**TimeAnimator**

Класс **TimeAnimator** появился в API 16. Класс предоставляет простой механизм обратного вызова слушателей, которые синхронизируются со всеми другими анимациями в системе. У него не задаётся длительность, интерполяция, или значение объекта-параметр как у других объектов **Animator**. Вместо этого, он просто запускается, после чего отправляет события на каждом кадре анимации к его слушателю **TimeListener** (если установлен), с информацией об анимации, общей продолжительности времени, и времени, прошедшее с предыдущего кадра анимации.

Сам по себе **TimeAnimator** ничего не анимирует. Его задача с помощью слушателя **TimeListener** обновить данные при каждом изменении анимации. Имеет методы.

* **void setCurrentPlayTime (long playTime)**
* **void setTimeListener(TimeAnimator.TimeListener listener)**
* **void start()**

Метод **setTimeListener()** регистрирует слушатель. Метод **start()** запускает анимацию.

**TimeListener** является интерфейсом с методом:

void onTimeUpdate(TimeAnimator animation, long totalTime, long deltaTime)

Первый аргумент указывает на нужную анимацию, если их будет несколько. Второй аргумент отвечает за общее время анимации в миллисекундах. Третий аргумент отвечает за время в миллисекундах, прошедшее с последнего обновления анимации.

Рассмотрим пример с **TimeAnimator**. Мы будем выводить информацию о значении **fps** (frames per second) во время выполнения другой анимации в компоненте **TextView**.

Создадим разметку для примера из изображения, текстовой метки и кнопки.

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<Button

android:id="@+id/buttonStart"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Start"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"/>

<ImageView

android:id="@+id/imageView"

android:layout\_width="150dp"

android:layout\_height="150dp"

android:onClick="onClick"

android:src="@mipmap/ic\_launcher"

android:layout\_below="@+id/buttonStart"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="55dp"/>

<TextView

android:id="@+id/textViewFps"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_below="@id/imageView" />

</RelativeLayout>

Создадим новый класс на основе **TimeListener**. Класс будет содержать два поля: переменную типа **double** для хранения значения **fps** и ссылку на **TextView**.

package ru.alexanderklimov.as132;

import android.animation.TimeAnimator;

import android.animation.TimeAnimator.TimeListener;

import android.annotation.TargetApi;

import android.os.Build;

import android.widget.TextView;

@TargetApi(Build.VERSION\_CODES.JELLY\_BEAN)

class FpsTimeListener implements TimeListener {

private double fps;

private TextView textView;

public FpsTimeListener(TextView textView) {

this.textView = textView;

this.fps = -1.0;

}

public void onTimeUpdate(TimeAnimator animation, long totalTime, long deltaTime) {

double currentFps;

if (deltaTime != 0)

currentFps = 1000.0 / (double) deltaTime;

else

currentFps = 0.9 \* fps;

if (fps < 0.0)

fps = currentFps;

else

fps = 0.9 \* fps + 0.1 \* currentFps;

textView.setText(String.format("fps: %.2f", fps));

}

}

Конструктор ссылается на текстовую метку и инициализирует значение **fps** отрицательным числом. В методе класса идёт вычисление **fps**.

Переходим в класс активности и пишем код

package ru.alexanderklimov.as132;

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.animation.AnimatorSet;

import android.animation.ObjectAnimator;

import android.animation.TimeAnimator;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.ImageView;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

private AnimatorSet mAnimatorSet;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

if (android.os.Build.VERSION.SDK\_INT >= 19)

{

ImageView someImageView = (ImageView) findViewById(R.id.imageView);

ObjectAnimator rotateAnimator = ObjectAnimator.ofFloat(someImageView, "rotation", 0, 360);

rotateAnimator.setDuration(1000);

rotateAnimator.setRepeatCount(3);

rotateAnimator.setRepeatMode(ObjectAnimator.RESTART);

TextView fpsTextView = (TextView) findViewById(R.id.textViewFps);

FpsTimeListener listener = new FpsTimeListener(fpsTextView);

final TimeAnimator timeAnimator = new TimeAnimator();

timeAnimator.setTimeListener(listener);

mAnimatorSet = new AnimatorSet();

mAnimatorSet.play(rotateAnimator).with(timeAnimator);

rotateAnimator.addListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

timeAnimator.end();

}

});

}

}

public void onClick(View view) {

mAnimatorSet.start();

}

}

При создании активности подготавливаем анимацию вращения вокруг своей оси три раза в течение одной секунды за оборот. И добавляем слушатель, который присоединяется к объекту **TimeAnimator**. Затем **AnimatorSet** объединяет две анимации **ObjectAnimator** и **TimeAnimator**. При нажатии на кнопку запускается анимация вращения картинки.

**TimeAnimator** следует остановить вручную, иначе анимация будет бесконечной. Для этого добавлен слушатель с методом **onAnimationEnd()**, в котором происходит остановка.

На основе статьи [Android Property Animations: The TimeAnimator](http://cogitolearning.co.uk/?p=1555)