使用动画启动 Activity

Material Design 应用中的 Activity 过渡通过常见元素之间的动画和转换效果，在不同状态之间建立视觉联系。您可以为进入和退出过渡，以及 Activity 之间共享元素的过渡指定自定义动画。

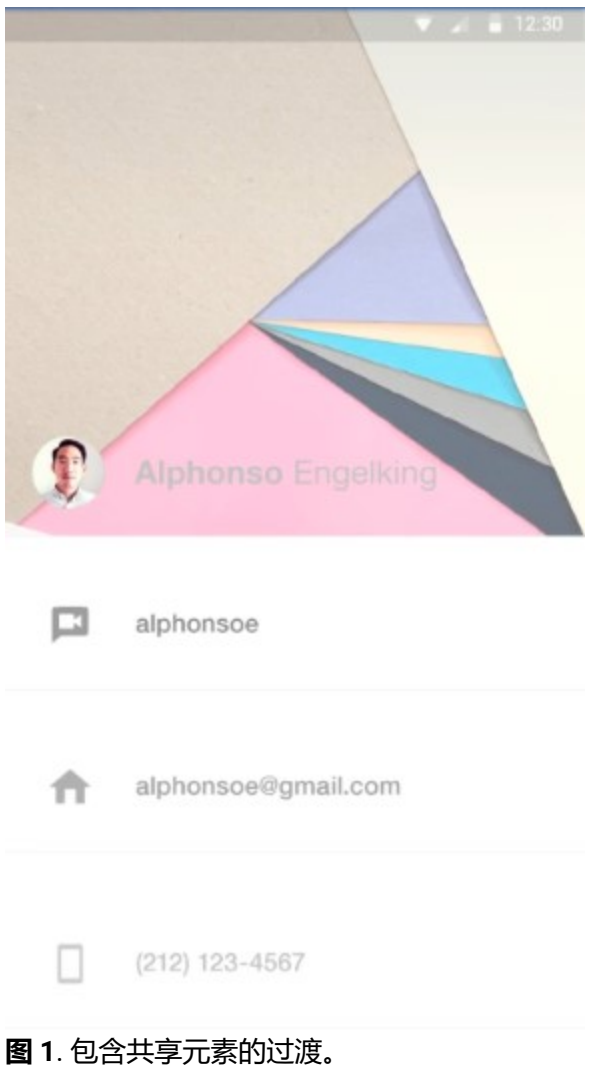

图 1. 包含共享元素的过渡。

- **进入**过渡决定了 Activity 中的视图如何进入场景。例如，在“爆炸式”进入过渡中，视图从外场进入场景，飞向屏幕中心。
- **退出**过渡决定了 Activity 中的视图如何退出场景。例如，在“爆炸式”退出过渡中，视图从屏幕中心离开场景。
- **共享元素**过渡决定了两个 Activity 共享的视图如何在这些 Activity 之间过渡。例如，如果两个 Activity 使用相同的图片（但位置和大小不同），changeImageTransform 共享元素过渡就会在这些 Activity 之间流畅地平移和缩放该图片。

Android 支持以下进入和退出过渡：

- 爆炸式 - 将视图移入场景中心或从中移出。
- 滑动式 - 将视图从场景的其中一个边缘移入或移出。
- 淡入淡出式 - 通过更改视图的不透明度，在场景中添加视图或从中移除视图。

系统支持将任何扩展 `Visibility` (/reference/android/transition/Visibility) 类的过渡作为进入或退出过渡。如需了解详情，请参阅 `Transition` (/reference/android/transition/Transition) 类的 API 参考文档。

Android 还支持以下共享元素过渡：

- changeBounds - 为目标视图布局边界的变化添加动画效果。
- changeClipBounds - 为目标视图裁剪边界的变化添加动画效果。
- changeTransform - 为目标视图缩放和旋转方面的变化添加动画效果。
- changeImageTransform - 为目标图片尺寸和缩放方面的变化添加动画效果。

