1. **Android Animations with Examples**

In android, **Animations** are used to change the appearance and behavior of the objects over a particular interval of time. The animations will provide a better look and feel high quality user interface for our applications.

Generally, the animations are useful when we want to notify users about the change’s happening in our app, such as new content loaded or new actions available, etc.

We have a different type of animations available in android, here we will discuss about most commonly used android animations such as:

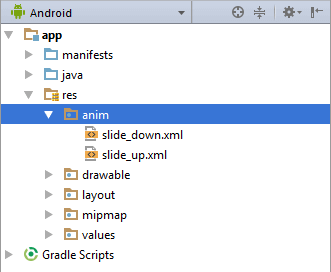
* [zoom in / zoom out](https://www.tutlane.com/tutorial/android/android-zoom-in-out-animations-with-examples),
* [fade in / fade out](https://www.tutlane.com/tutorial/android/android-fade-in-out-animations-with-examples),
* [slide up / slide down](https://www.tutlane.com/tutorial/android/android-slide-up-down-animations-with-examples) and
* [rotate clockwise or anti clockwise](https://www.tutlane.com/tutorial/android/android-rotate-animations-clockwise-anti-clockwise-with-examples), etc.

To create an animation effect to the objects in our android application, we need to follow below steps.

**Create XML File to Define Animation**

We need to create an xml file that defines the type of animation to perform in a new folder **anim** under **res** directory (**res** **anim**  **animation.xml**) with required properties. In case **anim** folder not exists in **res** directory, create a new one.

Following is the example of creating an XML files under **anim** folder to define [slide up / down](https://www.tutlane.com/tutorial/android/android-slide-up-down-animations-with-examples) animation properties.



The xml files will contain the code like as shown below based on the type of animation.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator">  
    <scale  
        android:duration="500"  
        android:fromXScale="1.0"  
        android:fromYScale="0.0"  
        android:toXScale="1.0"  
        android:toYScale="1.0" />  
</set>

In case if we want to use different type of animations such as [fade in / out](https://www.tutlane.com/tutorial/android/android-fade-in-out-animations-with-examples), [zoom in / out](https://www.tutlane.com/tutorial/android/android-zoom-in-out-animations-with-examples), etc. we need to create a new xml files in **anim** folder with required properties.

Following are the some of important animation attributes which will help us to change the behaviour of animation in our application.

| **Attributes** | **Description** |
| --- | --- |
| android:duration | It is used to define the duration for the animation to complete. |
| android:startOffset | It is used to define the waiting time before an animation starts. |
| android:interpolator | It is used to define the rate of change in animation. |
| android:repeatMode | It is useful when we want to repeat our animation. |
| android:repeatCount | It is used to define the number of times the animation repeats. In case if we set **infinite**, the animation will repeat infinite times. |
| android:fillAfter | It is used to define whether to apply animation transformation after the animation completes or not. |

**Android Load and Start the Animation**

In android, we can perform animations by using **AnimationUtils** component methods such as **loadAnimation()**. Following is the code snippet of loading and starting an animation using **loadAnimation()** and **startAnimation()**methods.

ImageView img = (ImageView)findViewById(R.id.imgvw);

Animation aniSlide = AnimationUtils.loadAnimation(getApplicationContext(),R.anim.slide\_up);  
img.startAnimation(aniSlide);

If we observe above code snippet, we are adding an animation to the image using **loadAnimation()** method. The second parameter in **loadAnimation()** method is the name of our animation xml file.

Here we used another method **startAnimation()** to apply the defined animation to **imageview** object.

**Different Types of Android Animations**

In android, we have a different type of animations such as [Fade In / Fade Out](https://www.tutlane.com/tutorial/android/android-fade-in-out-animations-with-examples), [Zoom In / Zoom Out](https://www.tutlane.com/tutorial/android/android-zoom-in-out-animations-with-examples), [Slide Up / Slide Down](https://www.tutlane.com/tutorial/android/android-slide-up-down-animations-with-examples), [Rotate in Clockwise or Anti Clockwise](https://www.tutlane.com/tutorial/android/android-rotate-animations-clockwise-anti-clockwise-with-examples), etc.

