あみゅーの(･´ｰ･｀)どや

(╬ °◡°)

2015-05
21

TypeEvaluator を使った簡単な ValueAnimator

Android

TypeEvaluatorがかなり便利だったのでソレについて

TypeEvaluatorとは

TypeEvaluator | Android Developers
コレです。

簡単に書くと...

```
ValueAnimator.ofFloat(0.f,1.f);
```

にしたValueAnimatorはセットしたDurationの間に0.fから1.fまでの値を受け取ることが出来ますね。
例えば上の状態だとsetDuration(1000)の場合、10msの時getAnimatorValuesで0.01fが取得できるみたいな
(Interpolatorがy=xな時)
コレが、単一の数字同士なら楽なんですが、例えば

```
float[][] hoge
```

な多次元配列のものを引数に渡し、いい感じに値を受け取りたいという時にTypeEvaluatorが活躍します。
例えば

```
float[][] hoge = {{0,0},{10,10},{100,100}};
float[][] hoge2 = {{0,0},{100,100},{0,0}};
```

で1000msの間にhogeからhoge2に値を移動したいValueAnimatorの場合、提供されている
ValueAnimator.of~ では対応できません。
(500msの時に{{0,0},{55,55},{50,50}}な値を取得したい)
ので、TypeEvaluatorをそこの中身の処理を書くといった感じになっています。

とりあえず、使い方とかコード見れば即わかりすると思うのでサンプルをどうぞ。

```
import android.animation.TypeEvaluator;
import android.animation.ValueAnimator;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.View;
import android.view.animation.AccelerateDecelerateInterpolator;

/**
 * Created by amyu on 15/05/20
 */
public class CubicToSample extends View {

    private Paint mPaint;

    private int mWidth;

    private Path mWavePath;

    private static final float[][] WAVE_CURRENT_POINT_1 = {
            {0.0f, 0.0f},
            {0.125f, 0.1f},
            {0.25f, 0.25f},
            {0.5f, 0.1f},
            {0.75f, 0.25f},
            {0.875f, 0.1f},
            {1.0f, 0.0f}
    };

    private static final float[][] WAVE_CURRENT_POINT_2 = {
            {0.0f, 0.0f},
            {0.125f, 0.1f},
            {0.3f, 0.02f},
            {0.5f, 0.2f},
            {0.7f, 0.02f},
            {0.875f, 0.1f},
            {1.0f, 0.0f}
    };

    private static final float[][] WAVE_CONTROL_POINT1_1 = {
            {0.0602f, 0.0f},
            {0.1718f, 0.1771f},
            {0.316f, 0.2476f},
            {0.5631f, 0.0976f},
            {0.8101f, 0.2523f},
            {0.9029f, 0.0491f}
    };

    private static final float[][] WAVE_CONTROL_POINT1_2 = {
            {0.0602f, 0.0f},
            {0.1718f, 0.1771f},
            {0.3631f, 0.02f},
            {0.5691f, 0.204f},
            {0.7631f, 0.02f},
            {0.9029f, 0.0491f}
    };

    private static final float[][] WAVE_CONTROL_POINT2_1 = {
            {0.0941f, 0.0491f},
            {0.1869f, 0.2523f},
            {0.434f, 0.0976f},
            {0.681f, 0.2476f},
            {0.8252f, 0.1771f},
            {0.9368f, 0.0f}
    };

    private static final float[][] WAVE_CONTROL_POINT2_2 = {
            {0.0941f, 0.0491f},
            {0.2369f, 0.02f},
            {0.4311f, 0.204f},
            {0.6369f, 0.02f},
            {0.8252f, 0.1771f},
            {0.9368f, 0.0f}
    };

    public CubicToSample(Context context) {
        this(context, null, 0);
    }

    public CubicToSample(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public CubicToSample(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);

        initView();
    }

    private void initView() {
        setUpPaint();
        setUpPath();
        setUpAnimator();
    }

    private void setUpAnimator() {
        mWaveControl1Animator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWaveControl1Animator.start();

        mWaveControl2Animator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWaveControl2Animator.start();

        mWavePositionAnimator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWavePositionAnimator.start();
    }

    private void setUpPaint() {
        mPaint = new Paint();
        mPaint.setColor(Color.RED);
        mPaint.setAntiAlias(true);
        mPaint.setStyle(Paint.Style.FILL);
    }

    private void setUpPath() {
        mWavePath = new Path();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        mWavePath.reset();
        mWavePath.moveTo(WAVE_CURRENT_POINT_2[0][0], WAVE_CURRENT_POINT_2[0][1]);
        float[][] controlPoint1 = (float[][]) mWaveControl1Animator.getAnimatedValue
        float[][] controlPoint2 = (float[][]) mWaveControl2Animator.getAnimatedValue

        float[][] endPoint = (float[][]) mWavePositionAnimator.getAnimatedValue();
        for (int i = 0; i < 6; i++) {
            mWavePath.cubicTo(
                    controlPoint1[i][0] * mWidth, controlPoint1[i][1] * 1000,
                    controlPoint2[i][0] * mWidth, controlPoint2[i][1] * 1000,
                    endPoint[i + 1][0] * mWidth, endPoint[i + 1][1] * 1000
            );
        }
        canvas.drawPath(mWavePath, mPaint);
    }

    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        mWidth = w;
        super.onSizeChanged(w, h, oldw, oldh);
    }


    private ValueAnimator mWaveControl1Animator;

    private ValueAnimator mWaveControl2Animator;

    private ValueAnimator mWavePositionAnimator;

    public void startWaveAnimation() {
        mWaveControl1Animator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWaveControl1Animator.setDuration(500);
        mWaveControl1Animator.addUpdateListener(mAnimatorUpdateListener);
        mWaveControl1Animator.start();

        mWaveControl2Animator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWaveControl2Animator.setDuration(500);
        mWaveControl2Animator.addUpdateListener(mAnimatorUpdateListener);
        mWaveControl2Animator.start();

        mWavePositionAnimator = ValueAnimator.ofObject(new WaveTypeEvaluator(), WAVE_
        mWavePositionAnimator.setDuration(500);
        mWavePositionAnimator.addUpdateListener(mAnimatorUpdateListener);
        mWavePositionAnimator.setInterpolator(new AccelerateDecelerateInterpolator()
        mWavePositionAnimator.start();

    }

    private ValueAnimator.AnimatorUpdateListener mAnimatorUpdateListener = new Value
        @Override
        public void onAnimationUpdate(ValueAnimator valueAnimator) {
            invalidate();
        }
    };

    public static class WaveTypeEvaluator implements TypeEvaluator<float[][]> {

        @Override
        public float[][] evaluate(float v, float[][] start, float[][] end) {
            float[][] newValue = new float[start.length][start[0].length];

            for (int i = 0; i < start.length; i++) {
                for (int j = 0; j < start[i].length; j++) {
                    newValue[i][j] = start[i][j] + (end[i][j] - start[i][j]) * v;
                }
            }
            return newValue;
        }
    }
}
```

