twitch.tv/dancojocar

youtube.com/dancojocar

Lecture #8
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.

https://material.io/design/motion

Property Animation

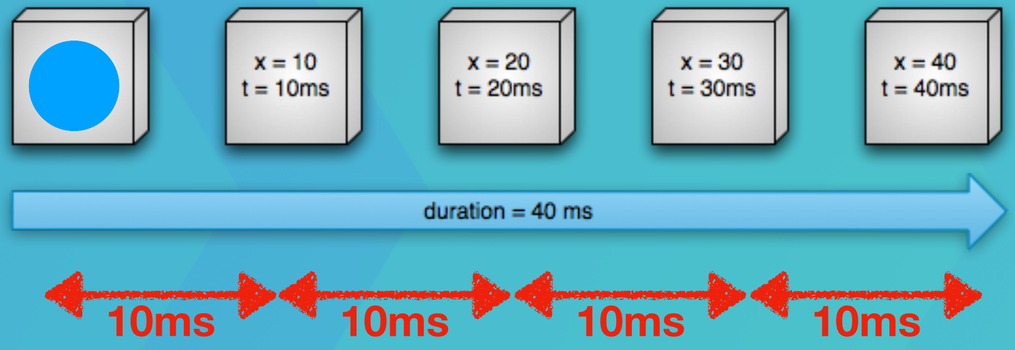
- Robust framework that allows to animate almost anything.
- Defines animation to change any object property over time.
- Characteristics of an animation:
  - Duration. Default length: 300ms.
  - Time interpolation. Defines how the values for the property are calculated.
  - Repeat count and behavior.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

