**Path: Перемещение картинки вдоль произвольной кривой**

Контур может быть полезен для создания интересной анимации - перемещения картинки вдоль произвольной кривой. Для начала познакомимся [со статьёй на эту тему](http://habrahabr.ru/post/137294/).

Возникла задача сделать анимацию — двигать картинку вдоль заданной кривой. Итак, у нас есть некоторая кривая, например, построенная из набора точек и для красивости сглаженная.

//набор точек

List<PointF> aPoints = new ArrayList<PointF>();

aPoints.add(new PointF(10f, 160f));

aPoints.add(new PointF(100f, 100f));

aPoints.add(new PointF(300f, 220f));

aPoints.add(new PointF(640f, 180f));

//строим сглаженную кривую

Path ptCurve = new Path();

PointF point = aPoints.get(0);

ptCurve.moveTo(point.x, point.y);

for(int i = 0; i < aPoints.size() - 1; i++){

point = aPoints.get(i);

PointF next = aPoints.get(i+1);

ptCurve.quadTo(

point.x, point.y,

(next.x + point.x) / 2, (point.y + next.y) / 2

);

}

Нам нужно получать координаты точек на нашей кривой, чтобы там выводить нашу картинку. Для этого воспользуемся классом **PathMeasure**. При помощи этого класса «замерим» длину кривой. А чтобы найти нужную точку, можно передать объекту этого класса длину, на которую точка удалена от начала.

Вот так, например, можно получить координаты точки посередине кривой:

PathMeasure pm = new PathMeasure(ptCurve, false);

float afP[] = {0f, 0f}; // здесь будут координаты

pm.getPosTan(pm.getLength() \* 0.5f, afP, null);

Последним параметром (я передал там null) можно, аналогично координатам, получить параметры касательной в этой точке.

Более того есть метод **getMatrix()**, который дает готовую матрицу трансформации — смещение и нужный поворот. Его мы и будем использовать для вывода спрайта.

Matrix mxTransform = new Matrix();

pm.getMatrix(

pm.getLength() \* 0.5f,

mxTransform,

PathMeasure.POSITION\_MATRIX\_FLAG +

PathMeasure.TANGENT\_MATRIX\_FLAG

);

mxTransform.preTranslate(-bmSprite.getWidth(), -bmSprite.getHeight());

canvas.drawBitmap(bmSprite, mxTransform, null);

Получилось в точности то, что и требовалось:

Полный код приведен ниже или можно скачать проект из репозитория — [SpriteAlongPath](https://bitbucket.org/TedBeer/spritealongpath/downloads) или воспользоваться меркуриалом —hg clone [bitbucket.org/TedBeer/spritealongpath](https://bitbucket.org/TedBeer/spritealongpath).

/\*\*

\* User: TedBeer

\* Date: 30/01/12

\* Time: 12:32

\*/

package net.tedbeer;

import android.app.Activity;

import android.os.Bundle;

import android.content.Context;

import android.graphics.\*;

import android.util.Log;

import android.view.Display;

import android.view.MotionEvent;

import android.view.View;

import android.view.WindowManager;

import android.widget.Toast;

import java.util.\*;

public class moveSprite extends Activity

{

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(new SceneView(this));

}

}

public class SceneView extends View {

private static Bitmap bmSprite;

private static Bitmap bmBackground;

private static Rect rSrc, rDest;

//animation step

private static int iMaxAnimationStep = 20;

private int iCurStep = 0;

//points defining our curve

private List<PointF> aPoints = new ArrayList<PointF>();

private Paint paint;

private Path ptCurve = new Path(); //curve

private PathMeasure pm; //curve measure

private float fSegmentLen; //curve segment length

public SceneView(Context context) {

super(context);

//destination rectangle

Display display = ((WindowManager) context.getSystemService(Context.WINDOW\_SERVICE)).getDefaultDisplay();

rDest = new Rect(0, 0, display.getWidth(), display.getHeight());

//load background

if (bmBackground == null) {

bmBackground = BitmapFactory.decodeResource(getResources(), R.drawable.winter\_mountains);

rSrc = new Rect(0, 0, bmBackground.getWidth(), bmBackground.getHeight());

}

//load sprite

if (bmSprite == null)

bmSprite = BitmapFactory.decodeResource(getResources(), R.drawable.sledge3);

//init random set of points

aPoints.add(new PointF(10f, 160f));

aPoints.add(new PointF(100f, 100f));

aPoints.add(new PointF(300f, 220f));

aPoints.add(new PointF(640f, 180f));

//init smooth curve

PointF point = aPoints.get(0);

ptCurve.moveTo(point.x, point.y);

for(int i = 0; i < aPoints.size() - 1; i++){

point = aPoints.get(i);

