6. Eseménykezelés

A szép komponensek és a tetszets elrendezések mit sem érnek, ha nem értesülünk a bekövetkezett eseményekrl, s nem reagálunk rájuk. Ha csak a képernyre és a beviteli perifériákra szorítkozunk, akkor az esemény érkezhet az érintképernyrl vagy a (virtuális)billentyzetrl. Ehhez jönnek a komponensek eseményei, amikor lenyomunk egy gombot, fókuszt váltunk a felületen vagy karaktert írunk vagy törlünk.

6.1. A kattintás

Egy grafikus felhasználói interfész leglényegesebb eseménye a kattintás, a legtöbb komponensnek van a kattintás lekezeléséhez metódusa, bár a kattintás tipikusan a gombokhoz való esemény, lássunk egy példát rá:

main.xml

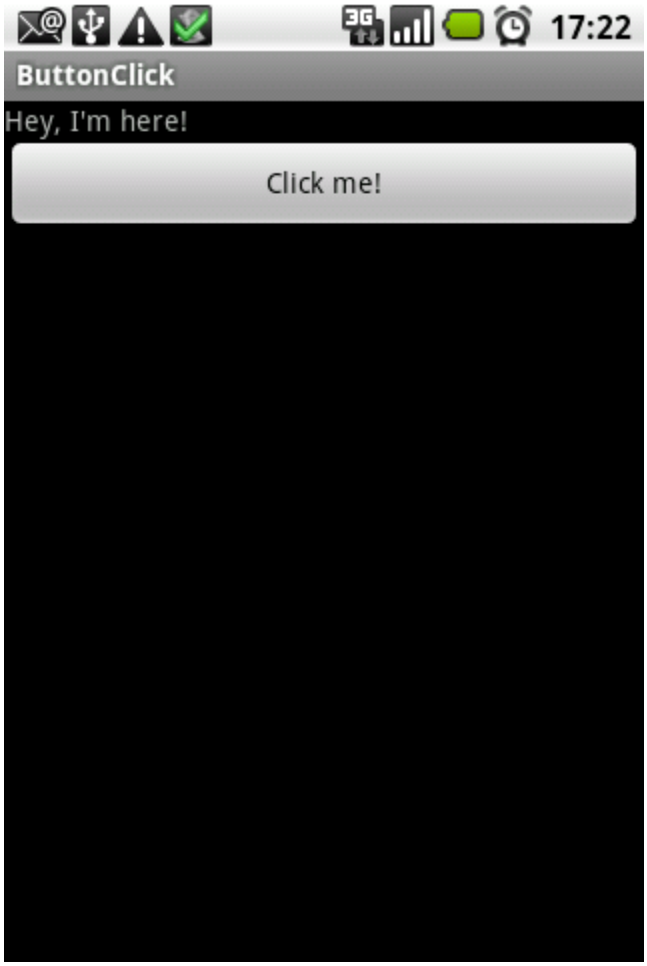
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent">
  <TextView android:id="@+main/label"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:text=""/>
  <Button android:id="@+main/button"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:text="Click me!"/>
</LinearLayout>
```

A hozzá tartozó program se túl hosszú:

MainActivity.java

```
public class MainActivity extends Activity implements View.OnClickListener
{

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    Button button = (Button) findViewById(R.main.button);
    button.setOnClickListener(this);
  }

  @Override
  public void onClick(View view)
  {
    TextView textView = (TextView) findViewById(R.main.label);
    textView.setText("Hey, I'm here!");
  }
}
```

A program implementálja a *View.OnClickListener* interfészt, s megvalósítja annak *onClick* metódusát. Ebben a metódusban egyszeren kiírjuk a *label* azonosítójú *TextView* komponensre, hogy *"Hey, I'm here!"*. Ahhoz, hogy mködjön a dolog, hozzá hozzá kell rendelnünk a nyomógombhoz az *OnClickListen er* implementációt, ezt a *setOnClickListener* metódussal tudjuk megtenni. Lássuk az eredményt:

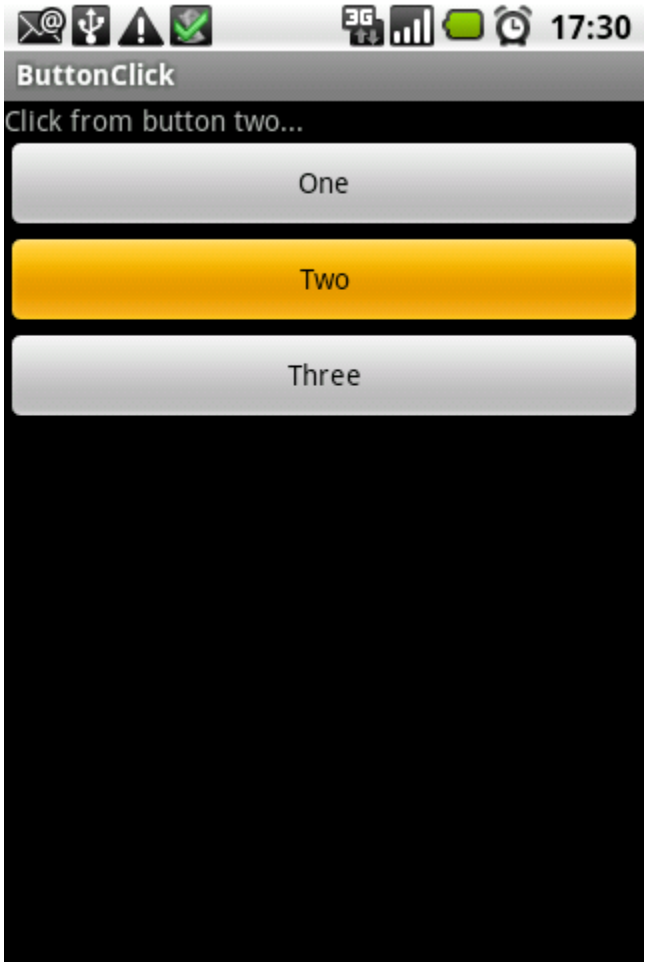


Felmerül a kérdés, hogy hogyan tudjuk megkülönböztetni a kattintás forrását, erre szolgál az *onClick* metódus *view* paramétere:

Java forrás