Now we will see how to create each animation with required properties in android application.

1. **Android Fade In / Out Animation**

To use [Fade In or Fade Out](https://www.tutlane.com/tutorial/android/android-fade-in-out-animations-with-examples) animations in our android applications, we need to define a new xml file with **<alpha>** tag like as shown below.

For **Fade In** animation, we need to increase the **alpha** value from **0** to **1** like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator">  
   <alpha  
        android:duration="2000"  
        android:fromAlpha="0.1"  
        android:toAlpha="1.0">  
    </alpha>  
</set>

For **Fade Out** animation, we need to decrease the alpha value from 1 to 0 like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"  
    android:interpolator="@android:anim/linear\_interpolator">  
    <alpha  
        android:duration="2000"  
        android:fromAlpha="1.0"  
        android:toAlpha="0.1" >  
    </alpha>  
</set>

1. **Android Slide Up / Down Animation**

To use [Slide Up or Slide Down](https://www.tutlane.com/tutorial/android/android-slide-up-down-animations-with-examples) animations in our android applications, we need to define a new xml file with **<scale>** tag like as shown below.

For **Slide Up** animation, we need to set android:fromYScale="1.0" and android:toYScale="0.0" like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android" android:interpolator="@android:anim/linear\_interpolator">  
    <scale  
        android:duration="500"  
        android:fromXScale="1.0"  
        android:fromYScale="1.0"  
        android:toXScale="1.0"  
        android:toYScale="0.0" />  
</set>

For **Slide Down** animation, we need to set android:fromYScale="0.0" and android:toYScale="1.0"  like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator">  
    <scale  
        android:duration="500"  
        android:fromXScale="1.0"  
        android:fromYScale="0.0"  
        android:toXScale="1.0"  
        android:toYScale="1.0" />  
</set>

1. **Android Zoom In / Out Animation**

To use [Zoom In or Zoom Out](https://www.tutlane.com/tutorial/android/android-zoom-in-out-animations-with-examples) animations in our android applications, we need to define a new xml file with **<scale>** tag like as shown below.

For **Zoom In** animation, we need to set android:pivotX="50%" and android:pivotY="50%" to perform the zoom from centre of the element. Also we need to use fromXScale, fromYScale attributes to define the scaling of object and we need keep these value lesser than toXScale, toYScale like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
    <scale  
        xmlns:android="http://schemas.android.com/apk/res/android"  
        android:duration="1000"  
        android:fromXScale="2"  
        android:fromYScale="2"  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:toXScale="4"  
        android:toYScale="4" >  
    </scale>  
</set>

In android,**Zoom** **Out** animation is same as **Zoom In** animation but fromXScale, fromYScale

attribute values must be greater than to XScale, to YScale like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
    <scale  
        android:duration="2500"  
        android:fromXScale="1.0"  
        android:fromYScale="1.0"  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:toXScale=".2"  
        android:toYScale=".2" />  
</set>

1. **Android Rotate Clockwise / Anti Clockwise Animation**

To use [Rotate animation](https://www.tutlane.com/tutorial/android/android-rotate-animations-clockwise-anti-clockwise-with-examples) in our android applications, we need to define a new xml file with **<rotate>** tag like as shown below.

To **Rotate** animation in **Clockwise**, we need to set android:fromDegrees and android:toDegrees

property values and these will define a rotation angles like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android" android:interpolator="@android:anim/cycle\_interpolator">  
    <rotate android:fromDegrees="0"  
        android:toDegrees="360"  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:duration="5000" />  
</set>

To **Rotate** animation in **Anti Clockwise**, we need to set android:fromDegrees and

android:toDegrees property values and these will define a rotation angles like as shown below.

<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/cycle\_interpolator">  
    <rotate android:fromDegrees="360"  
        android:toDegrees="0"  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:duration="5000" />  
</set>

This is how we can use different types of animations in android applications based on our requirements.