PointF next = aPoints.get(i+1);

ptCurve.quadTo(point.x, point.y, (next.x + point.x) / 2, (point.y + next.y) / 2);

}

pm = new PathMeasure(ptCurve, false);

fSegmentLen = pm.getLength() / iMaxAnimationStep;//20 animation steps

//init paint object

paint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

paint.setStyle(Paint.Style.STROKE);

paint.setStrokeWidth(3);

paint.setColor(Color.rgb(0, 148, 255));

}

@Override

protected void onDraw(Canvas canvas) {

canvas.drawBitmap(bmBackground, rSrc, rDest, null);

canvas.drawPath(ptCurve, paint);

//animate the sprite

Matrix mxTransform = new Matrix();

if (iCurStep <= iMaxAnimationStep) {

pm.getMatrix(fSegmentLen \* iCurStep, mxTransform,

PathMeasure.POSITION\_MATRIX\_FLAG + PathMeasure.TANGENT\_MATRIX\_FLAG);

mxTransform.preTranslate(-bmSprite.getWidth(), -bmSprite.getHeight());

canvas.drawBitmap(bmSprite, mxTransform, null);

iCurStep++; //advance to the next step

invalidate();

} else {

iCurStep = 0;

}

}

@Override

public boolean onTouchEvent(MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION\_DOWN ) { //run animation

invalidate();

return true;

}

return false;

}

}

Вернёмся к нашим примерам. Помните, мы проектировали [лестницу для котов](http://developer.alexanderklimov.ru/android/catshop/android.graphics.path.php#methods)?



Доработаем лестницу, чтобы она стала замкнутой, и добавим изображение кота-андроида из [набора счастливых котят](http://alexanderklimov.ru/2013/10/06/12-lucky-cats/).

Оформим в виде нового класса **AnimationPathView**.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.PathMeasure;

import android.util.AttributeSet;

import android.view.View;

import android.widget.Toast;

public class AnimationPathView extends View {

private Paint mPaint;

private Path mPath;

private Bitmap mBitmap;

private PathMeasure mPathMeasure;

private Matrix mMatrix;

private int mOffsetX, mOffsetY;

private float mPathLength;

private float mStep; // distance each step

private float mDistance; // distance moved

private float[] mPosition;

private float[] mTan;

public AnimationPathView(Context context) {

super(context);

// TODO Auto-generated constructor stub

init();

}

public AnimationPathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public AnimationPathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

mBitmap = BitmapFactory.decodeResource(getResources(),

R.drawable.ic\_android\_cat);

mOffsetX = mBitmap.getWidth() / 2;

mOffsetY = mBitmap.getHeight() / 2;

mPath = new Path();

mPath.moveTo(50, 50);

mPath.lineTo(150, 50);

mPath.lineTo(150, 100);

mPath.lineTo(250, 100);

mPath.lineTo(250, 150);

mPath.lineTo(350, 150);

mPath.lineTo(350, 200);

mPath.lineTo(450, 200);

mPath.lineTo(450, 250);

mPath.lineTo(50, 250);

mPath.close();

// mPath.lineTo(50, 50);

mPathMeasure = new PathMeasure(mPath, false);

mPathLength = mPathMeasure.getLength();

Toast.makeText(getContext(), "Path Length: " + mPathLength,

Toast.LENGTH\_LONG).show();

mStep = 1;

mDistance = 0;

mPosition = new float[2];

mTan = new float[2];

mMatrix = new Matrix();

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

canvas.drawPath(mPath, mPaint);

if (mDistance < mPathLength) {

mPathMeasure.getPosTan(mDistance, mPosition, mTan);

mMatrix.reset();

float degrees = (float) (Math.atan2(mTan[1], mTan[0]) \* 180.0 / Math.PI);

mMatrix.postRotate(degrees, mOffsetX, mOffsetY);

mMatrix.postTranslate(mPosition[0] - mOffsetX, mPosition[1] - mOffsetY);

canvas.drawBitmap(mBitmap, mMatrix, null);

mDistance += mStep;

} else {

mDistance = 0;

}

invalidate();

}

}

Подключим компонент программно в активности:

setContentView(new AnimationPathView(this));

Готово.

При повороте на углах лестницы фигура резко меняет своё направление. Сделаем это движение плавным. Для примера я также изменил смещение по высоте, чтобы фигура скользила вдоль лестницы, "перебирая лапами".

