a low stiffness and
implementation 'com.nineoldandroids:library:2.6.0'
    }
    val velocity = vt.yVelocity
    }
    setStartVelocity(velocity)
val springAnim = findViewById<View>(R.id.imageView).let { img ->
    // Setting up a spring animation to animate the view's translationY property with the final
    // spring position at 0.
    SpringAnimation(img, DynamicAnimation.TRANSLATION_Y, 0f)
}
```

# Stiffness



Figure 6: High stiffness



Figure 7: Medium stiffness



Figure 8: Low stiffness



Figure 9: Very low stiffness

# Auto Animate Layout Updates

twitch.tv/dancojocar

youtube.com/dancojocar

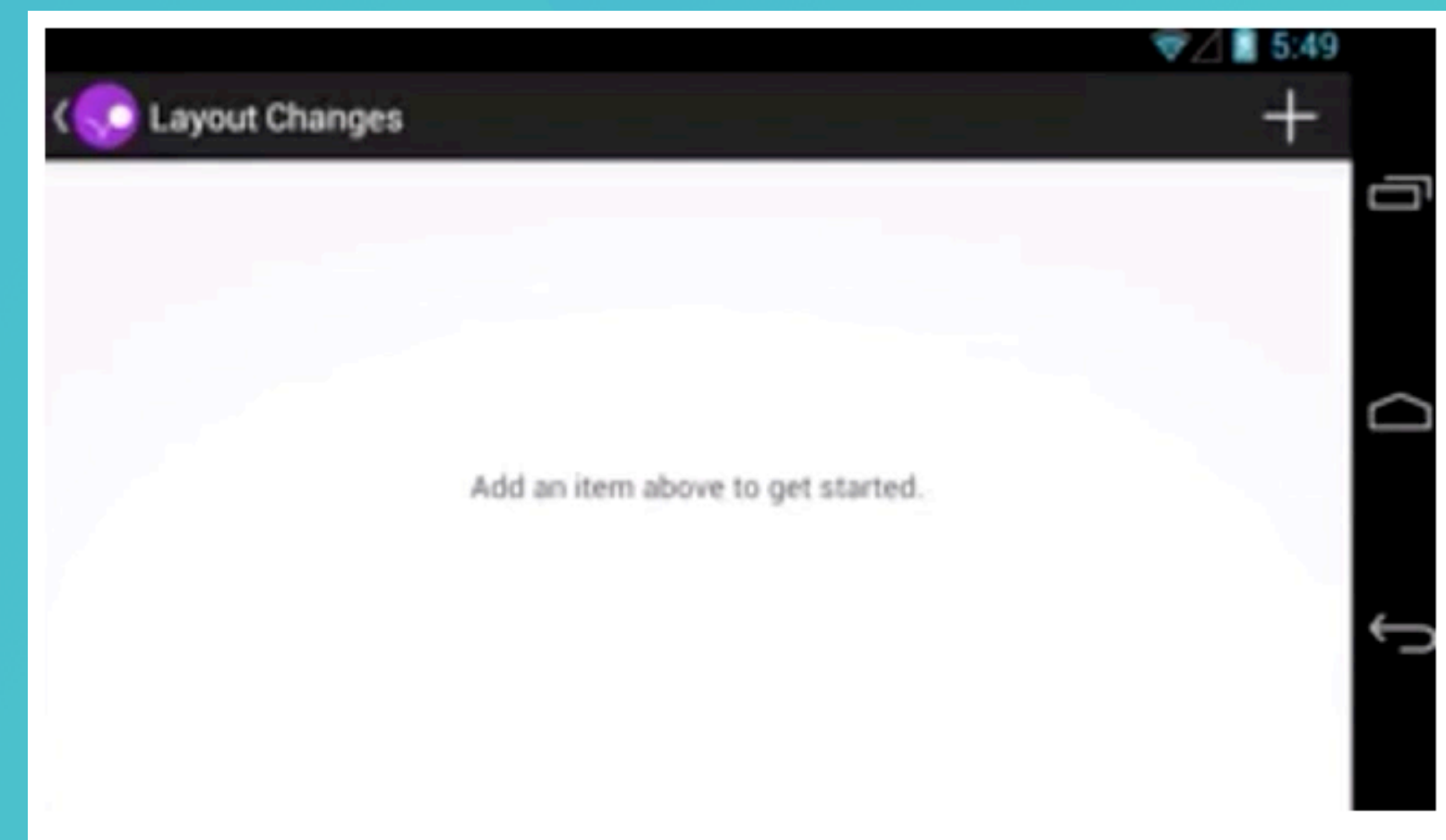
DEMO

Create the layout

```
<LinearLayout android:id="@+id/container"
    android:animateLayoutChanges="true"
    ...
/>
```

Add, update, or remove items from the layout

```
lateinit var mContainerView: ViewGroup
...
private fun addItem() {
    val newView: View = ...
    mContainerView.addView(newView, 0)
}
```



<https://developer.android.com/training/animation/layout>



# Animate Layout Changes Using Transitions

[twitch.tv/dancojocar](https://twitch.tv/dancojocar)

[youtube.com/dancojocar](https://youtube.com/dancojocar)

Define layouts for scenes

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/master_layout">
    <TextView
        android:id="@+id/title"
        ...
        android:text="Title" />
    <FrameLayout
        android:id="@+id/scene_root">
        <include layout="@layout/a_scene" />
    </FrameLayout>
</LinearLayout>

<!-- res/layout/another_scene.xml -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/scene_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/text_view2"
        android:text="Text Line 2" />
    <TextView
        android:id="@+id/text_view1"
        android:text="Text Line 1" />
</RelativeLayout>
```