How property animation works

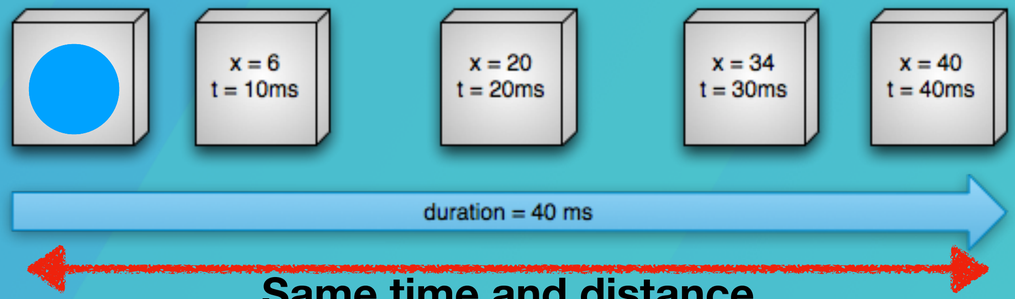
twitch.tv/dancojocar

youtube.com/dancojocar

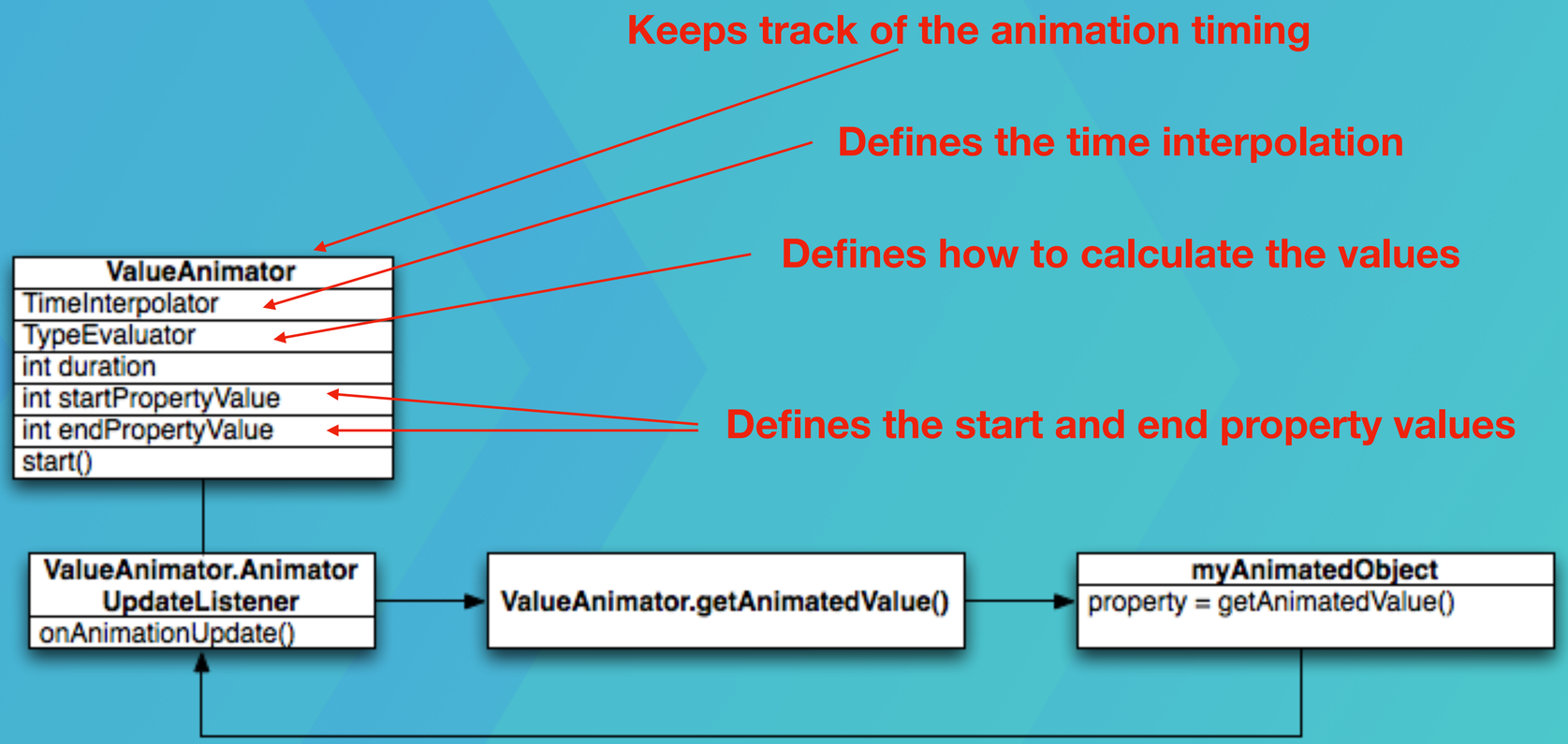
Linear animation



Non-linear animation

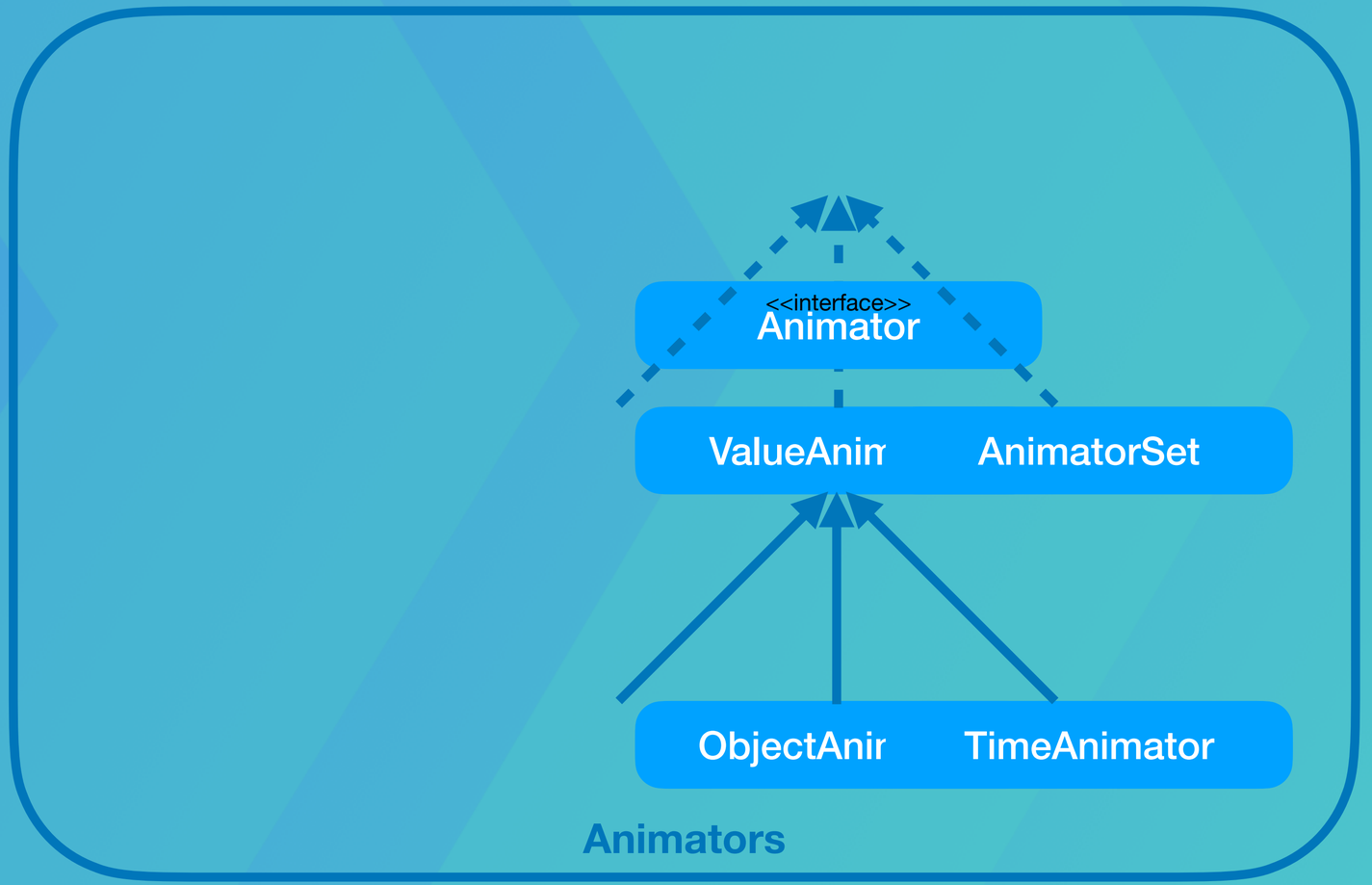


