Android, interfaces utilisateur

Département Informatique

2019

LinearLayout

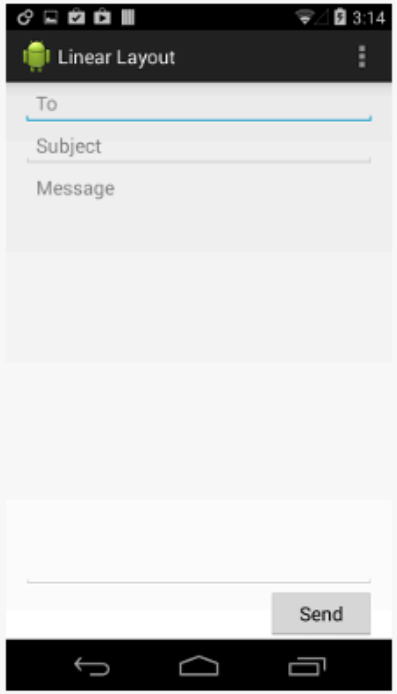
- **Empilement des vues**
  - **Horizontalement** ou **verticalement**
  - **Proportions variables**

- Source : `http://developer.android.com`

LinearLayout : exemple

- source : `http://developer.android.com`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



Dimensions

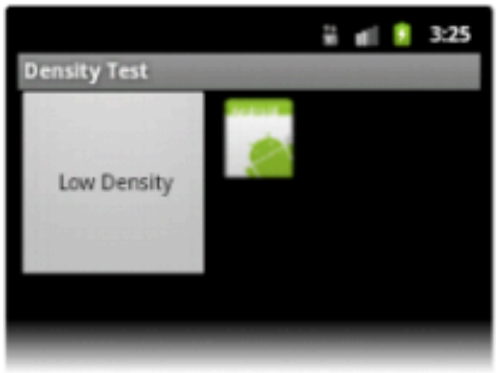
- S'appliquent à la **largeur** ou **hauteur** des vues
  - `android:layout_width`
  - `android:layout_height`
- S'expriment de manière
  - **Absolue** (unités de longueur ou nombre de pixels)
  - **Relative** (par rapport au groupe de vues parent)
  - **Pondérée** (poids relatif par rapport aux autres vues)

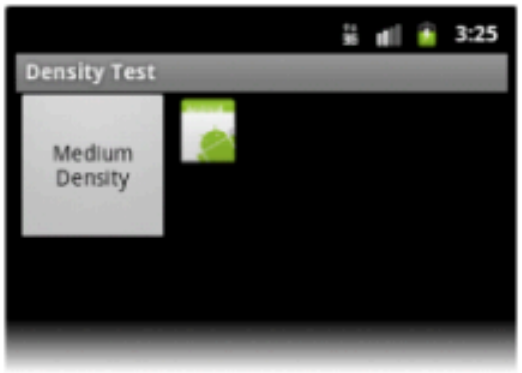
Dimensions absolues

- **Unités de longueur**
  - `mm` (millimètres)
  - `in` (inches) $\rightarrow$ 1 in = 25.4 mm
  - `pt` (points) $\rightarrow 1pt = \frac{1}{72}$ in)
- **Pixels**
  - `px` (pixels vrais)
  - `dp` (*density independant pixels*)
    - **Indépendant de la résolution** (1px = 1 dp en 160 dpi *mdpi*)
    - $px = dp * \frac{dpi}{160}$
  - `sp` (*scale dependant, density independant pixels*)
    - Recommandé pour les **tailles de texte**
    - **Remise à l'échelle automatique en fonction des préférences**
    - cf. *Paramètres* $\rightarrow$ *Affichage* $\rightarrow$ *Taille de la police*