# Create the Scene

Generate scenes from layouts

```
val mSceneRoot: ViewGroup = findViewById(R.id.scene_root)
val mAScene: Scene = Scene.getSceneForLayout(mSceneRoot, R.layout.a_scene, this)
val mAnotherScene: Scene = Scene.getSceneForLayout(mSceneRoot,
                                                    R.layout.another_scene, this)
```

Create a scene in your code

```
val mSceneRoot = mSomeLayoutElement as ViewGroup
val mViewHierarchy = someOtherLayoutElement as ViewGroup
val mScene: Scene = Scene(mSceneRoot, mViewHierarchy)
```

Apply a transition

```
var mFadeTransition: Transition =
    TransitionInflater.from(this)
        .inflateTransition(R.transition.fade_transition)

var mFadeTransition: Transition = Fade()

TransitionManager.go(mEndingScene, mFadeTransition)
```



# Start an Activity using an Animation

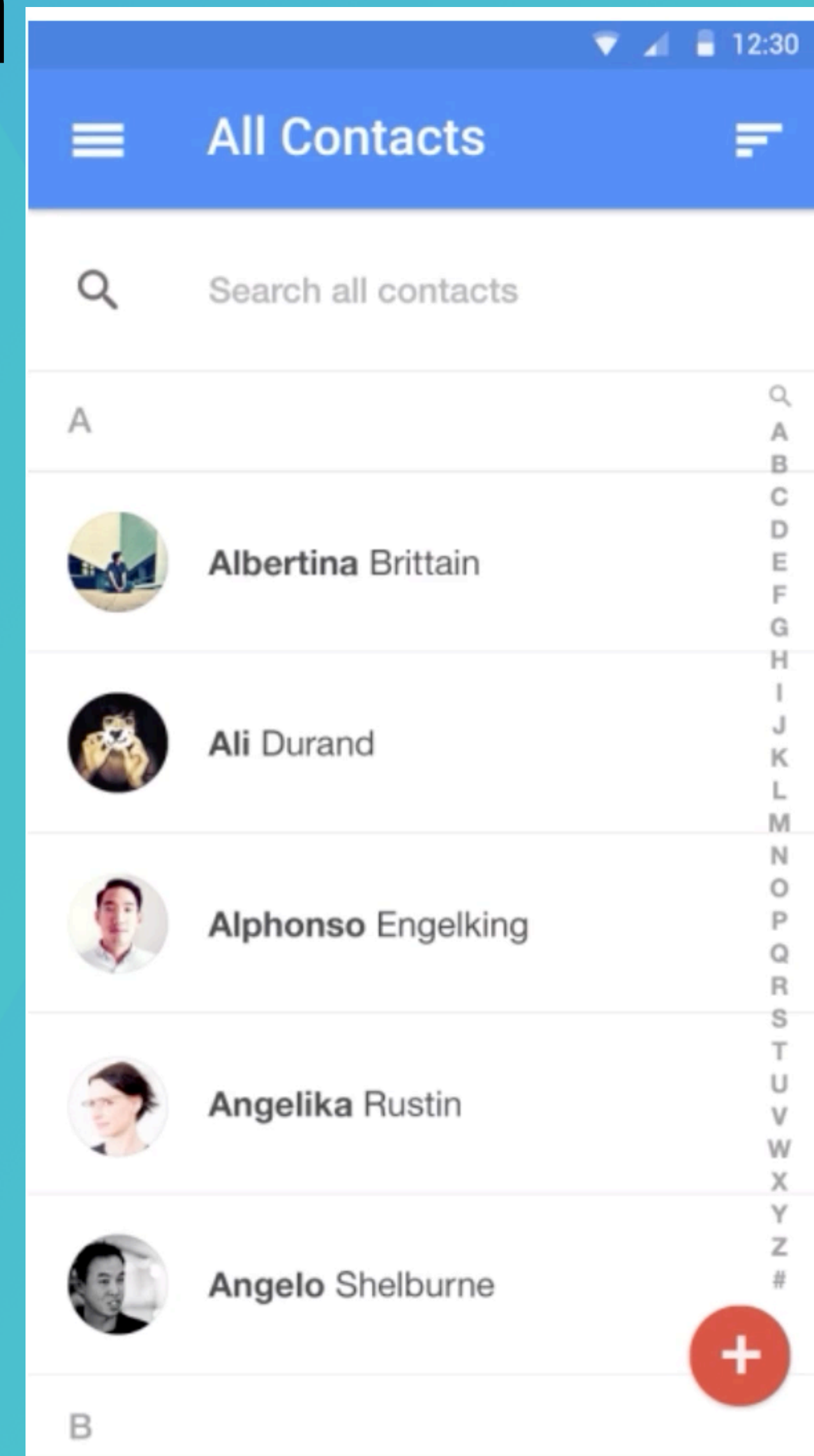
twitch.tv/dancojocar

youtube.com/dancojocar

DEMO

```
// get the element that receives the click event
val imgContainerView =
    findViewById<View>(R.id.img_container)

// get the common element for the
// transition in this activity
val androidRobotView =
    // Rename the Pair class from the
    // Android framework to avoid a name clash
    // import android.util.Pair as UtilPair
    // define a click listener
imgContainerView.setOnClickListener( {
    val intent = Intent(this, Activity2::class.java)
    // create the transition animation
    // - the images in the layouts
    // of both activities are defined
    // with android.transitionName
    val options = ActivityOptions
        .makeSceneTransitionAnimation(
            this, androidRobotView, "robot")
    // start the new activity
    startActivity(intent, options.toBundle())
})
```



# Lecture outcomes

- Animate bitmaps.
- Animate UI visibility and motion.
- Physics-based motion.
- Animate layout changes.
- Animate between activities.