Model



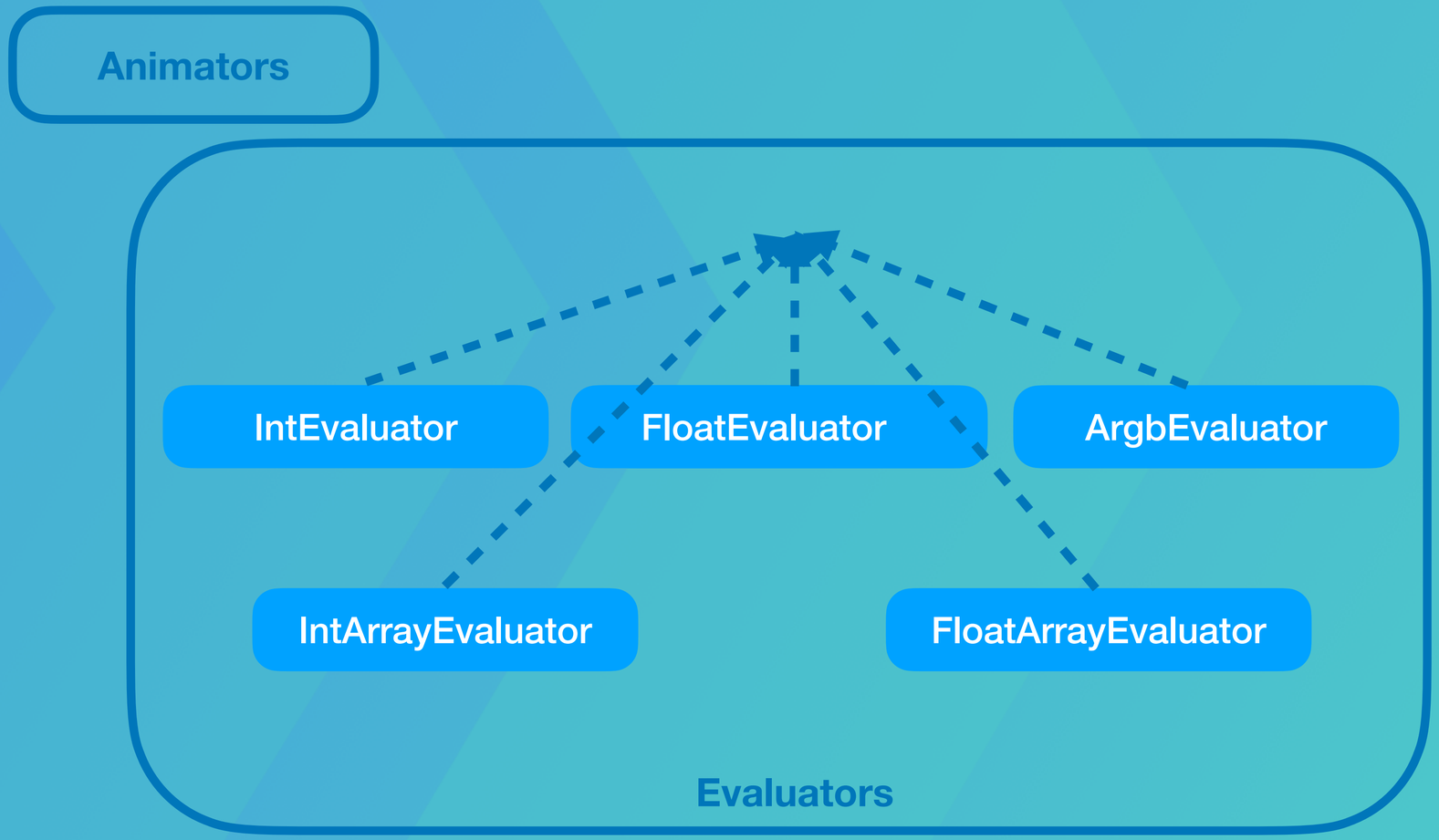
https://developer.android.com/guide/topics/graphics/prop-animation

