创建自定义过渡动画

通过自定义过渡，您可以创建无法通过任何内置过渡类获得的动画。例如，您可以定义一个自定义过渡，该过渡使文本和输入字段的前景颜色变为灰色来表明这些字段已在新屏幕中停用。此类更改有助于用户看到您已停用的字段。

自定义过渡与某种内置过渡类型一样，可将动画应用于起始和结束场景的子视图。不过，与内置过渡类型不同，您必须提供捕获属性值和生成动画的代码。您可能还需要为动画定义目标视图的子集。

本文介绍了如何捕获属性值并生成动画以创建自定义过渡。

扩展过渡类

如需创建自定义过渡，请向您的项目添加一个类以扩展 `Transition` (/reference/android/transition/Transition) 类，并替换以下代码段中显示的函数：

KOTLIN (#KOTLIN)JAVA

```
public class CustomTransition extends Transition {

    @Override
    public void captureStartValues(TransitionValues values) {}

    @Override
    public void captureEndValues(TransitionValues values) {}

    @Override
    public Animator createAnimator(ViewGroup sceneRoot,
                                   TransitionValues startValues,
                                   TransitionValues endValues) {}
}
```

下面的部分介绍了如何替换这些函数。

捕获视图属性值

过渡动画使用属性动画 (/guide/topics/graphics/prop-animation)中所述的属性动画系统。属性动画可以更改指定时间段内的起始值和结束值之间的视图属性，因此框架需要同时具有属性的起始值和结束值才能构建动画。

不过，属性动画通常只需要视图的所有属性值中的一小部分。例如，颜色动画需要颜色属性值，而移动动画需要位置属性值。由于动画所需的属性值因过渡而异，因此过渡框架不会为某个过渡提供所有属性值。相反，框架会调用允许过渡仅捕获其所需的属性值并将它们存储到框架中的回调函数。

捕获起始值

如需将起始视图值传递给框架，请实现 `captureStartValues(transitionValues)` (/reference/android/transition/Transition#captureStartValues(android.transition.TransitionValues)) 函数。该框架会针对起始场景中的每个视图调用此函数。该函数参数是一个 `TransitionValues` (/reference/android/transition/TransitionValues) 对象，其中包含对视图的引用和 `Map` (/reference/java/util/Map) 实例，您可以在该实例中存储所需的视图值。在您的实现中，通过将这些属性值存储到 Map 实例中对其进行检索并将它们回传至框架。

为了确保属性值的键不会与其他 `TransitionValues` (/reference/android/transition/TransitionValues) 键冲突，请使用以下命名方案：

```
package_name:transition_name:property_name
```

以下代码段展示了 **`captureStartValues()`** (/reference/android/transition/Transition#captureStartValues(android.transition.TransitionValues)) 函数的一个实现：

KOTLIN (#KOTLIN)JAVA

```
public class CustomTransition extends Transition {

    // Define a key for storing a property value in
    // TransitionValues.values with the syntax
    // package_name:transition_class:property_name to avoid collisions
    private static final String PROPNAME_BACKGROUND =
            "com.example.android.customtransition:CustomTransition:background";

    @Override
    public void captureStartValues(TransitionValues transitionValues) {
        // Call the convenience method captureValues
        captureValues(transitionValues);
    }

    // For the view in transitionValues.view, get the values you
    // want and put them in transitionValues.values
    private void captureValues(TransitionValues transitionValues) {
        // Get a reference to the view
        View view = transitionValues.view;
        // Store its background property in the values map
        transitionValues.values.put(PROPNAME_BACKGROUND, view.getBackground());
    }
    ...
}
```

捕获结束值

该框架会针对结束场景中的每个目标视图调用一次 **`captureEndValues(TransitionValues)`** (/reference/android/transition/Transition#captureEndValues(android.transition.TransitionValues)) 函数。在所有其他方面，**`captureEndValues()`** (/reference/android/transition/Transition#captureEndValues(android.transition.TransitionValues)) 的工作原理与 **`captureStartValues()`** (/reference/android/transition/Transition#captureStartValues(android.transition.TransitionValues)) 相同。

以下代码段展示了 **`captureEndValues()`** (/reference/android/transition/Transition#captureEndValues(android.transition.TransitionValues)) 函数的一个实现：

KOTLIN (#KOTLIN)JAVA

```
@Override
public void captureEndValues(TransitionValues transitionValues) {
    captureValues(transitionValues);
}
```

在本例中，**`captureStartValues()`** (/reference/android/transition/Transition#captureStartValues(android.transition.TransitionValues)) 和 **`captureEndValues()`** (/reference/android/transition/Transition#captureEndValues(android.transition.TransitionValues)) 函数都会调用 **`captureValues()`** 以检索和存储值。**`captureValues()`** 检索的视图属性是相同的，但在起始场景和结束场景中具有不同的值。该框架会为视图的起始状态和结束状态维护单独的 Map 实例。

创建自定义动画程序

如需为视图在起始场景中的状态和结束场景中的状态之间的变化添加动画效果，请通过替换 **`createAnimator()`** (/reference/android/transition/Transition#createAnimator(android.view.ViewGroup, android.transition.TransitionValues, android.transition.TransitionValues)) 函数来提供动画程序。当框架调用此函数时，它会传入场景根视图和包含您捕获的起始值和结束值的 **`TransitionValues`** (/reference/android/transition/TransitionValues) 对象。

框架调用 `createAnimator()` (/reference/android/transition/Transition#createAnimator(android.view.ViewGroup, android.transition.TransitionValues, android.transition.TransitionValues)) 函数的次数取决于起始场景和结束场景之间发生的变化。例如，假设将淡出/淡入动画作为自定义过渡实现。如果起始场景具有五个目标，其中两个目标从结束场景中移除，而结束场景具有起始场景中的三个目标以及一个新目标，框架将会调用 `createAnimator()` (/reference/android/transition/Transition#createAnimator(android.view.ViewGroup, android.transition.TransitionValues, android.transition.TransitionValues)) 六次：其中三次调用为驻留在两个场景对象中的目标的淡出/淡入添加动画效果；另外两次调用为从结束场景中移除的目标的淡出添加动画效果；还有一次调用为结束场景中新目标的淡入添加动画效果。

对于同时存在于起始场景和结束场景中的目标视图，该框架会为 `startValues` 和 `endValues` 参数提供 `TransitionValues` (/reference/android/transition/TransitionValues) 对象。对于仅存在于起始场景或结束场景中的目标视图，该框架会为相应的参数提供 `TransitionValues` (/reference/android/transition/TransitionValues) 对象，并为其他参数提供 `null`。

如需在创建自定义过渡时实现 `createAnimator(ViewGroup, TransitionValues, TransitionValues)` (/reference/android/transition/Transition#createAnimator(android.view.ViewGroup, android.transition.TransitionValues, android.transition.TransitionValues)) 函数，请使用您捕获的视图属性值创建 `Animator` (/reference/android/animation/Animator) 对象并将其返回到框架。如需查看示例实现，请参阅 CustomTransition (https://github.com/googlesamples/android-CustomTransition) 示例中的 `ChangeColor` (https://github.com/android/animation-samples/blob/master/CustomTransition/Application/src/main/java/com/example/android/customtransition/ChangeColor.java) 类。如需详细了解属性动画程序，请参阅属性动画 (/guide/topics/graphics/prop-animation)。

应用自定义过渡

自定义过渡与内置过渡的工作原理相同。您可以使用过渡管理程序来应用自定义过渡，如应用过渡 (/training/transitions/transitions#Apply)中所述。