Animation and Graphics Overview

Android provides a variety of powerful APIs for applying animation to UI elements and drawing custom 2D and 3D graphics. The sections below provide an overview of the APIs and system capabilities available and help you decide with approach is best for your needs.

Animation

The Android framework provides two animation systems: property animation and view animation. Both animation systems are viable options, but the property animation system, in general, is the preferred method to use, because it is more flexible and offers more features. In addition to these two systems, you can utilize Drawable animation, which allows you to load drawable resources and display them one frame after another.

Property Animation

Introduced in Android 3.0 (API level 11), the property animation system lets you animate properties of any object, including ones that are not rendered to the screen. The system is extensible and lets you animate properties of custom types as well.

View Animation

View Animation is the older system and can only be used for Views. It is relatively easy to setup and offers enough capabilities to meet many application's needs.

Drawable Animation

Drawable animation involves displaying `Drawable` resources one after another, like a roll of film. This method of animation is useful if you want to animate things that are easier to represent with Drawable resources, such as a progression of bitmaps.

2D and 3D Graphics

When writing an application, it's important to consider exactly what your graphical demands will be. Varying graphical tasks are best accomplished with varying techniques. For example, graphics and animations for a rather static application should be implemented much differently than graphics and animations for an interactive game. Here, we'll discuss a few of the options you have for drawing graphics on Android and which tasks they're best suited for.

Canvas and Drawables

Android provides a set of `View` widgets that provide general functionality for a wide array of user interfaces. You can also extend these widgets to modify the way they look or behave. In addition, you can do your own custom 2D rendering using the various drawing methods contained in the `Canvas` class or create `Drawable` objects for things such as textured buttons or frame-by-frame animations.

Hardware Acceleration

Beginning in Android 3.0, you can hardware accelerate the majority of the drawing done by the Canvas APIs to further increase their performance.

OpenGL

Android supports OpenGL ES 1.0 and 2.0, with Android framework APIs as well as natively with the Native Development Kit (NDK). Using the framework APIs is desireable when you want to add a few graphical enhancements to your application that are not supported with the Canvas APIs, or if you desire platform independence and don't demand high performance. There is a performance hit in using the framework APIs compared to the NDK, so for many graphic intensive applications such as games, using the NDK is beneficial (It is

important to note though that you can still get adequate performance using the framework APIs. For example, the Google Body app is developed entirely using the framework APIs). OpenGL with the NDK is also useful if you have a lot of native code that you want to port over to Android. For more information about using the NDK, read the docs in the `docs/` directory of the NDK download.