px *Vs* dp

- Source : `http://developer.android.com`





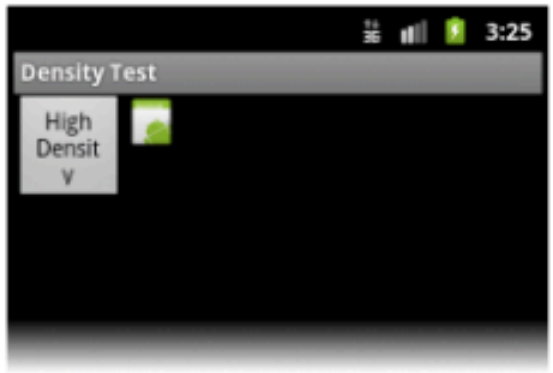
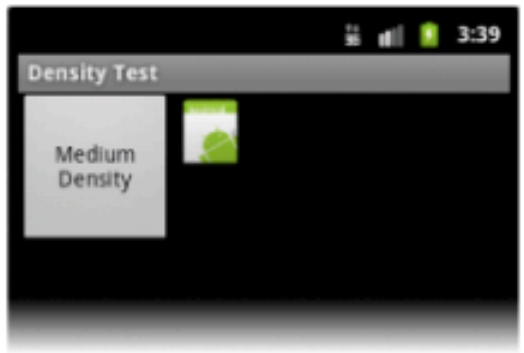


Figure 2. Example application without support for different densities, as shown on low, medium, and high-density screens.



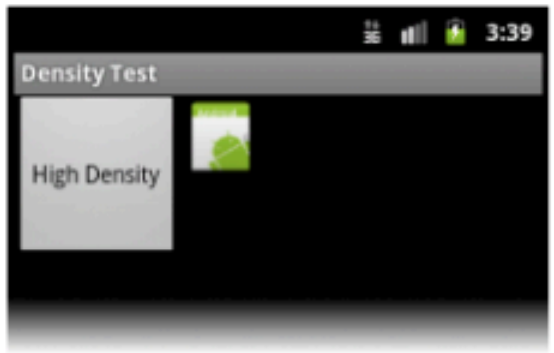


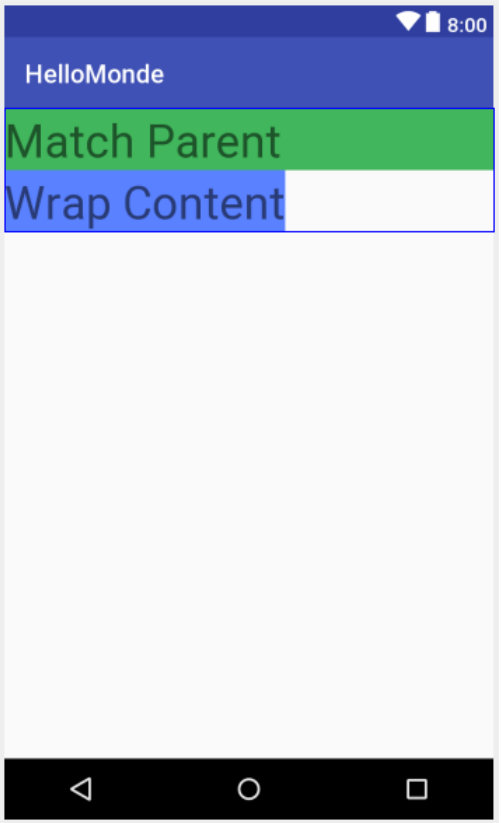
Figure 3. Example application with good support for different densities (it's density independent), as shown on low, medium, and high density screens.

Dimensions relatives

- `match_parent`
  - **requiert que la vue fasse la même taille que son parent**
- `wrap_content`
  - **Enveloppe le contenu**
- `fill_parent`
  - **Strictement équivalent à `match_parent`**
  - http ://developer.android.com/guide/topics/resources/layout-resource.html
- `match_constraint` **(dans le designer)**
  - Spécifique au `constraint_layout`, équivalent à 0dp
  - **la dimension de la vue sera aussi grande que permi par les contraintes**

