**ViewPropertyAnimation 클래스**

Class Overview

이 클래스는 뷰 오브젝트의 선택된 프로퍼티들에 대한 애니매이션을 자동적으로 최적화 한다. 만약 하나나 두개의 프로퍼티가 animate 될거라면, ‘ObjectAnimator’가 좋다. ObjectAnimator가 부르는 property setter는 적절하게 property를 set하고 view를 업데이트한다. 그러나 몇가지 프로퍼티들을 동시에 animate하고 싶거나, 특정 프로퍼티를 animate하는데 더 편하게 코딩하고 싶으면, ViewPropertyAnimator가 더 좋다.

이 클래스는 동시에 이루어지는 애니매이션에서 더 좋은 성능을 보여준다. 왜냐면 각 프로퍼티들이 저마다의 inavlidation을 하는게 아니라 한번의 invalidation을 하도록 최적화하기 때문이다. 또한, 코딩하기 쉽다., 왜냐면 호출하는 쪽은 View 오브젝트에게 어떤 프로퍼티를 animate 할 건지만 말해주면 되고, 이 클래스는 근본적인 Animator 클래스를 세밀하게 구성하고 있기 때문이다.

이 클래스는 caller에 의해 만들어지지 않는다. animate될 프로퍼티를 가지는 View가 만든다. animate()를 호출하면 적절한 ViewPropertyAnimator 오브젝트의 레퍼런스를 해당 뷰에 리턴한다.

**Introducing ViewPropertyAnimator**

지난번의 ‘Animation in Honeycomb’ 에서 프로퍼티 애니매이션에 대해서 얘기했다. 이것은 어떤 오브젝트의 어떤 프로퍼티라도 animate 할 수 있는 거였다.

먼저, 만약 View의 새로운 속성인 alpha, translationX 같은 게 친숙하지 않다면, 이전의 글을 보셈.

**Refresher: Using ‘ObjectAnimator’**

ObjectAnimaotr 클래스를 이용하면 뷰 프로퍼티중에 하나를 간단하게 animate시킬수 있었다. Animator를 만들고, duration이나 반복설정을 하고 start 하면 됐었다. 예를 들어. myView라는 오브젝트를 fade out 시키고 싶으면 alpha 프로퍼티를 아래와 같이 하면 됐었다.

ObjectAnimator.ofFloat(myView, "alpha", 0f).start();

완전 쉽다.

하지만 더 좋아질 수 있을 것 같다. 특히, View 속성은 너무 일반적으로 animate된다. 더 멋지게 만들었고 성능도 좋아졌다.

세가지가 더 좋아졌다. animation을 코딩할때 ‘property’라는 개념이 존재하지 않는다는게 향상되었고, 여러 속성을 동시에 animate 하는거, 각 프로퍼티마다 독립적인 situation을 갖는다는 거.

안드로이드는 프로퍼티에 대한 개념이 없다. 그래서 ObjectAnimator는 프로퍼티를 나타내는 문자열을 target object의 setter function을 의미하도록 하는 기능을 이용했다. 예를 들어, ‘alpha’라는 문자열을 setAlpha()메서드로 변하는 거다. 이거는 JNI가 하는데 오버헤드가 있다. 당근 더 좋아질 필요가 있다. 주어진 API에 의해서 우리는 JNI에 의한 오버헤드 없이 간단하게 오브젝트의 여러 값들을 직접적으로 설정할 수 있다.

다른 오버헤드는 Animator 그 자신이다. 모든 animation들이 하나의 timing machanism을 공유하니까 timing events를 처리하는 오버헤드가 누적되면 안되는데도, 얘네는 프로퍼티를 처리하는 같은 일을 각각이 하고 있다. 이 task들은 합쳐져야 된다. 기존의 시스템에서 이를 실현하기 위한 한 방법은 ‘PropertyValuesHolder를 이용하는 것이다. 이 클래스는 Animator 오브젝트를 하나만 가지면서도 여러 프로퍼티들을 함께 animate 시킬 수 있게 해준다. 근데 코드가 많아진다. 앞으로 소개할 새로운 접근방법은 간단하게 이를 도와준다.