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.PathMeasure;

import android.util.AttributeSet;

import android.view.View;

import android.widget.Toast;

public class AnimationPathView extends View {

private Paint mPaint;

private Path mPath;

private Bitmap mBitmap;

private PathMeasure mPathMeasure;

private Matrix mMatrix;

private int mOffsetX, mOffsetY;

private float mPathLength;

private float mStep; // distance each step

private float mDistance; // distance moved

private float[] mPosition;

private float[] mTan;

private float mCurX, mCurY;

private float mCurAngle; // current angle

private float mTargetAngle; // target angle

private float mStepAngle; // angle each step

public AnimationPathView(Context context) {

super(context);

// TODO Auto-generated constructor stub

init();

}

public AnimationPathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public AnimationPathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

mBitmap = BitmapFactory.decodeResource(getResources(),

R.drawable.ic\_android\_cat);

mOffsetX = mBitmap.getWidth() / 2;

mOffsetY = mBitmap.getHeight();

mPath = new Path();

mPath.moveTo(150, 150);

mPath.lineTo(250, 150);

mPath.lineTo(250, 200);

mPath.lineTo(350, 200);

mPath.lineTo(350, 250);

mPath.lineTo(450, 250);

mPath.lineTo(450, 300);

mPath.lineTo(550, 300);

mPath.lineTo(550, 350);

mPath.lineTo(150, 350);

mPath.close();

mPathMeasure = new PathMeasure(mPath, false);

mPathLength = mPathMeasure.getLength();

Toast.makeText(getContext(), "Path Length: " + mPathLength,

Toast.LENGTH\_LONG).show();

mStep = 1;

mDistance = 0;

mCurX = 0;

mCurY = 0;

mStepAngle = 1;

mCurAngle = 0;

mTargetAngle = 0;

mPosition = new float[2];

mTan = new float[2];

mMatrix = new Matrix();

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

canvas.drawPath(mPath, mPaint);

mMatrix.reset();

if ((mTargetAngle - mCurAngle) > mStepAngle) {

mCurAngle += mStepAngle;

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

} else if ((mCurAngle - mTargetAngle) > mStepAngle) {

mCurAngle -= mStepAngle;

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

} else {

mCurAngle = mTargetAngle;

if (mDistance < mPathLength) {

mPathMeasure.getPosTan(mDistance, mPosition, mTan);

mTargetAngle = (float) (Math.atan2(mTan[1], mTan[0]) \* 180.0 / Math.PI);

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mCurX = mPosition[0] - mOffsetX;

mCurY = mPosition[1] - mOffsetY;

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

mDistance += mStep;

} else {

mDistance = 0;

}

}

invalidate();

}

}

Результат.

Мы создавали контур программно. Но мы можем создать контур динамически движением пальца на экране, который можно скопировать в наш первый контур. И пустить по созданному контуру фигурку кота.

package ru.alexanderklimov.path;

import android.content.Context;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Matrix;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.PathMeasure;

import android.util.AttributeSet;

import android.view.MotionEvent;

import android.view.View;

import android.widget.Toast;

public class AnimationPathView extends View {

private Paint mPaint;

private Path mPath;

private Path mTouchPath;

private Bitmap mBitmap;

private PathMeasure mPathMeasure;

private Matrix mMatrix;

private int mOffsetX, mOffsetY;

private float mPathLength;

private float mStep; // distance each step

private float mDistance; // distance moved

private float[] mPosition;

private float[] mTan;

private float mCurX, mCurY;

private float mCurAngle; // current angle

private float mTargetAngle; // target angle

private float mStepAngle; // angle each step

public AnimationPathView(Context context) {

super(context);

// TODO Auto-generated constructor stub

init();

}

public AnimationPathView(Context context, AttributeSet attrs) {

super(context, attrs);

init();

}

public AnimationPathView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

init();

}

private void init() {

mPaint = new Paint();

mPaint.setAntiAlias(true);

mPaint.setColor(Color.RED);

mPaint.setStrokeWidth(3);

mPaint.setStyle(Paint.Style.STROKE);

mBitmap = BitmapFactory.decodeResource(getResources(),

R.drawable.ic\_android\_cat);

mOffsetX = mBitmap.getWidth() / 2;

mOffsetY = mBitmap.getHeight();

mTouchPath = new Path();

mPath = new Path();

mPath.moveTo(150, 150);

mPath.lineTo(250, 150);

mPath.lineTo(250, 200);

mPath.lineTo(350, 200);

mPath.lineTo(350, 250);

mPath.lineTo(450, 250);

mPath.lineTo(450, 300);

mPath.lineTo(550, 300);

mPath.lineTo(550, 350);

mPath.lineTo(150, 350);

mPath.close();

mPathMeasure = new PathMeasure(mPath, false);

mPathLength = mPathMeasure.getLength();

Toast.makeText(getContext(), "Path Length: " + mPathLength,

Toast.LENGTH\_LONG).show();

mStep = 1;

mDistance = 0;

mCurX = 0;

mCurY = 0;

mStepAngle = 1;

mCurAngle = 0;

mTargetAngle = 0;

mPosition = new float[2];

mTan = new float[2];

mMatrix = new Matrix();

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

if (mPath.isEmpty()) {

return;