CubicToSample.java hosted with ❤ by GitHub　view raw

gist.github.com

で、TypeEvaluatorがここの部分

```
public static class WaveTypeEvaluator implements TypeEvaluator<float[][]> {

        @Override
        public float[][] evaluate(float v, float[][] start, float[][] end) {
            float[][] newValue = new float[start.length][start[0].length];

            for (int i = 0; i < start.length; i++) {
                for (int j = 0; j < start[i].length; j++) {
                    newValue[i][j] = start[i][j] + (end[i][j] - start[i][j]) * v;
                }
            }
            return newValue;
        }
}
```

forの中にlengthを使うなとか言うツッコミは置いておいて。
結論書くと、自分で時間に対する値を設定できるという感じです。
オレオレTypeEvaluatorだったら基本的に何でも捌けて非常に良いですね!!!!!

例えば、今回は多次元配列を例にしていますが、作ったModelClassを引数に渡して

```
public static class WaveTypeEvaluator implements TypeEvaluator<MyPoint> {

        @Override
        public MyPoint evaluate(float v, MyPoint start, MyPoint end) {
            MyPoint newPoint = new MyPoint();
            newPoint.setX(start.getX() + (end.getX() - start.getX()) * v);
            return newPaint;
        }
}
```

こんな感じにイケるんですね。
便利。

ついでにこのViewはこんな感じ

は一素晴らしきTypeEvaluator。
また、ValueAnimatorと少しだけ仲良く慣れました。

最後に

TypeEvaluator、多次元配列だけじゃなくて、ここイジれば「インターポレイラー」も実はこの中に組むことが出来たりと、普通に便利でした。
もっと、Viewのカスタマイズしていって知見をためたい。
あと、内定欲しい。

ではではー

あみゅー (id:amyu_dev) 5年前

★+

1 B! ブックマーク　1 シェア　ツイート　tumblr. ＋

はてなブックマークでのコメント (1 + 0)

B! はてなブックマークでコメントする

alexam android animation 2015/08/20 ★+

はてなブックマークで確認

関連記事

2015-06-27
CustomViewを作るときのテンプレートをAndroid初心者が考えた
CustomViewを作り機会が多い今日このごろ。そこで考えるのがAPI...

2015-02-04
RegionとPathを用いた特定の部分のTouchEventの取り方
前回書いたViewの記事の補足を少し。なんか日本語の記事が見つ...

コメントを書く

« 減衰な Interpolator Interpolator を作ったリッチな View の A... »

プロフィール

あみゅー (id:amyu_dev)
Yuki Mima
22
OCP(Oracle Master Gold)
LPIClv3
Androidが最高
http://amyu.github.io/
＋ 読者になる 41

人気エントリ

エントリー
RecyclerViewのpreloadが素敵だよねっていう話 - あみゅーの(･´ｰ･｀)どや 10users
Interpolator を作ったリッチな View の Animation - あみゅーの(･´ｰ･｀)どや 17users
新時代のToolBarへ - あみゅーの(･´ｰ･｀)どや 18users
徹底比較！ iBeaconモジュール4種類を比較してみたら面白い結果に！ - あみゅーの... 18users
iBeaconのセキュリティと Android - あみゅーの奮闘記2 16users
iBeaconのセキュリティについて ~秋葉原でクラックしてきた~ - あみゅーの奮闘記2 58users

検索

記事を検索

最新記事

ODD開発について
DroidKaigiのCFP書いた
RecyclerViewのpreloadが素敵だよねっていう話
公開したView達
CustomViewを作るときのテンプレートをAndroid初心者が考えた

月別アーカイブ

- ▼ 2015 (21)
  - 2015 / 12 (1)
  - 2015 / 11 (1)
  - 2015 / 10 (1)
  - 2015 / 8 (1)
  - 2015 / 6 (2)
  - 2015 / 5 (3)
  - 2015 / 4 (4)
  - 2015 / 3 (2)
  - 2015 / 2 (5)
  - 2015 / 1 (1)
- ▶ 2014 (9)

カテゴリー

Android (20)

最近のコメント