마지막으로, 프로퍼티들은 그 부모와 오브젝트의 업데이트(invalidate)를 위해 몇가지 동작을 수행한다. 예를 들어**, 뷰를 X축으로 이동시키는 것은 이전에 점유했던 포지션과 지금 점유하고 있는 포지션을 모두 invalidate한다. Y도 유사하다. 만약 두 프로퍼티가 병렬적으로 animate된다면, 수고가 중복된다. ViewPropertyAnimator가 이것을 커버한다.**

**Introducing: ViewPropertyAnimator**

ViewPropertyAnimator는 여러 프로퍼티를 하나의 Animator 안에서 동시에 animate하는 간단한 방법을 제공한다. 그리고 animate될 프로퍼티의 값을 계산하는 동시에 target View에 이를 적용시키고 inavlidate 한다. 일반적인 ObjectAnimator 보다 더 효율적이다.

옛날 : ObjectAnimator.ofFloat(myView, "alpha", 0f).start();

지금 : myView.animate().alpha(0);

짧고 읽기 쉽다. 또한 다른 프로퍼티 애니매이션과 합치기도 쉽다. 예를 들어 x랑 y랑 둘다 animate 시키고 싶으면 아래와 같이 하면 된다.

myView.animate().x(500).y(500);

설명->

- animate() : ViewPropertyAnimator의 인스턴스를 리턴한다.

- Auto-start : start()를 하지 않았다. 새로운 API에서는 애니매이션을 시작하는 것이 암시적이다. 선언하자마자 애니매이션이 시작된다. 다음 업데이트가 있을때까지 기다린다. ViewPropertyAnimator는 모든 선언된 애니매이션을 collenting한다는 뜻이다. 선언을 유지하는 동안, event queue에 애니매이션들을 모으고 다음 프레임이 이를 실행시킨다.

- Fluent : ViewPropertyAnimator는 좋은 인터페이스를 갖는다. 여러 프로퍼티의 애니매이션을 한줄로 표현할 수 있으니깐.. X(), y()의 호출은 ViewPropertyAnimator의 인스턴트를 호출한다. 다른 메서드들과 엮일 수 있도록.

Perfomance Anxiety

성능에서 좋아진건 JNI같은거 안쓰고 View 속성을 직접적으로 건드린다는 거. 그리고 동시에 여러 프로퍼티를 엄청 쉽게 animate 하는거 .

ㅇㅖ)

옛날

ObjectAnimator animX = ObjectAnimator.ofFloat(myView, "x", 50f);

    ObjectAnimator animY = ObjectAnimator.ofFloat(myView, "y", 100f);

    AnimatorSet animSetXY = new AnimatorSet();

    animSetXY.playTogether(animX, animY);

    animSetXY.start();

쪼끔 개선된거 (holder 쓴거)

PropertyValuesHolder pvhX = PropertyValuesHolder.ofFloat("x", 50f);

    PropertyValuesHolder pvhY = PropertyValuesHolder.ofFloat("y", 100f);

    ObjectAnimator.ofPropertyValuesHolder(myView, pvhX, pvyY).start();

지금(ViewPropertyAnimator)

myView.animate().x(50f).y(100f);

**내부적으로 퍼포먼스 측면에서 좋은건, invalidate를 한 프레임당 한번만 한다는 것이다. 예전에는 setX()가 뷰를 invalidate, setY()가 뷰를 invalidate 각각 했었다. 근데 이제는 한 프레임당 target View를 한번만 invalidate한다.**

android.animation 패키지

이 클래스들은 프로퍼티 애니매이션 시스템 기능을 제공한다. 프로퍼티 애니매이션은 모든 타입의 프로퍼티를 animate하게 해주는 것이다. int, float등이 디폴트로 지원된다. ‘TypeEvaluator’를 이용해서 시스템에게 어떤 값을 계산할건지만 알려주면 어떤 타입이든지 animate 할 수 있다.

여러 종류의 인터폴레이터를 지정할 수 있고, keyframes를 지정할 수 있고, 여러 애니매이션을 순차적으로, 혹은 동시에 실행할 수 있다.

\*\* 프로퍼티 애니매이션은 눈에 보이는거 말고 다른 속성들을 animate하게 하는거. 커스텀 타입도 됨.

================================================================