18-MotionLayout 讲义

过渡动画

过渡动画使用

有两种使用方式

- `TransitionManager#beginDelayedTransition()`
- `TransitionManager#go()`

> 其中使用 `go()` 调用，需要重新绑定数据（因为 View 会被重新添加）。

过渡动画原理

过渡动画指的是两个场景之间的过渡，一个「开始场景」一个「结束场景」

1. 我们就要从场景上记录里面控件的各种参数
2. 根据两个场景的各种参数，创建出属性动画，播放属性动画

对应 Transition 上的关键函数

1. 捕获并记录对应属性

   `captureStartValues` 和 `captureEndValues`
2. 基于捕获的值创建对应动画，并播放动画

   `createAnimator` 和 `playTransition`

MotionLayout

MotionLayout 使用

- ConstraintLayout 改为 MotionLayout
- 创建 MotionScene 文件并使用 `app:layoutDescription` 关联

Constraint 节点

```
<Constraint android:id="">
  <Motion/>
  <Layout/>
  <Transform/>
  <CustomAttribute/>
  <PropertySet/>
</Constraint>
```

- 运动模型

  弧线路径，时间模型等

- 布局相关

  注意，`width`、`height` 和 `margin` 的命名空间是 `android:` (beta1 开始)

  而约束相关的命名空间是 `app`（或 `motion`）

- 动画变换

  做旋转，位移，缩放，海拔等属性

- 自定义属性
  - `attributeName` 会加上 `set/get` 反射找到真正的函数名，比如 `backgroundColor` 就会调用 `setBackgroundColor()` 函数
  - `custom(xxx)Value` 对应属性的数据类型
- 特定的属性

  `visibility`、`alpha` 等属性

时间模型

- 每个控件可以单独设置时间模型

  在 `<Motion>` 节点中使用 `app:transitionEasing`

- 整个 Transition 也可以设置插值器

  在 `<Transition>` 中使用 `app:motionInterpolator`

MotionScene

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

  <Transition
           app:constraintSetStart="@+id/start"
           app:constraintSetEnd="@+id/end"/>

  <ConstraintSet android:id="@+id/start"/>
  <ConstraintSet android:id="@+id/end"/>
</MotionScene>
```

Transition 节点

```
<Transition
  app:constraintSetStart="@+id/start"
  app:constraintSetEnd="@+id/end"
  app:duration="1000">
    <OnSwipe />
    <OnClick />
    <KeyPositionSet >
        <KeyAttribute>
            <CustomAttribute/>
        </KeyAttribute>
        <KeyPostion/>
        <KeyCycle/>
        <KeyTimeCycle/>
    </KeyPositionSet>
</Transition>
```

onSwipe

```
<OnSwipe
         app:touchAnchorId="@id/view"
         app:dragDirection="dragDown"/>
```

`<OnSwipe>` 添加在 `<Transtion>` 节点中支持的参数：

- `touchRegionId` 指的是哪一个控件响应触摸事件。
- `touchAnchorId` 一般不用
- `autoComplete` 默认的是 true ， 会根据动画完成的百分比自动到最近的一个状态
- `dragDirection` 拖拽的方向

OnClick

```
<OnClick
        app:clickAction="toggle"
        app:targetId="@id/toggle"/>
```

- targetId 指定控件
- clickAction
  - toggle 反转状态
  - transitionToEnd/Start 通过动画到结束/起始状态
  - jumpToEnd/Start 没有动画直接到结束/起始状态

KeyPositionSet

- `app:motionTarget` 目标对象 ID
- `app:framePosition` 百分比 (0 - 100)

KeyAttribute 属性关键字

```
<KeyAttribute
               app:framePosition="0"
               app:motionTarget="@id/image_film_cover"
               android:elevation="12dp">
    <CustomAttribute
               app:attributeName="BackgroundColor"
               app:customColorValue="@color/colorAccent"/>
</KeyAttribute>
```

可以设置位移，旋转，缩放等属性，同时还可以通过 `CustomAttribute` 添加自定义属性

KeyPosition 位置关键帧

`percentX/Y` 在关键帧时，对应路径的对应百分比

`percentWidth/Height` 在关键帧时，控件大小改变的百分比

`curveFit` 运动路径的样式（直线，曲线等）

`keyPositionType` 坐标系

- `parentRelative`

  （0，0）为父容器左上角

  （1，1）为父容器右下角

- `deltaRelative`

  （0，0）为起始控件中心

  （1，1）为结束控件中心

- `pathRelative`

  （0，0）为起始控件中心

  （1，0）为结束控件中心

KeyCycle 和 KeyTimeCycle

通过 3 个 KeyCycle 定义一个准确的循环关键帧

```
<KeyFrameSet>
  <KeyCycle
          android:rotation="0"
          app:motionTarget="@id/fab_favourite"
          app:wavePeriod="0"
          app:framePosition="0"
          app:waveShape="sin"/>

  <KeyCycle
          android:rotation="180"
          app:motionTarget="@id/fab_favourite"
          app:wavePeriod="3"
          app:framePosition="50"
          app:waveShape="sin"/>

  <KeyCycle
          android:rotation="0"
          app:motionTarget="@id/fab_favourite"
          app:wavePeriod="0"
          app:framePosition="100"
          app:waveShape="sin"/>
</KeyFrameSet>
```

`wavePeriod` 循环次数 （KeyTimeCycle 表示的每秒循环次数）

`waveShape` 数学模型