}

canvas.drawPath(mPath, mPaint);

mMatrix.reset();

if ((mTargetAngle - mCurAngle) > mStepAngle) {

mCurAngle += mStepAngle;

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

invalidate();

} else if ((mCurAngle - mTargetAngle) > mStepAngle) {

mCurAngle -= mStepAngle;

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

invalidate();

} else {

mCurAngle = mTargetAngle;

if (mDistance < mPathLength) {

mPathMeasure.getPosTan(mDistance, mPosition, mTan);

mTargetAngle = (float) (Math.atan2(mTan[1], mTan[0]) \* 180.0 / Math.PI);

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mCurX = mPosition[0] - mOffsetX;

mCurY = mPosition[1] - mOffsetY;

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

mDistance += mStep;

invalidate();

} else {

// mDistance = 0;

mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);

mMatrix.postTranslate(mCurX, mCurY);

canvas.drawBitmap(mBitmap, mMatrix, null);

}

}

// invalidate();

}

@Override

public boolean onTouchEvent(MotionEvent event) {

int action = event.getAction();

switch (action) {

case MotionEvent.ACTION\_DOWN:

mTouchPath.reset();

mTouchPath.moveTo(event.getX(), event.getY());

break;

case MotionEvent.ACTION\_MOVE:

mTouchPath.lineTo(event.getX(), event.getY());

break;

case MotionEvent.ACTION\_UP:

mTouchPath.lineTo(event.getX(), event.getY());

mPath = new Path(mTouchPath);

mPathMeasure = new PathMeasure(mPath, false);

mPathLength = mPathMeasure.getLength();

mStep = 1;

mDistance = 0;

mCurX = 0;

mCurY = 0;

mStepAngle = 1;

mCurAngle = 0;

mTargetAngle = 0;

invalidate();

break;

}

return true;

}

}

**Замеряем производительность**

Разные устройства имеют свои технические характеристики - размеры экрана, разрешения и т.д. И для каждого устройства скорость движения объекта по траектории будет своя. Чтобы замерить производительность кода, добавим дополнительные строчки в класс **AnimationPathView**.

private Paint mTextPaint;

private long mLastTime;

private void init() {

...

mTextPaint = new Paint();

mTextPaint.setColor(Color.BLUE);

mTextPaint.setStrokeWidth(1);

mTextPaint.setStyle(Paint.Style.FILL);

mTextPaint.setTextSize(26);

...

mLastTime = System.currentTimeMillis();

}

@Override

protected void onDraw(Canvas canvas) {

// TODO Auto-generated method stub

super.onDraw(canvas);

if (mPath.isEmpty()) {

return;

}

long startNanos = System.nanoTime();

long startMillis = System.currentTimeMillis();

...

long endNanos = System.nanoTime();

long betweenFrame = startMillis - mLastTime;

int fps = (int) (1000 / betweenFrame);

String strProcessingTime = "Processing Time (ns = 0.000001ms) = "

+ (endNanos - startNanos);

String strBetweenFrame = "Between Frame (ms) = " + betweenFrame;

String strFPS = "Frame Per Second (approximate) = " + fps;

mLastTime = startMillis;

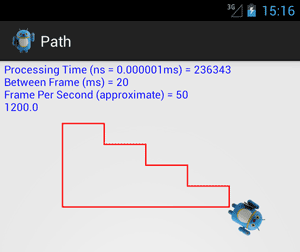
canvas.drawText(strProcessingTime, 10, 30, mTextPaint);

canvas.drawText(strBetweenFrame, 10, 60, mTextPaint);

canvas.drawText(strFPS, 10, 90, mTextPaint);

canvas.drawText(String.valueOf(mPathLength), 10, 120, mTextPaint);

}



Для управления скоростью движения объекта вдоль контура добавьте код из статьи **Change speed of Animation follow touch path** (ссылка внизу). Для изменения направления движения объекта смотрите статью **Animation follow touch path forward and backward**.

**Использованные материалы**

[Android-er: Animation of moving bitmap along path](http://android-er.blogspot.ru/2014/05/animation-of-moving-bitmap-along-path.html)

[Android-er: Smooth turning along path](http://android-er.blogspot.ru/2014/05/smooth-turning-along-path.html)

[Android-er: Animation follow touch path](http://android-er.blogspot.ru/2014/05/animation-follow-touch-path.html)

[Android-er: Know the performance, timing and speed of animation](http://android-er.blogspot.ru/2014/05/know-performance-timing-and-speed-of.html)

[Android-er: Change speed of Animation follow touch path](http://android-er.blogspot.ru/2014/06/change-speed-of-animation-follow-touch.html)

[Android-er: Animation follow touch path forward and backward](http://android-er.blogspot.ru/2014/06/animation-follow-touch-path-forward-and.html)