Plus One

Section 4

プロパティアニメーションでアプリの表現力を豊かに！

安達 正

開発対象
Android 3.1
(API Level 12)

Android 3.0から「プロパティアニメーション」という新しいアニメーションシステムがサポートされました。本稿では、前半でプロパティアニメーションの概要と APIについて解説し、後半では実践編としてプロパティアニメーションを使用してビューをアニメーションさせる方法、及びビュー以外のオブジェクトをアニメーションさせる方法について解説します。

1 プロパティアニメーションの概要

Android のアニメーションシステム

Androidフレームワークには、Android 1.0（API Level 1）からサポートされているビューアニメーション（View Animation）システムと、Android 3.0（API Level 11）からサポートされたプロパティアニメーション（Property Animation）システムがあります。

ビューアニメーションシステム

ビューアニメーションシステムとは、ビューに対してのみアニメーションを実行させることができるアニメーションシステムです。アニメーションは、アニメーション実行中のビューの描画に対してのみ反映されます。主なAPIは、`android.view.animation`パッケージで提供されています。ビューの透過度を変化させるアルファアニメーションや、ビューを移動、拡大縮小、回転させるアニメーションなどを実現できます。

プロパティアニメーションシステム

プロパティアニメーションシステムとは、任意のオブジェクトのプロパティの値を変化させることでアニメーションを実現するシステムです。ビューアニメーションのようにビューをアニメーションさせることができるのは当然のこと、任意のオブジェクトに対してもアニメーションさせること

ができます。主なAPIは、Android 3.0で追加された`android.animation`パッケージで提供されています。また、Android 3.1（API Level 12）では`android.view`パッケージにも`ViewPropertyAnimator`クラスが追加されています。

プロパティアニメーションの仕組み

図1は、画面内のボタンの表示位置（x）を0から100へ、1000ミリ秒間かけて移動させるアニメーションを250ミリ秒間隔で画面キャプチャしたものです。

図1　ボタンを移動させるアニメーションの画面キャプチャ

図1からわかるとおり、ボタンを移動させるアニメーションでは、時間の経過とともに、画面内のボタンの表示位置が変化する視覚的な変化だけでなく、ボタン自身に着目してみると、ボタンの属性である位置情報の値が変化していることがわかります。逆に言えば、ボタンの属性である位置情報の値を変化させれば、ボタンをアニメーションさせることができるとも言えます。この時、プロパティアニメーションでは、ボタンのことをオブジェクト、位置情報などのフィールドをプロパティ、1枚1枚の静止画像をフレームと呼びます。

リスト1は、このボタンのアニメーションを、プロパティアニメーションAPIを使用して実装したコードです。このように、プロパティアニメーションでは、変化させたいオブジェクト、プロパティ、値を指定することで、アニメーションを表現します。

リスト1　プロパティアニメーションAPIを使用してボタンを移動させる

```
Button button = (Button) findViewById(R.id.button);
// 指定したオブジェクトのプロパティを変化させる
ObjectAnimator animator = ObjectAnimator.ofFloat(
    button, "x",  // アニメーションの対象オブジェクト（ボタン）、プロパティ名（x）
    0f, 100f); // 変化させたい値（0から100へ）
animator.setDuration(1000); // アニメーションの継続時間（1000ミリ秒）を設定する
animator.start(); // アニメーションを開始する
```

2 プロパティアニメーション API

Animator クラス

Animator クラスは、プロパティアニメーションを操作するための基本的なメソッドを定義したスーパークラスです。ただし、Animator は抽象クラスとして定義されているため、直接インスタンス化することはできません。使用するには、図2に示したような Animator のサブクラスをインスタンス化して使用します。

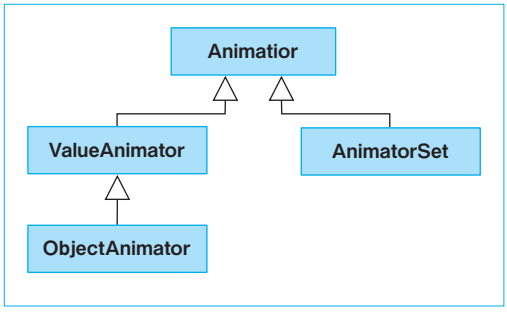

図2　Animator クラス継承図

Animator クラスで定義されている主なメソッドは、表1のとおりです。これらのメソッドは、後述する ValueAnimator、ObjectAnimator、AnimatorSet クラスで使用します。

表1　Animator クラスの主なメソッド

メソッド名	説明
start	アニメーションを開始します。
cancel	アニメーションをキャンセルします。
setDuration	アニメーション継続時間を設定します（単位はミリ秒）。
setInterpolator	アニメーション補間関数を設定します。
addListener	アニメーション中のイベントを受け取るリスナを追加します。

Animator.AnimatorListener インタフェース

Animator.AnimatorListener インタフェースは、アニメーションの開始や終了などのアニメーション中のイベントを受け取るためのインタフェースです。メソッドは**表2**のとおりです。

表2　Animator.AnimatorListenerインタフェースのメソッド

メソッド名	説明
onAnimationStart	アニメーション開始時に呼び出されます。
onAnimationEnd	アニメーション終了時に呼び出されます。
onAnimationCancel	アニメーションキャンセル時に呼び出されます。
onAnimationRepeat	アニメーションの繰り返し時に呼び出されます。

AnimatorListenerAdapter クラス

AnimatorListenerAdapter クラスは、Animator.AnimatorListener インタフェースで実装する必要があるすべてのメソッドを、何もしない処理で実装したクラスです。このクラスを使用すると、新たに実装するリスナでは、必要なイベントのメソッド実装のみで済むため、不要なイベントのメソッド実装を省略できます。

リスト2は、アニメーション終了時のイベントを受け取り、処理するコードです。

リスト2　アニメーション終了時のイベントを受け取る

```
// ボタンを移動するアニメーション
final Button button = (Button) findViewById(R.id.button);
// 指定したオブジェクトのプロパティを変化させる
ObjectAnimator animator = ObjectAnimator.ofFloat(button, "x", 0f, 100f);
animator.setDuration(1000);
// アニメーション中のイベントを受け取るリスナを追加する
animator.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        // ここに必要な処理を記述する
        // ここでは、アニメーション終了後、ボタンを非表示に設定している
        button.setVisibility(View.GONE);
    }
});
animator.start();// アニメーションを開始する
```