```
@Override
public void onClick(View view)
{
  TextView textView = (TextView) findViewById(R.main.label);

  switch (view.getId())
  {
    case R.main.button1:
      textView.setText("Click from button one...");
      break;
    case R.main.button2:
      textView.setText("Click from button two...");
      break;
    case R.main.button3:
      textView.setText("Click from button three...");
      break;
  }
}
```

Természetesen ehhez létre kell hoznunk három nyomógombot, illetve mind a háromhoz hozzá kell rendelni az eseménykezelt, csak ezek után tudunk a kattintás forrása után érdekldni a komponenes azonosítóját felhasználva:



6.2. Az érintés

Az érintképerny megköveteli az érintés kezelését, ez leginkább az egérmutató követéséhez hasonlít a desktop világban, ám annál több információt tudunk kinyerni - ha támogatja azt a hardver illetve a platform. Az Android által támogatott plusz információk közül az egyik az érintés erssége (*pressure*), a másik az érintés kiterjedése (*size*), ám nem minden hardver támogatja ezeket az információkat. A példaprogram tekintetében tegyünk egy teljes felületet beborító *TextView* komponenst a képernyre:

main.xml

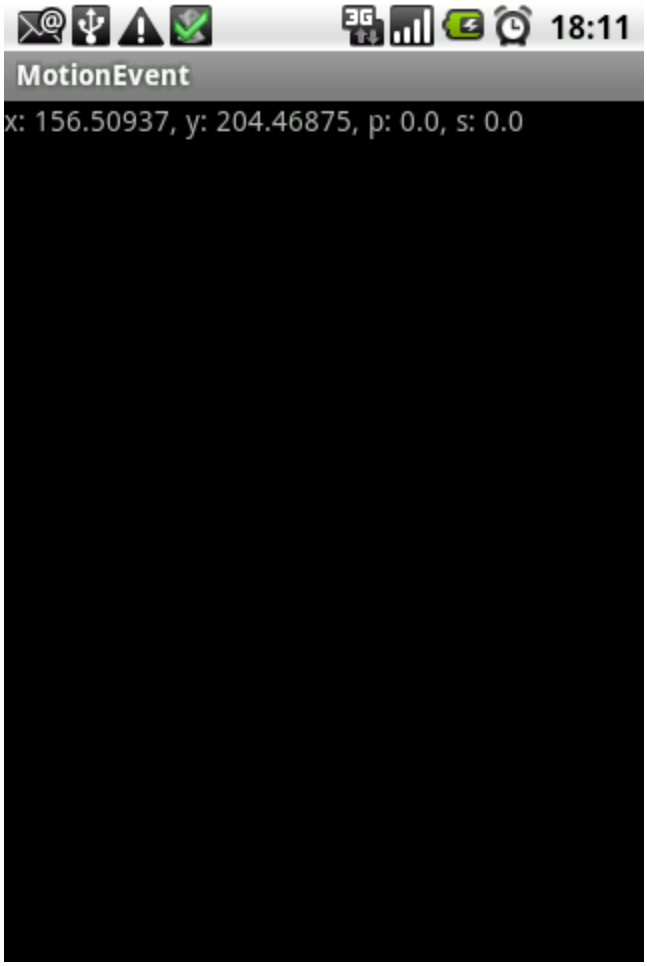
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent">
  <TextView android:id="@+main/label"
          android:layout_height="fill_parent"
          android:layout_width="fill_parent"
          android:text=""/>
</LinearLayout>
```

Majd írjuk meg hozzá az alábbi kis programocskát:

MainActivity.java

```
public class MainActivity extends Activity implements View.OnTouchListener
{

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    TextView textView = (TextView) findViewById(R.main.label);
    textView.setOnTouchListener(this);
  }

  @Override
  public boolean onTouch(View view, MotionEvent event)
  {
    float x = event.getX();
    float y = event.getY();
    float p = event.getPressure();
    float s = event.getSize();

    TextView textView = (TextView) findViewById(R.main.label);
    textView.setText("x: " + x + ", y: " + y + ", p: " + p + ", s: " + s);

    return true;
  }
}
```