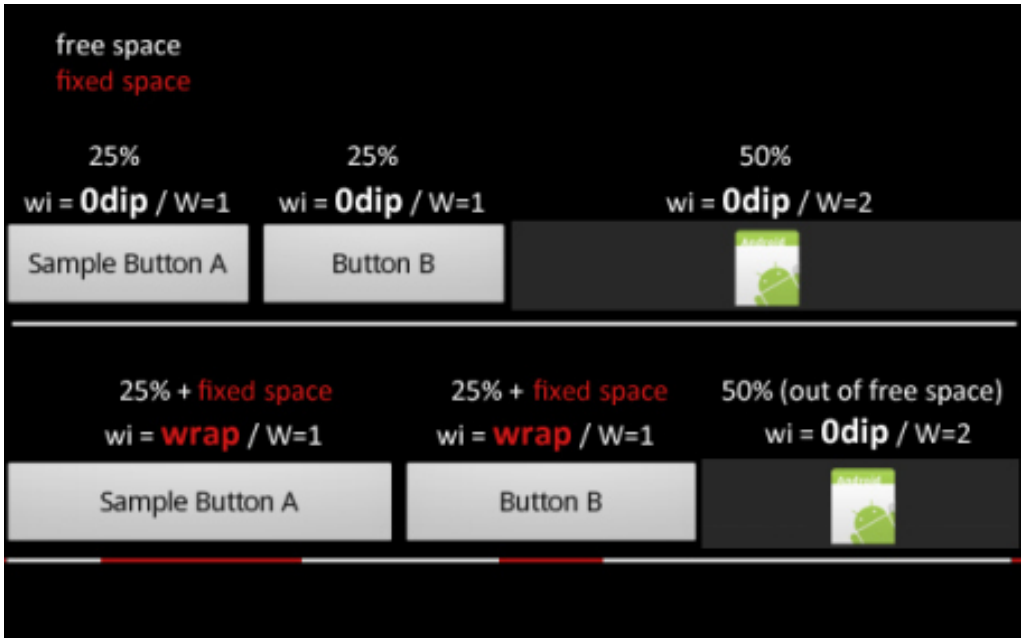
match/*wrap* par l'exemple

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:orientation="vertical">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/anotherColor"
        android:text="Match Parent"
        android:textSize="36sp"/>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorAccent"
        android:text="Wrap Content"
        android:textSize="36sp"/>
</LinearLayout>
```



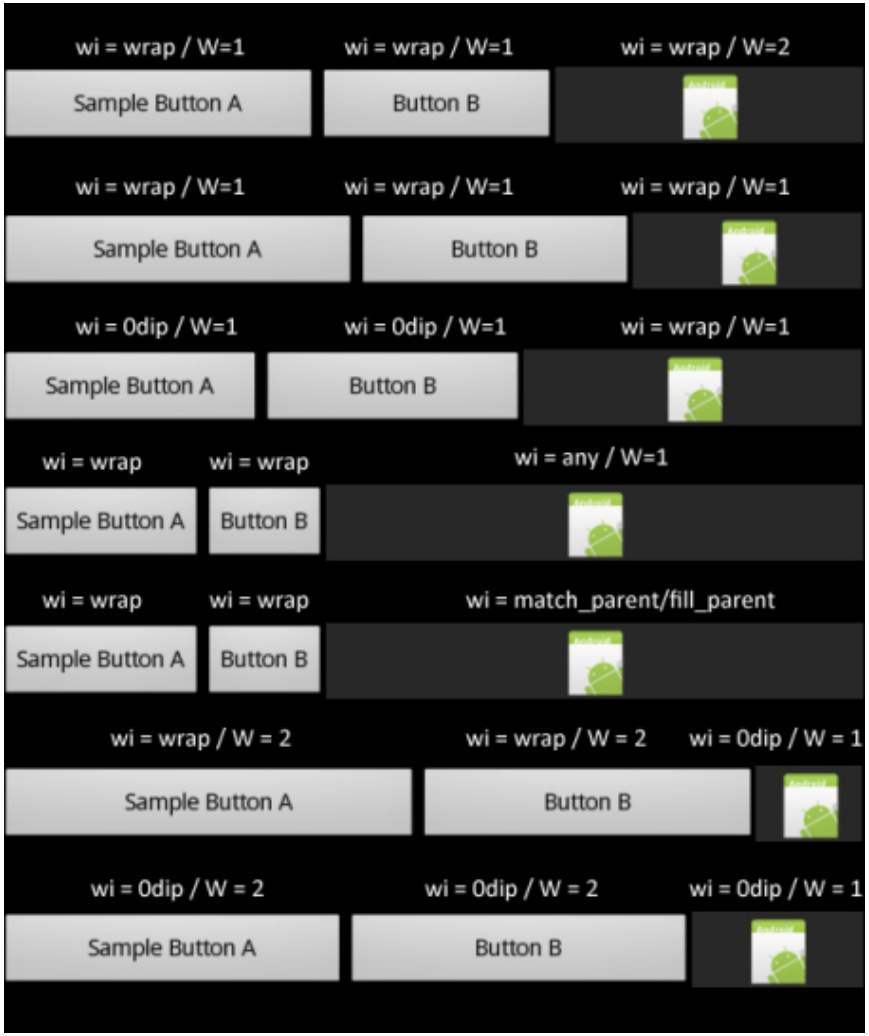
Poids relatif : attribut `android:layout_weight`

- **Partage de l'espace proportionnellement au poids**
  - **pourcentage** $\frac{poids}{somme\ des\ poids}$
- Remarques :
