Android Animation Techniques

# Introduction:

In the Android framework there are several animation techniques which can be used for both 2D as well as 3D graphics. The choice on which technique to use greatly depends on what we are trying to achieve since all of them have their pros and cons.

There are three main systems used in Android to create animation:

* Property Animation
* View Animation
* Drawable Animation

Along with the animation techniques there are frameworks which allow us to create 2D and 3D graphics in order to be used for our animations. The most common method is using Canvas and Drawables, which is powerful enough for most animation needs; there is also an alternative of using the OPEN GL ES framework which allows us to create more sophisticated and powerful graphics.

# Property Animation:

Property Animation focuses on the object properties in order to animate it. That can involve changing its X and Y coordinates in order to move it around the screen, altering its size properties to scale or rotate 2D or even 3D object by choosing a pivot point. We can also change alpha transparency of objects, which is useful to fade objects in and out of the view.

All of this is performed based on the duration of the animation; we can choose what kind of movement we want, over a specific timeframe. Longer duration or smaller change in properties will result in a slower animation, where as shorter animation duration or bigger change in properties over a timeframe will result in a quicker animation.

The animation interpolation can be linear meaning the movement will be the same within the duration of an animation (for example; change X coordinate from 0 to 50 over 50ms, 10ms being in increase of X coordinate by 10) resulting in a smooth consistent animation, or non-linear meaning the animation will change speed as the change in properties will not be constant within a time frame. We can even make the animation stop and resume after a given timeframe. All of this is possible due to the use of the “TimeInterpolators” provided by android or the use of Key frames which we can create and set up ourselves. Keyframes are a tool commonly used in filmmaking and animation, Keyframe is also a class used by Value Animator.

Property animation can be declared in both XML as well as programmatically. The main advantage of doing it in XML is the fact that we can reuse the code for more than one object or more than one activity, synchronising the animation sequence is also easier using XML.

# View Animation:

View animation, also referred to as “Tween Animation” is an older animation system implemented in the Android API. Just like property animation it allows us to animate view objects by moving them around, scaling, rotating and even changing the alpha transparency settings. The main disadvantage of using View Animation is the fact that it only allows us to work with view objects, whereas property animation allows us to work with any object.

Just like property animation we have the option to animate objects using XML or programmatically, with XML being the preferred option due to the ability to reuse code for more than one object; it is also a much cleaner, more readable implementation.

The speed of an animation is defined by assigning an Interpolar, the same technique which I have covered in the Property Animation.

The main problem with view animation compared to property animation is the fact that if we wanted to animate a button to move across the screen, visually it would do exactly what we ask it to do, but the physical button which we can click that provides functionality would stay in the original button location. Therefore if we wanted to animate a button using a view animation, it would no longer work when clicked and in order to move the functionality of the button to make it clickable again we would have to code that separately.

View animation is a very basic technique and it doesn’t require much code to write, it’s a good option when working with basic animation in views, where it’s flaws or inability to animate objects outside of view are not a problem.

# Drawable Animation:

Drawable animation is a technique where all of the images are loaded in and then played in sequence in order to create movement. It is done via the *AnimationDrawable* class.

Drawable animation is a very common technique for animation; it allows the use of spritesheets. Spritesheet is a series of images which are all in a slightly different positions, so when played back quickly in series, they appear to be moving, it’s just like having individual frames of a movie which are played back quickly to create movement. Spritesheets have been in most classic games and are still used today in modern games, for example a very successful game “Cut the rope” is also based on drawable animation using spritesheets.

Drawable Animation offers a lot of flexibility, the smoothness of animation can be chosen by deciding how many images we would like to use for our movement, the more we use, the smoother the animation and it will allow us to play it back slower without being able to notice the transition, however it will mean we will have to implement more individual images. It’s a bit like frame rate, the more we have the smoother it is, however it will also require more computational power in order to be played back.

# 2D and 3D Graphics:

The android framework provides a variety of drawing APIs for 2D graphics which allow a user to generate and animate custom graphics for their application needs. There are two main ways of how graphics is done on the android frameworks, drawing graphics using the View object in the layout section or drawing graphics onto the Canvas object. Android framework also has a Drawable library which provides us with a range of tools for drawing shapes and graphics.

## Canvas & Drawable:

Canvas and Drawables are the most sensible option for creating custom 2D graphics for an android game, it’s perfect for rapidly changing animation unlike drawing straight on top of View. Graphics can be either drawn using the tools provided by the Drawable library or imported from our project resources, therefore techniques such as the use of sprites can be utilised.

The supported file types are PNG, JPG or GIF, however PNG is the proffered file type for the drawable library. When images are imported from project resource files, the images are automatically optimized using a lossless compression in order to reduce their size.

Drawable class has a useful subclass called ShapeDrawable which is a handy tool to create two dimensional graphics of primitive shapes. We can define and customise the size, position as well as colour properties and get our shape drawn onto the View.  
An alternative to this is using vector drawables which are defined by a set of points, lines or curves which are also an option in the Drawable library.

## OpenGL ES:

OpenGL ES is a library in the Android framework which allows creating sophisticated 2D and 3D graphics which can be used for animation. The objects are formed through the use of matrices with a set of coordinates in order to define a shape. OpenGL has a very powerful toolset however it is very complex and requires a lot of coding in order to come up with 3D graphics of high quality, for my game I will most likely be focusing on the use of Canvas as well as drawables.

# Sources:

<https://www.codeproject.com/Articles/825700/Beginners-Guide-to-Android-Animation-Graphics#opengles>

<https://developer.android.com/guide/topics/graphics/overview.html>

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

<https://developer.android.com/guide/topics/graphics/view-animation.html>

<https://developer.android.com/guide/topics/graphics/drawable-animation.html>

<https://android-developers.googleblog.com/2011/05/introducing-viewpropertyanimator.html>

<https://developer.android.com/guide/topics/graphics/opengl.html>