当我们需要在屏幕上形成画线时，Path类的应用是必不可少的，而Path类的lineTo和quadTo方法实现的绘制线路形式也是不一样的，下面就以代码的实现来直观的探究这两个方法的功能实现区别；

1. Path--->quadTo(float x1, float y1, float x2, float y2):

该方法的实现是当我们不仅仅是画一条线甚至是画弧线时会形成平滑的曲线，该曲线又称为"贝塞尔曲线"(Bezier curve)，其中，x1，y1为控制点的坐标值，x2，y2为终点的坐标值；

贝塞尔曲线的形成，就比如我们把一条橡皮筋拉直，橡皮筋的头尾部对应起点和终点，然后从拉直的橡皮筋中选择任意一点（除头尾对应的点外）扯动橡皮筋形成的弯曲形状，而那个扯动橡皮筋的点就是控制点；

下就面以一个Demo来结合理解quadTo函数的应用，代码如下：

1).自定义View：

package com.feixun.hu.pt;

import android.content.Context;

import android.gesture.GestureStroke;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.Rect;

import android.graphics.Paint.Style;

import android.view.MotionEvent;

import android.view.View;

import android.widget.Toast;

public class DrawingWithBezier extends View

{

private float mX;

private float mY;

private final Paint mGesturePaint = new Paint();

private final Path mPath = new Path();

public DrawingWithBezier(Context context)

{

super(context);

mGesturePaint.setAntiAlias(true);

mGesturePaint.setStyle(Style.STROKE);

mGesturePaint.setStrokeWidth(5);

mGesturePaint.setColor(Color.WHITE);

}

@Override

public boolean onTouchEvent(MotionEvent event)

{

// TODO Auto-generated method stub

switch (event.getAction())

{

case MotionEvent.ACTION\_DOWN:

touchDown(event);

break;

case MotionEvent.ACTION\_MOVE:

touchMove(event);

}

//更新绘制

invalidate();

return true;

}

@Override

protected void onDraw(Canvas canvas)

{

// TODO Auto-generated method stub

super.onDraw(canvas);

//通过画布绘制多点形成的图形

canvas.drawPath(mPath, mGesturePaint);

}

//手指点下屏幕时调用

private void touchDown(MotionEvent event)

{

//mPath.rewind();

//重置绘制路线，即隐藏之前绘制的轨迹

mPath.reset();

float x = event.getX();

float y = event.getY();

mX = x;

mY = y;

//mPath绘制的绘制起点

mPath.moveTo(x, y);

}

//手指在屏幕上滑动时调用

private void touchMove(MotionEvent event)

{

final float x = event.getX();

final float y = event.getY();

final float previousX = mX;

final float previousY = mY;

final float dx = Math.abs(x - previousX);

final float dy = Math.abs(y - previousY);

//两点之间的距离大于等于3时，生成贝塞尔绘制曲线

if (dx >= 3 || dy >= 3)

{

//设置贝塞尔曲线的操作点为起点和终点的一半

float cX = (x + previousX) / 2;

float cY = (y + previousY) / 2;

//二次贝塞尔，实现平滑曲线；previousX, previousY为操作点，cX, cY为终点

mPath.quadTo(previousX, previousY, cX, cY);

//第二次执行时，第一次结束调用的坐标值将作为第二次调用的初始坐标值

mX = x;

mY = y;

}

}

}

package com.feixun.hu.pt;

import android.app.Activity;

import android.os.Bundle;

import android.view.Window;

import android.view.WindowManager;

public class MainActivity extends Activity

{

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

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

//setContentView(new MySurfaceView(this));

setContentView(new DrawingWithBezier(this));

//setContentView(new DrawingWithoutBezier(this));

}

}

用SurfaceView实现

package com.feixun.hu.pt;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Path;

import android.graphics.Rect;

import android.graphics.Paint.Style;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView

{

private Context mContex;

private float mX;

private float mY;

private SurfaceHolder sfh;

private Canvas canvas;

private float mCurveEndX;

private float mCurveEndY;

private final Paint mGesturePaint = new Paint();

private final Path mPath = new Path();

private final Rect mInvalidRect = new Rect();

private boolean isDrawing;

public MySurfaceView(Context context)

{

super(context);

mContex = context;

sfh = this.getHolder();

mGesturePaint.setAntiAlias(true);

mGesturePaint.setStyle(Style.STROKE);

mGesturePaint.setStrokeWidth(5);

mGesturePaint.setColor(Color.WHITE);

// TODO Auto-generated constructor stub

}

public void drawCanvas() {

try {

canvas = sfh.lockCanvas();

if (canvas != null) {

canvas.drawColor(Color.BLACK);

canvas.drawPath(mPath, mGesturePaint);

}

} catch (Exception e) {

// TODO: handle exception

} finally {

if (canvas != null)

sfh.unlockCanvasAndPost(canvas);

}

}

@Override

public boolean onTouchEvent(MotionEvent event)

{

// TODO Auto-generated method stub

switch (event.getAction())

{

case MotionEvent.ACTION\_DOWN:

touchDown(event);

invalidate();

return true;

case MotionEvent.ACTION\_MOVE:

if (isDrawing)

{

Rect rect = touchMove(event);

if (rect != null) {

invalidate(rect);

}

return true;

}

break;

case MotionEvent.ACTION\_UP:

if (isDrawing)

{

touchUp(event);

invalidate();

return true;

}

break;

}

return super.onTouchEvent(event);

}

private void touchDown(MotionEvent event)

{

isDrawing = true;

mPath.reset();

float x = event.getX();

float y = event.getY();

mX = x;

mY = y;

mPath.moveTo(x, y);

mInvalidRect.set((int) x, (int) y, (int) x , (int) y);

mCurveEndX = x;

mCurveEndY = y;

}

private Rect touchMove(MotionEvent event)

{

Rect areaToRefresh = null;

final float x = event.getX();

final float y = event.getY();

final float previousX = mX;

final float previousY = mY;

final float dx = Math.abs(x - previousX);

final float dy = Math.abs(y - previousY);

if (dx >= 3 || dy >= 3)

{

areaToRefresh = mInvalidRect;

areaToRefresh.set((int) mCurveEndX , (int) mCurveEndY ,

(int) mCurveEndX, (int) mCurveEndY);

//设置贝塞尔曲线的操作点为起点和终点的一半

float cX = mCurveEndX = (x + previousX) / 2;

float cY = mCurveEndY = (y + previousY) / 2;

//实现绘制贝塞尔平滑曲线；previousX, previousY为操作点，cX, cY为终点

mPath.quadTo(previousX, previousY, cX, cY);

//mPath.lineTo(x, y);

// union with the control point of the new curve

/\*areaToRefresh矩形扩大了border(宽和高扩大了两倍border)，

\* border值由设置手势画笔粗细值决定

\*/

areaToRefresh.union((int) previousX, (int) previousY,

(int) previousX, (int) previousY);

/\* areaToRefresh.union((int) x, (int) y,

(int) x, (int) y);\*/

// union with the end point of the new curve

areaToRefresh.union((int) cX, (int) cY ,

(int) cX, (int) cY);

//第二次执行时，第一次结束调用的坐标值将作为第二次调用的初始坐标值

mX = x;

mY = y;

drawCanvas();

}

return areaToRefresh;

}

private void touchUp(MotionEvent event)

{

isDrawing = false;

}

}

package com.feixun.hu.pt;

import android.app.Activity;

import android.os.Bundle;

import android.view.Window;

import android.view.WindowManager;

public class MainActivity extends Activity

{

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

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(new MySurfaceView(this));

//setContentView(new DrawingWithBezier(this));

//setContentView(new DrawingWithoutBezier(this));

}

}