Ez a pár soros program mindössze annyit csinál, hogy lekérdezi és kiírja az érintés négy jellemzjét:



Szintén érintképernyhöz köthet esemény a hosszú érintés, amelyre legtöbb esetben egy felbukkanó menü a válasz, de ne szaladjunk ennyire elre, elégedjünk meg egy puszta üzenettel:

MainActivity.java

```
public class MainActivity extends Activity implements View.OnLongClickListener
{

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    TextView textView = (TextView) findViewById(R.main.label);
    textView.setOnLongClickListener(this);
  }

  @Override
  public boolean onLongClick(View view)
  {
    TextView textView = (TextView) findViewById(R.main.label);
    textView.setText("Why do you touch me?!");

    return true;
  }
}
```

A program eredményeképpen hosszabb érintés után meg fog jelenni a *"Why do you touch me?"* szöveg.

6.3. A fókuszváltás

Gyakori feladat, hogy egy beviteli mez elhagyásakor, vagy belelépésekor végrehajtsunk valamilyen eseményt, például ellenrizzük a tartalmát, hogy megfelel-e a kívánalmainknak és rögtön jelezzük ezt a felhasználó számára. Amikor egy beviteli mez megkapja a "figyelmet", akkor keletkezik egy fókusz-esemény, amelyre fel tud iratkozni bármelyik komponens a *setOnFocusListener* metódus meghívásával. Nézzük a példaprogramot:

main.xml

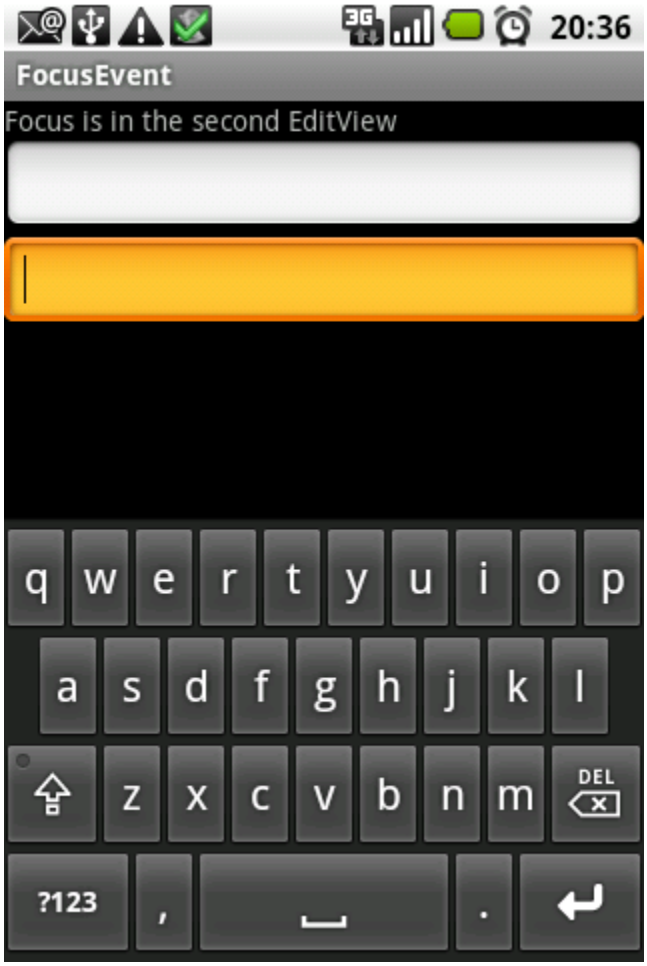
```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_height="fill_parent"
              android:layout_width="fill_parent">
  <TextView android:id="@+main/label"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:text=""/>
  <EditText android:id="@+main/edit1"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:text=""/>
  <EditText android:id="@+main/edit2"
          android:layout_height="wrap_content"
          android:layout_width="fill_parent"
          android:text=""/>
</LinearLayout>
```

A két beviteli mez elé tettünk egy egyszer címkét, ebben fogjuk megjelentetni, hogy éppen melyik beviteli mez az aktív:

MainActivity.java

```
public class MainActivity extends Activity implements View.OnFocusChangeListener
{

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    EditText editText1 = (EditText) findViewById(R.main.edit1);
    editText1.setOnFocusChangeListener(this);
    EditText editText2 = (EditText) findViewById(R.main.edit2);
    editText2.setOnFocusChangeListener(this);
  }

  @Override
  public void onFocusChange(View view, boolean hasFocus)
  {
    TextView textView = (TextView) findViewById(R.main.label);
    if (hasFocus)
    {
      switch (view.getId())
      {
        case R.main.edit1:
          textView.setText("Focus is in the first EditView");
          break;
        case R.main.edit2:
          textView.setText("Focus is in the second EditView");
          break;
      }
    }
  }
}
```

A program futása során mindig az a szöveg jelenik meg a címkében, amelyik beviteli mez az aktív:



-.- folytatása következik -.-