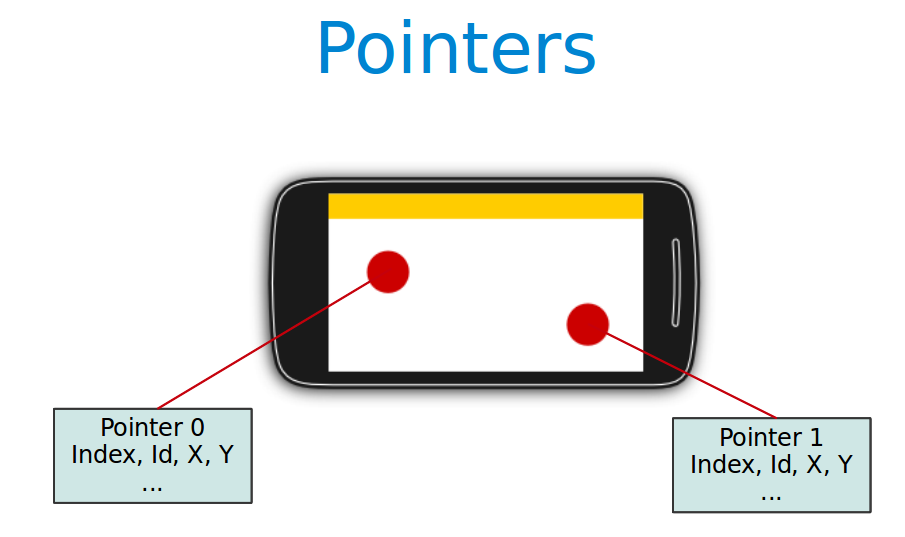
Android Touch

The Android standard **View**class support touch events. You can react to touch events in your custom views and your activities. Android supports multiple pointers, e.g. fingers which are interacting with the screen.



The base class for touch support is the **MotionEvent** class which is passed to Views via the **onTouchEvent()** method. To react to touch events you override the **onTouchEvent()** method.

The **MotionEvent** class contains the touch related information, e.g., the number of pointers, the X/Y coordinates and size and pressure of each pointer.

**Single touch**

If single input is used you can use the **getX()** and **getY()** methods to get the current position of the first finger.

Via the **getAction()** method you receive the action which was performed. The **MotionEvent** class provides the following constants to determine the action which was performed.

| **Event** | **Description** |
| --- | --- |
| **MotionEvent.ACTION\_DOWN** | **New touch started** |
| **MotionEvent.ACTION\_MOVE** | **Finger is moving** |
| **MotionEvent.ACTION\_UP** | **Finger went up** |
| **MotionEvent.ACTION\_CANCEL** | **Current event has been canceled, something else took control of the touch event** |
| **MotionEvent.ACTION\_POINTER\_DOWN** | **Pointer down (multi-touch)** |
| **MotionEvent.ACTION\_POINTER\_UP** | **Pointer up (multi-touch)** |

Example Program

**Activity Snippet**

**public** **class** MainActivity **extends** Activity

{

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

@Override

**public** **boolean** onTouchEvent(MotionEvent event)

{

**float** x = event.getX();

**float** y = event.getY();

**switch** (event.getAction())

{

**case** MotionEvent.*ACTION\_DOWN* :

{

Toast.*makeText*(**this**, "U Touched Screen",3000).show();

Toast.*makeText*(**this**, "X = "+x+" and Y = "+y,3000).show();

**return** **true**;

}

}

**return** **true**;

}

}

**Paint class ( android.graphics.Paint )**

The Paint class holds the style and color information about how to draw geometries, text and bitmaps.

Paint paint = **new** Paint ();

paint.setStrokeWidth(6f);

paint.setColor(Color.*RED*);

paint.setStyle(Paint.Style.*STROKE*);

paint.setStrokeJoin(Paint.Join.*ROUND*);

**setStrokeWidth(float width)**

Set the width for stroking. Pass 0 to stroke in hairline mode. Hairlines always draws a single pixel independent of the canva's matrix.

**setColor(int color)**

Set the paint's color.

**setStyle(Style style)**

Set the paint's style.

**setStrokeJoin(Join join)**

Set the paint's Join.

|  |  |  |
| --- | --- | --- |
| [Paint.Join](http://developer.android.com/reference/android/graphics/Paint.Join.html) | BEVEL | The outer edges of a join meet with a straight line |
| [Paint.Join](http://developer.android.com/reference/android/graphics/Paint.Join.html) | MITER | The outer edges of a join meet at a sharp angle |
| [Paint.Join](http://developer.android.com/reference/android/graphics/Paint.Join.html) | ROUND | The outer edges of a join meet in a circular arc. |

@Override

**protected** **void** onDraw(Canvas canvas)

{

//canvas.drawPath(path, paint);

canvas.drawCircle(50, 25, 100, paint);

}

**onDraw([Canvas](eclipse-javadoc:%E2%98%82=TouchExmapleWithPaint/src%3Ccom.nnk.touchexmaplewithpaint%7BPaintView.java%E2%98%83PaintView~onDraw~QCanvas;%E2%98%82Canvas) canvas)**

Implement this to do your drawing.

**Path Class (android.graphics.Path )**

The Path class encapsulates compound (multiple contours) geometric paths consisting of straight line segments, quadratic curves, and cubic curves. It can be drawn with **canvas.drawPath(path, paint)**, either filled or stroked (based on the paint's Style), or it can be used for clipping or to draw text on a path.

**moveTo(float x, float y)**

Set the beginning of the next contour to the point (x,y).

**Parameters:**

**x** The x-coordinate of the start of a new contour

**y** The y-coordinate of the start of a new contour

**lineTo(float x, float y)**

Add a line from the last point to the specified point (x,y).

If no moveTo() call has been made for this contour, the first point is automatically set to (0,0).

**Parameters:**

**x** The x-coordinate of the end of a line

**y** The y-coordinate of the end of a line

**Example Program**

**public** **class** PaintView **extends** View

{

Paint paint = **new** Paint();

Path path = **new** Path();

**public** PaintView(Context context, AttributeSet attrs)

{

**super**(context, attrs);

// paint.setAntiAlias(true);

paint.setStrokeWidth(6f);

paint.setColor(Color.*RED*);

paint.setStyle(Paint.Style.*STROKE*);

paint.setStrokeJoin(Paint.Join.*ROUND*);

}

@Override

**protected** **void** onDraw(Canvas canvas)

{

//canvas.drawPath(path, paint);

canvas.drawCircle(50, 25, 100, paint);

}

@Override

**public** **boolean** onTouchEvent(MotionEvent event)

{

**float** x = event.getX();

**float** y = event.getY();

**switch** (event.getAction())

{

**case** MotionEvent.*ACTION\_DOWN* :

{

path.moveTo(x, y);

**return** **true**;

}

**case** MotionEvent.*ACTION\_MOVE* :

{

path.lineTo(x, y);

**break**;

}

}

invalidate();

**return** **true**;

}

}

**Activity**

**public** **class** MainActivity **extends** Activity

{

@Override

**protected** **void** onCreate(Bundle savedInstanceState)

{

**super**.onCreate(savedInstanceState);

setContentView(**new** PaintView(**this**, **null**));

}

}