API



https://developer.android.com/reference/android/animation/Animator

API

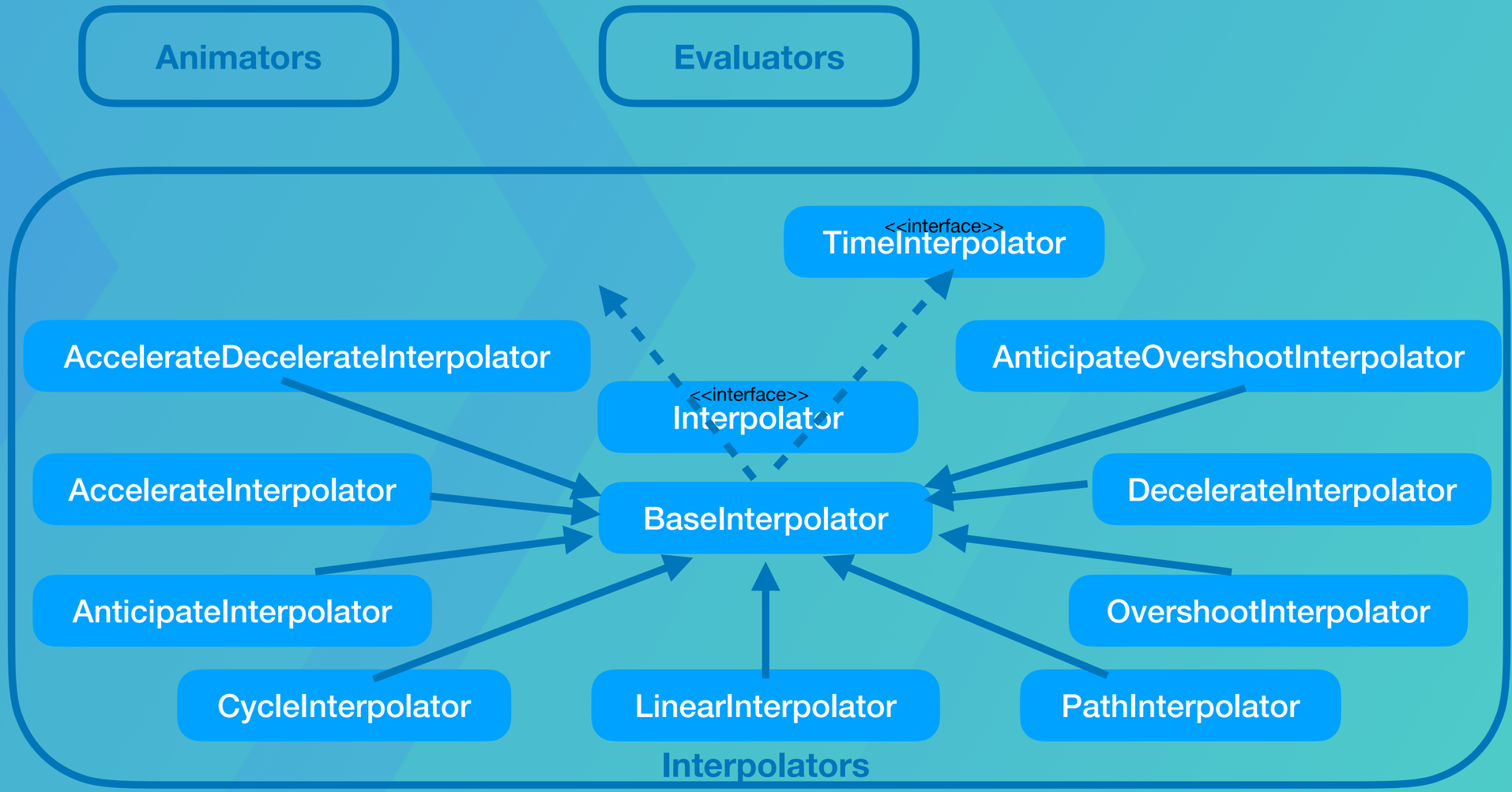


https://developer.android.com/reference/android/animation/TypeEvaluator

twitch.tv/dancojocar

youtube.com/dancojocar

API



https://developer.android.com/reference/android/view/animation/Interpolator

API

Animators

Evaluators

Interpolators

```
ValueAnimator.ofObject(...).apply {
  // ...
  addUpdateListener { updatedAnimation ->
    // You can use the animated value in a property that uses the
    // same type as the animation. In this case, you can use the
    // float value in the translationX property.
    textView.translationX = updatedAnimation.animatedValue as Float
  }
  // ...
}
```

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

Choreograph using an AnimatorSet

twitch.tv/dancojocar

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()
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

