Touchscreens

Introduction

- Many of today handsets include a touchscreen. Most of the user's input comes through the touchscreen.
- The touchscreen is made up of special materials that can pick up pressure and convert that to screen coordinates.

MotionEvent

- When the user touches the screen the `MotionEvent` class is instantiated and passed over to the appropriate method in our application.

  One of the possible methods is the `onTouchEvent()` method that was defined in `View`.

- The `MotionEvent` object contains information about where and when the touch took place.

Events Sequence

- When the user touches the screen a sequence of events starts.
- Each one of these events is described by a `MotionEvent` object. One of the `MotionEvent` properties is `Action`.
- Calling the `getAction` method on the `MotionEvent` object that was passed over to our method we shall get one of the following values: `ACTION_DOWN`, `ACTION_MOVE`, `ACTION_UP`, `ACTION_OUTSIDE` or `ACTION_CANCEL`.

Events Handling

- There are two possible ways for handling the motion events.
- One option is overriding the `onTouchEvent()` method defined in `View`.
- The other option is defining an events listener that implements the `View.OnTouchListener` interface and attach it with our view object by calling the `setOnTouchListener()` method on it.

Handling Method Returned Value

- If the method that handles the motion event (either `onTouchEvent()` or `onTouch()`) consumes the event and no one else needs to be notified then it should return true.
- If the returned value is `false` then none of the additional motion events (of the same sequence) will be passed over to that method.

Handling Method Returned Value

- Setting up an events listener with `onTouch()` the motion events will be delivered to it before been delivered to the View's `onTouchEvent()` method.
- If the events listener `onTouch()` method returns true the motion events won't be passed over to the View's `onTouchEvent()` method.

Sample

```
public class TouchActivity extends Activity
{
    class TouchMe extends TextView
    {
        TouchMe()
        {
            super(TouchActivity.this);
        }
        @Override
        public boolean onTouchEvent(MotionEvent event)
        {
            String str = "code: "+event.getAction();
            str+="\nx: "+event.getRawX();
            str+="\ny: "+event.getRawY();
            setText(str);
            return true;
        }
    }
```

Sample

```
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TouchMe view = new TouchMe();
        setContentView(view);
        view.requestFocus();
    }
}
```

Sample

Gestures

- Gesture is a prerecorded touchscreen motion our application can expect to receive from the user.
- Gesture is a touch sequence that starts when the finger touches the screen and ends when that very same finger lifts up from the screen.

The `GestureOverlayView` Class

- This class describes a transparent overlay through which we can get gesture input.
- Once instantiating this class we can place the object we get either on top of other widgets or have it containing widgets by itself.

The `OnGestureListener` Interface

- This interface was defined as an inner type within the `GestureOverlayView` class.
- This interface describes a listener we can set for the `GestureOverlayView` object in our application.

```
...
GestureOverlayView overlay = new GestureOverlayView(context);
overlay.addOnGestureListener(new MyGestureListener());
...
```

The `OnGestureListener` Interface

- This interface includes the following four abstract callback methods.

```
public abstract void onGesture (
                GestureOverlayView overlay,
                MotionEvent event)

public abstract void onGestureCancelled (
                GestureOverlayView overlay,
                MotionEvent event)
```

The `OnGestureListener` Interface

```
public abstract void onGestureEnded (
                GestureOverlayView overlay,
                MotionEvent event)

public abstract void onGestureStarted (
                GestureOverlayView overlay,
                MotionEvent event)
```

The `getGesture()` Method

- Calling this method on our `GestureOverlayView` object we shall get a reference for a `Gesture` object that describes the current gesture.

```
...
public void onGestureEnded(GestureOverlayView overlay, MotionEvent event)
{
            myGesture = overlay.getGesture();
            ...
}
...
```

The `clear()` Method

- Calling this method on our `GestureOverlayView` object we shall delete the data of the current gesture.

```
...
public void onGestureEnded(GestureOverlayView overlay, MotionEvent event)
{
        myGesture = overlay.getGesture();
        ...
        overlay.clear();
}
...
```