  - un poids à **zéro** fixe la dimension au strict nécessaire
  - `width` ou `height` = **`0dp`** → contenu non pris en compte
  - `width` ou `height` = **`wrap_content`** → contenu pris en compte



Poids relatif : attribut `android:layout_weight`

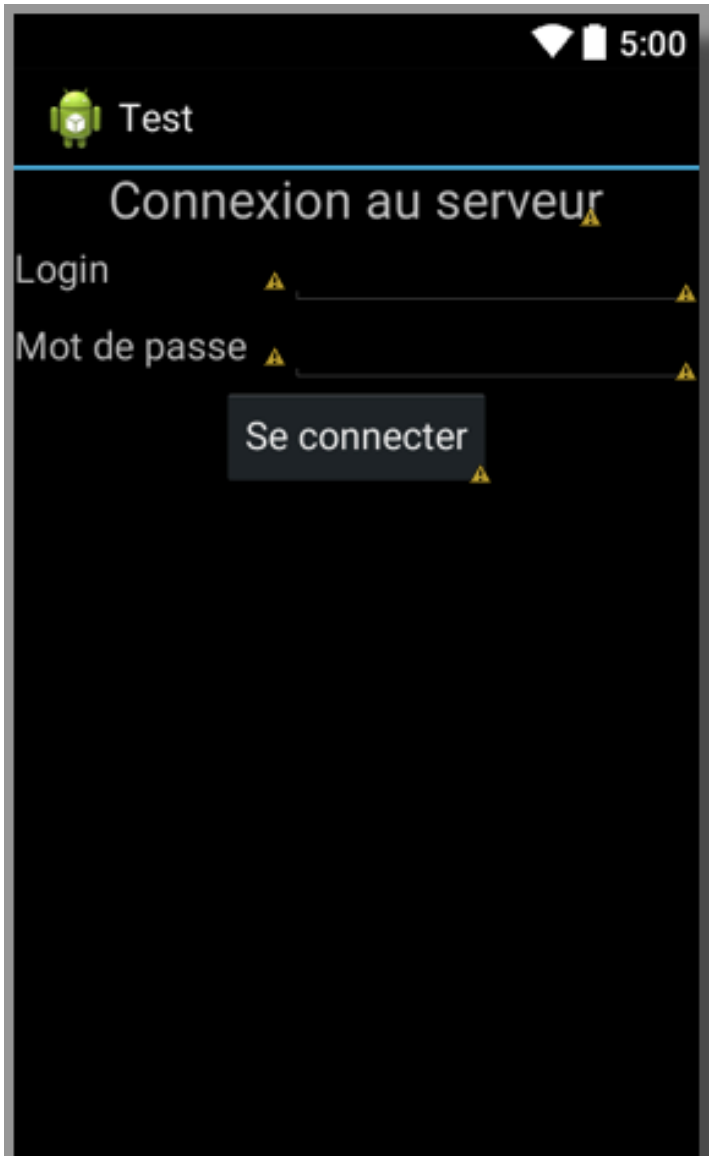
- Un autre exemple (source : `http://ugia.io`)



Alignement des vues et de leur contenu

- Deux attributs :
  - `android:layout_gravity` → placement **par rapport au parent**
  - `android:gravity` → placement du **contenu**
- Valeurs possibles
  - `top`, `bottom`
  - `left`, `right`
  - `center`, `center_vertical`, `center_horizontal`
  - `fill`, `fill_vertical`, `fill_horizontal`
- Valeurs **composables** avec |