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)
    }
  })
}
```

twitch.tv/dancojocar

youtube.com/dancojocar

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

youtube.com/dancojocar

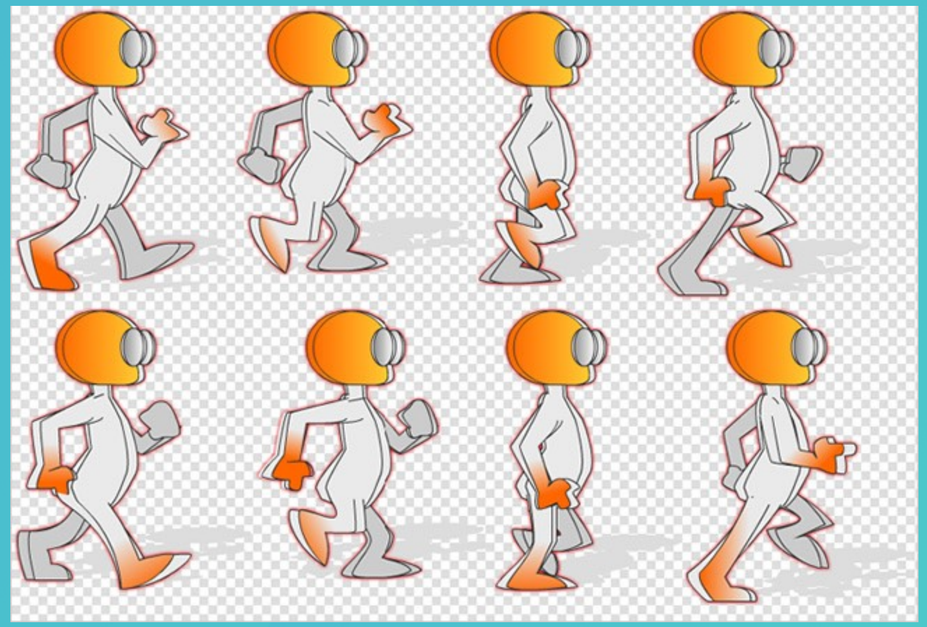
Define: res/xml/animate_scale.xml

```
<?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"
```

```
<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"
    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>
```

```
private lateinit var rocketAnimation: AnimationDrawable

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

Reveal or hide a view using animation

twitch.tv/dancojocar

youtube.com/dancojocar

```
Cross-fade animation
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ScrollView
        android:id="@+id/content_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="16dp" />
    <ProgressBar
        android:id="@+id/loading_spinner"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

```
private fun crossfade() {
    // Set the content view to 0% opacity but visible, so that it is visible
    // (but fully transparent) during the animation.
    contentView.alpha = 0f
    contentView.visibility = View.VISIBLE

    // Animate the content view to 100% opacity, and clear any animation
    // listener set on the view.
    contentView.animate()
        .alpha(1f)
        .setDuration(shortAnimationDuration.toLong())
        .setListener(null)

    // Animate the loading view to 0% opacity. After the animation ends,
    // set its visibility to GONE as an optimization step (it won't
    // participate in layout passes, etc.)
    loadingView.animate()
        .alpha(0f)
        .setDuration(shortAnimationDuration.toLong())
        .setListener(object : AnimatorListenerAdapter() {
            override fun onAnimationEnd(animation: Animator) {
```

```
private var currentAnimator: Animator? = null

private val shortAnimationDuration: Int = 0

override fun onCreate(savedInstanceState: Bundle?) {
    ...

    // Retrieve and cache the system's default "short" animation time.
    shortAnimationDuration = resources.getInteger(
            android.R.integer.config_shortAnimTime)
}
```

https://developer.android.com/training/animation/reveal-or-hide-view

Move a View with Animation

twitch.tv/dancojocar

youtube.com/dancojocar

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, true)
    }
    val pathInterpolator = PathInterpolator(path)
}
```

```
ObjectAnimator.ofFloat(view, "translationX", 100f).apply {
    duration = 2000
    interpolator = pathInterpolator
    start()
}
```

```
<pathInterpolator xmlns:android="http://schemas.android.com/apk/res/android"
    android:controlX1="0"
    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 {
        …
        // Setting the spring with a low stiffness
        spring.stiffness = SpringForce.STIFFNESS_LOW
        …
        val velocity = vt.yVelocity
        setStartVelocity(velocity)
    }
}
```

```
dependencies {
  implementation 'androidx.dynamicanimation:dynamicanimation:1.0.0'
}
```

```
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)
}
```