当您在应用中启用 Activity 过渡后，系统会在进入和退出 Activity 之间激活默认的交替淡变过渡。

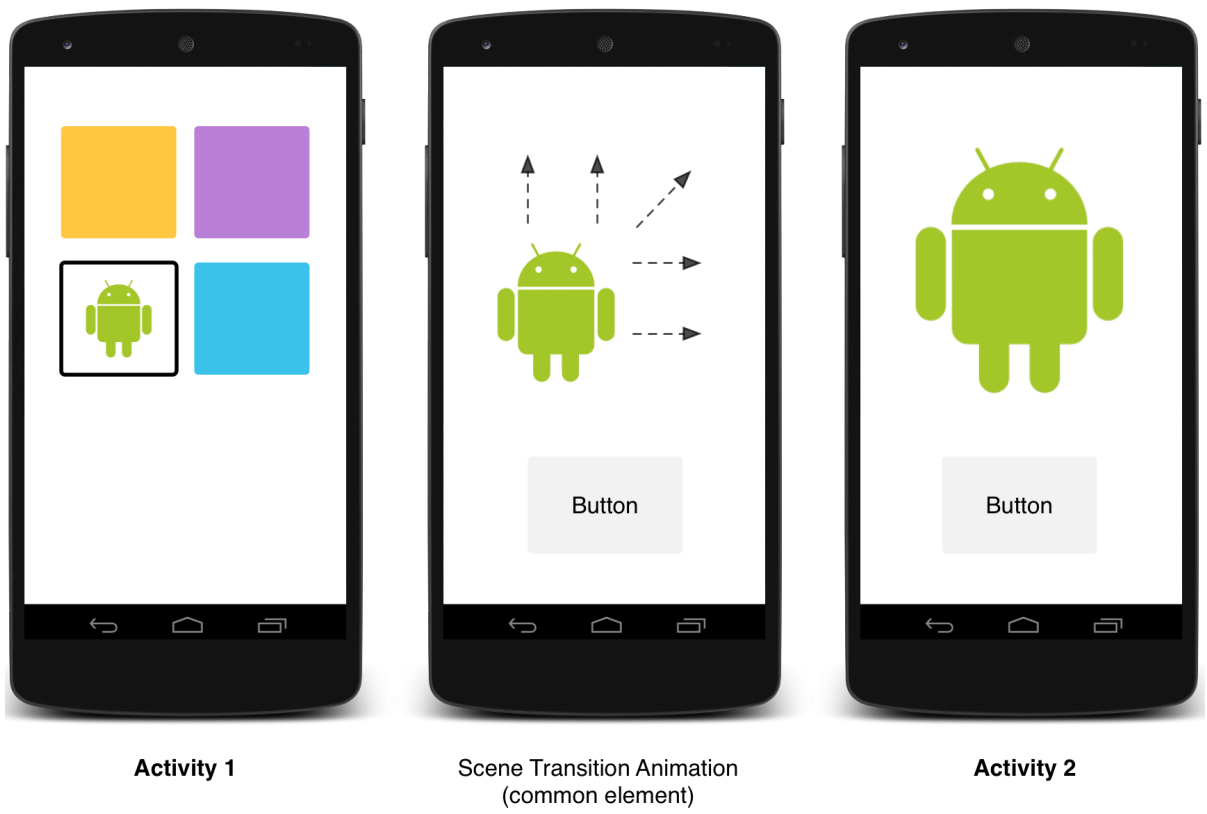

图 2 - 具有一个共享元素的场景过渡。

如需查看为使用共享元素的 Activity 添加动画效果的示例代码，请参阅 ActivitySceneTransitionBasic (https://github.com/android/animation/tree/master/ActivitySceneTransitionBasic)。

检查系统版本

Activity 过渡 API 适用于 Android 5.0 (API 21) 及更高版本。如需保持与更低版本的 Android 系统兼容，请先在运行时检查系统 `version` (/reference/android/os/Build.VERSION#SDK_INT)，然后再针对以下任何功能调用 API：

KOTLIN (#KOTLIN)JAVA

```
// Check if we're running on Android 5.0 or higher
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
    // Apply activity transition
} else {
    // Swap without transition
}
```

指定自定义过渡

首先，在定义从 Material 主题背景继承的样式时，使用 **android:windowActivityTransitions** 属性启用窗口内容过渡。您还可以在样式定义中指定进入、退出和共享元素过渡：

```
<style name="BaseAppTheme" parent="android:Theme.Material">
  <!-- enable window content transitions -->
  <item name="android:windowActivityTransitions">true</item>

  <!-- specify enter and exit transitions -->
  <item name="android:windowEnterTransition">@transition/explode</item>
  <item name="android:windowExitTransition">@transition/explode</item>

  <!-- specify shared element transitions -->
  <item name="android:windowSharedElementEnterTransition">
    @transition/change_image_transform</item>
  <item name="android:windowSharedElementExitTransition">
    @transition/change_image_transform</item>
</style>
```

此示例中的 **change_image_transform** 过渡定义如下：

```
<!-- res/transition/change_image_transform.xml -->
<!-- (see also Shared Transitions below) -->
<transitionSet xmlns:android="http://schemas.android.com/apk/res/android">
  <changeImageTransform/>
</transitionSet>
```

changeImageTransform 元素对应于 **ChangeImageTransform** (/reference/android/transition/ChangeImageTransform) 类。如需了解详情，请参阅 **Transition** (/reference/android/transition/Transition) 的 API 参考文档。

如需改为在代码中启用窗口内容过渡，请调用 **Window.requestFeature()** (/reference/android/view/Window#requestFeature(int)) 函数：