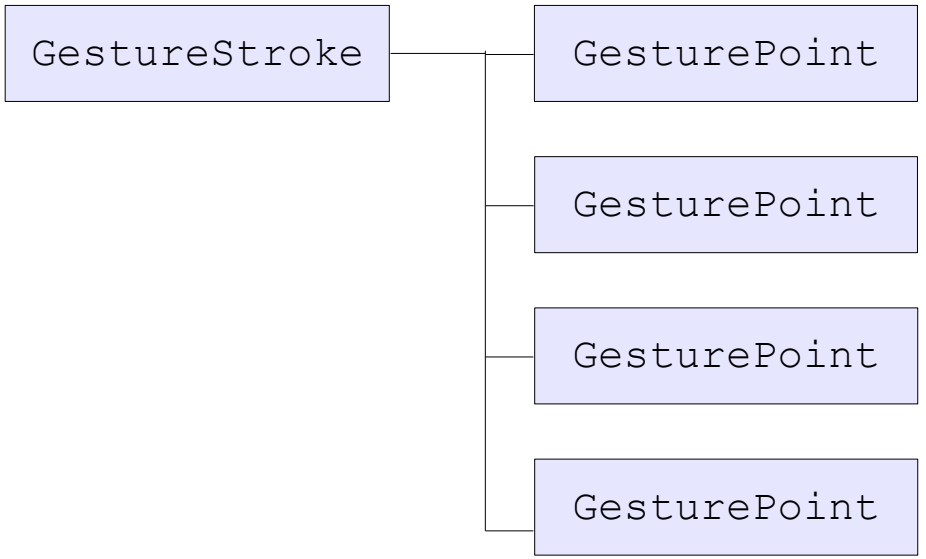
The `GestureStroke` Class

- Each `Gesture` object holds one (or more) `GestureStroke` objects. Calling the `getStrokes()` method on a `Gesture` object we shall get an array list of `GestureStroke` objects.

```
...
public void onGestureEnded(GestureOverlayView overlay, MotionEvent event)
{
        myGesture = overlay.getGesture();
        ArrayList<GestureStroke> array = myGesture.getStrokes();
        ...
        overlay.clear();
}
...
```

The `GesturePoint` Class

- Each `GestureStroke` is composed of `GesturePoint` objects.



The `FadeOffSet` Value

- The Android platform uses a value known as `FadeOffSet`. It is the time in milliseconds that if we wait longer than that time for drawing the next gesture stroke the Android platform assumes we are starting a new gesture.
- The default value of `FadeOffSet` is 420 milliseconds. If we are drawing a gesture and lift up the finger for more than 420 milliseconds before drawing the next gesture stroke then the Android platform will assume that we have already finished.

The `FadeOffSet` Value

- We can change the `FadeOffSet` value using the `android:fadeOffSet` attribute within the `GestureOverlayView` XML element.

The `GestureLibrary` Class

- This class can hold gestures and allow us to compare a new gesture with those it holds.

```
...
ArrayList<Gesture> array = myGestureLib.getGestures(entryName)
...
ArrayList<Prediction> array = myGestureLib.recognize(theNewGesture);
...
```

The `GestureLibraries` Class

- This class allows us easily to get a `GestureLibrary` object from a file.

```
...
GestureLibrary myLib = GestureLibraries.fromFile("thelib");
...
```

Simple Gesture Demo

```
public class GestureDemoActivity extends Activity
{
    private Gesture gesture = null;
    private EditText txt = null;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gesturelayout);
        GestureOverlayView overlay = (GestureOverlayView)
            findViewById(R.id.gesturesoverlay);
        txt = (EditText) findViewById(R.id.txt);
        overlay.addOnGestureListener(new MyGestureListener());
    }

    public class MyGestureListener implements
            GestureOverlayView.OnGestureListener
    {
        public void onGestureStarted(
                GestureOverlayView overlay,
                MotionEvent event)
        {
            gesture = null;
        }
```

Simple Gesture Demo

```
        public void onGesture(
                    GestureOverlayView overlay,
                    MotionEvent event)
        {
        }

        public void onGestureEnded(
                    GestureOverlayView overlay,
                    MotionEvent event)
        {
            gesture = overlay.getGesture();
            txt.setText("gesture length is "+gesture.getLength());
        }

        public void onGestureCancelled(
                    GestureOverlayView overlay,
                    MotionEvent event)
        {
        }
    }
}
```

Simple Gesture Demo

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <android.gesture.GestureOverlayView
        android:id="@+id/gesturesoverlay"
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1.0"
        android:gestureStrokeType="multiple" />

    <EditText android:text="..." android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"></EditText>

</LinearLayout>
```

Simple Gesture Demo