https://developer.android.com/reference/android/view/VelocityTracker

Stiffness

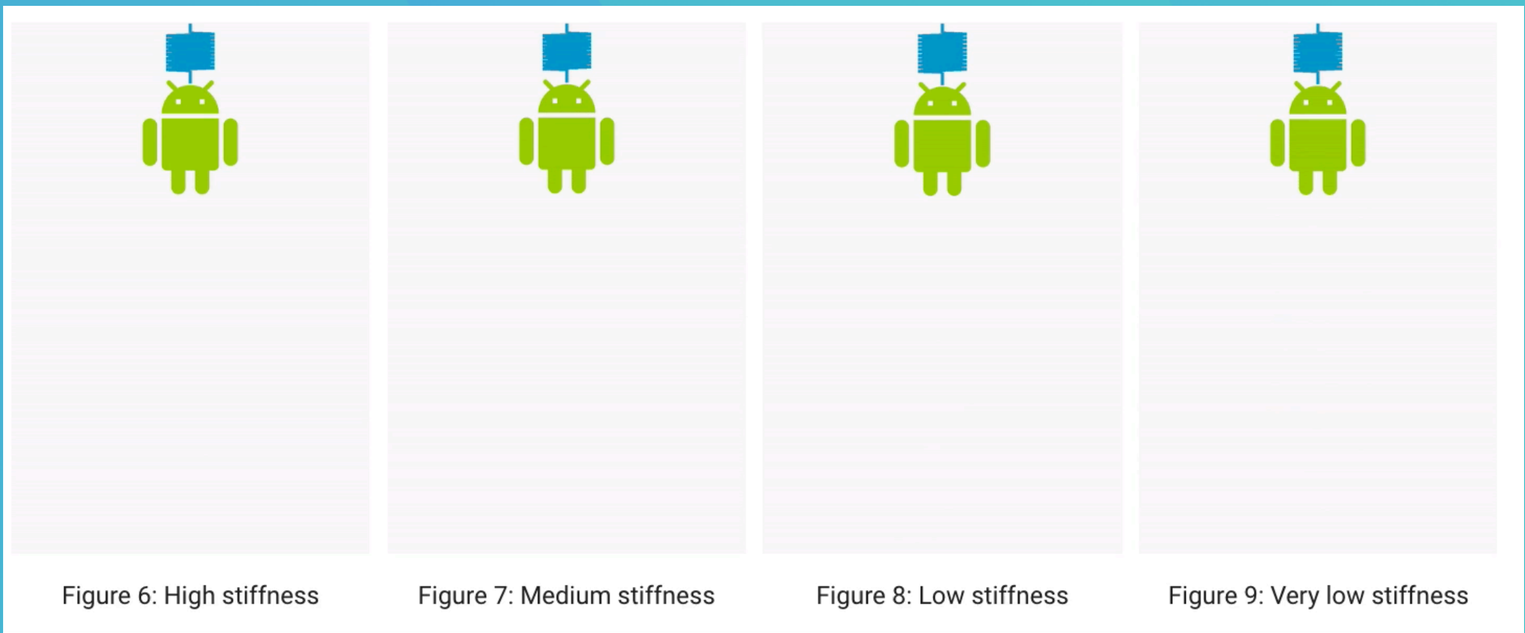


Figure 6: High stiffness

Figure 7: Medium stiffness

Figure 8: Low stiffness

Figure 9: Very low stiffness

Auto Animate Layout Updates

twitch.tv/dancojocar

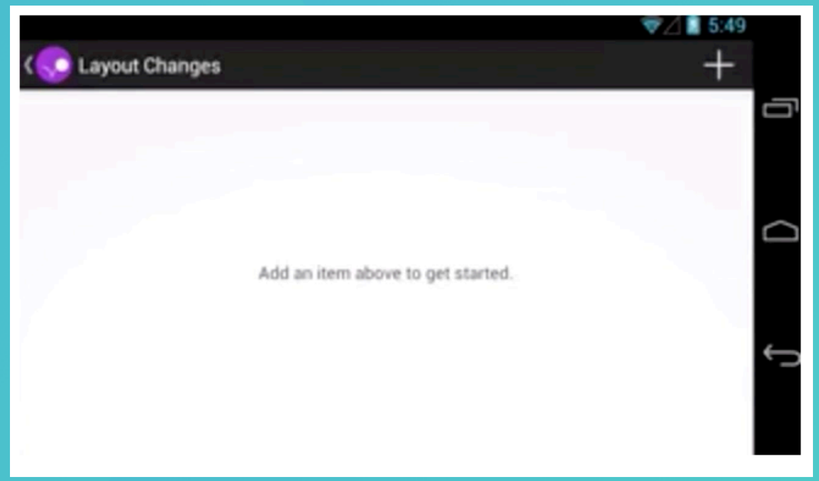
youtube.com/dancojocar

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

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_view2
        android:text="Text Line 1" />
</RelativeLayout>
```

twitch.tv/dancojocar

youtube.com/dancojocar

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

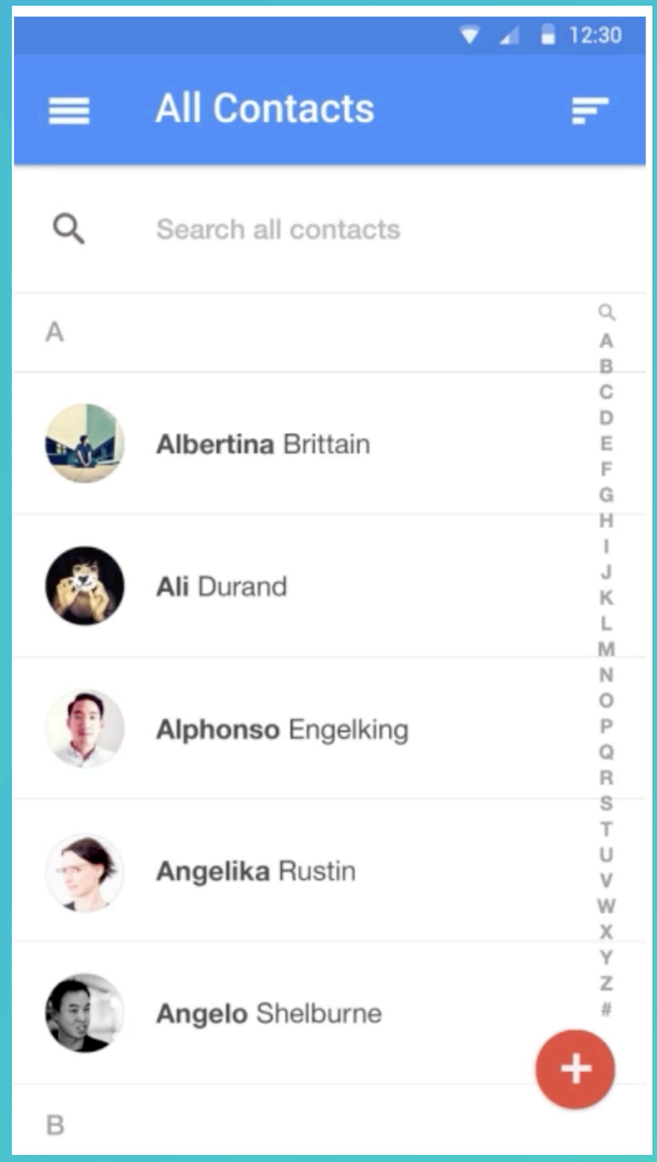
DEMO

```
// 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 =
        findViewById<View>(R.id.image_small)
//define a click listener
imgContainerView.setOnClickListener( {
    // create the transition animation
    // - the images in the layouts
    // of both activities are defined
    // with android:transitionName="robot"
    val options = ActivityOptions
            .makeSceneTransitionAnimation(
                this, androidRobotView, "robot")
    // start the new activity
    startActivity(intent, options.toBundle())
})
```

```
// Rename the Pair class from the
// Android framework to avoid a name clash
import android.util.Pair as UtilPair

val intent = Intent(this, Activity2::class.java)
val options = ActivityOptions.makeSceneTransitionAnimation(
        this,
        UtilPair.create(view1, "agreedName1"),
        UtilPair.create(view2, "agreedName2")
)
```



Lecture outcomes

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