KOTLIN (#KOTLIN)JAVA

```
// inside your activity (if you did not enable transitions in your theme)
getWindow().requestFeature(Window.FEATURE_CONTENT_TRANSITIONS);

// set an exit transition
getWindow().setExitTransition(new Explode());
```

如需在代码中指定过渡，请使用 **Transition** (/reference/android/transition/Transition) 对象调用以下函数：

- **Window.setEnterTransition()** (/reference/android/view/Window#setEnterTransition(android.transition.Transition))
- **Window.setExitTransition()** (/reference/android/view/Window#setExitTransition(android.transition.Transition))
- **Window.setSharedElementEnterTransition()** (/reference/android/view/Window#setSharedElementEnterTransition(android.transition.Transition))
- **Window.setSharedElementExitTransition()** (/reference/android/view/Window#setSharedElementExitTransition(android.transition.Transition))

setExitTransition() (/reference/android/view/Window#setExitTransition(android.transition.Transition)) 和 **setSharedElementExitTransition()** (/reference/android/view/Window#setSharedElementExitTransition(android.transition.Transition)) 函数用于定义正在调用的 Activity 的退出过渡。**setEnterTransition()** (/reference/android/view/Window#setEnterTransition(android.transition.Transition)) 和 **setSharedElementEnterTransition()** (/reference/android/view/Window#setSharedElementEnterTransition(android.transition.Transition)) 函数用于定义之前调用的 Activity 的进入过渡。

如需获得完整的过渡效果，您必须针对正在调用的 Activity 和之前调用的 Activity **同时启用窗口内容过渡**。否则，正在调用的 Activity 会开始退出过渡，但您随后会看到窗口过渡（例如缩放或淡入淡出）。

如需尽快开始进入过渡，请针对之前调用的 Activity 使用 **Window.setAllowEnterTransitionOverlap()** (/reference/android/view/Window#setAllowEnterTransitionOverlap(boolean)) 函数。这样一来，您的进入过渡效果会更加精彩。

启动使用过渡的 Activity

如果您启用过渡并为某个 Activity 设置退出过渡，则当您启动另一个 Activity 时会激活此过渡，如下所示：

KOTLIN (#KOTLIN)JAVA

```
startActivity(intent,
              ActivityOptions.makeSceneTransitionAnimation(this).toBundle());
```

如果您为第二个 Activity 设置了进入过渡，当您启动该 Activity 时也会激活此过渡。如需在启动另一个 Activity 时停用过渡，请提供一个 **null** 选项包。

启动具有一个共享元素的 Activity

要在具有一个共享元素的两个 Activity 之间添加屏幕过渡动画，请执行以下操作：

1. 在主题背景中启用窗口内容过渡。
2. 在样式中指定共享元素过渡。
3. 将过渡定义为 XML 资源。
4. 使用 **`android:transitionName`** 属性为两个布局中的共享元素指定一个通用名称。
5. 使用 **`ActivityOptions.makeSceneTransitionAnimation()`** (/reference/android/app/ActivityOptions#makeSceneTransitionAnimation(android.app.Activity, android.util.Pair<android.view.View, java.lang.String>...)) 函数。

KOTLIN (#KOTLIN)**JAVA**

```
// get the element that receives the click event
final View imgContainerView = findViewById(R.id.img_container);

// get the common element for the transition in this activity
final View androidRobotView = findViewById(R.id.image_small);

// define a click listener
imgContainerView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(this, Activity2.class);
        // create the transition animation - the images in the layouts
        // of both activities are defined with android:transitionName="robot"
        ActivityOptions options = ActivityOptions
            .makeSceneTransitionAnimation(this, androidRobotView, "robot");
        // start the new activity
        startActivity(intent, options.toBundle());
    }
});
```

对于您在代码中生成的共享动态视图，请使用 **`View.setTransitionName()`** (/reference/android/view/View#setTransitionName(java.lang.String)) 函数以指定两个 Activity 中的通用元素名称。

如需在完成第二个 Activity 时反转场景过渡动画，请调用 **`Activity.finishAfterTransition()`** (/reference/android/app/Activity#finishAfterTransition()) 函数，而非 **`Activity.finish()`** (/reference/android/app/Activity#finish())。

启动具有多个共享元素的 Activity

如需在具有多个共享元素的两个 Activity 之间添加场景过渡动画，请使用 **`android:transitionName`** 属性（或者在两个 Activity 中使用 **`View.setTransitionName()`** (/reference/android/view/View#setTransitionName(java.lang.String)) 函数）定义两个布局中的共享元素，并创建一个 **`ActivityOptions`** (/reference/android/app/ActivityOptions) 对象，如下所示：

KOTLIN (#KOTLIN)**JAVA**

```
ActivityOptions options = ActivityOptions.makeSceneTransitionAnimation(this,
        Pair.create(view1, "agreedName1"),
        Pair.create(view2, "agreedName2"));
```


Last updated 2020-06-11 UTC.