  - `android:layout_gravity = "top|right"`

LinearLayout : un exemple un peu plus complexe



LinearLayout : un exemple un peu plus complexe

- LinearLayout (vertical)
  - `android:layout_width="fill_parent"`
  - `android:layout_height="wrap_content"`

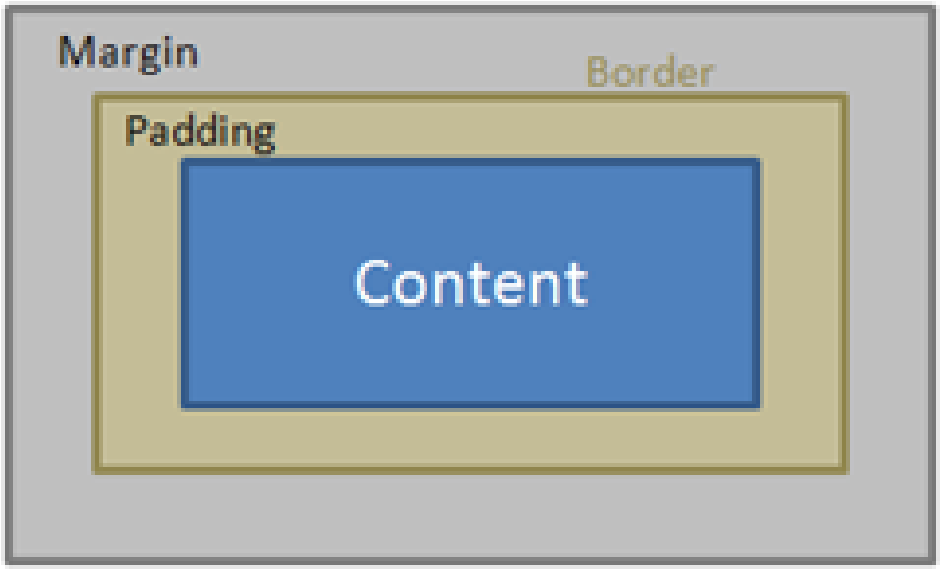
LinearLayout : un exemple un peu plus complexe

- `TextView` (width=wrap_content, height=wrap_content)
  - `android:layout_gravity="center_horizontal"`
  - `android:textSize="24sp"`
- `Button` (width=wrap_content, height=wrap_content)
  - `android:layout_gravity="center"`

LinearLayout : un exemple un peu plus complexe

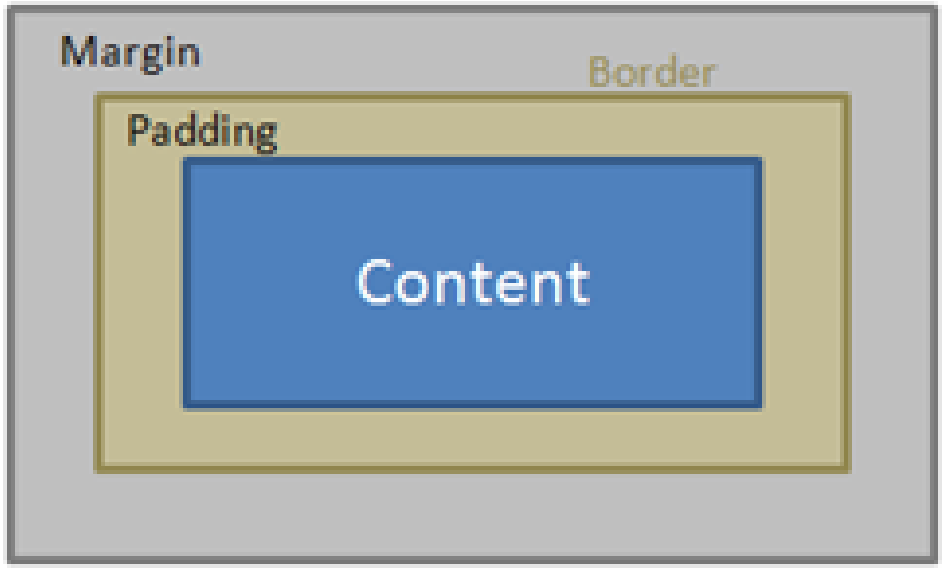
- `LinearLayout` (horizontal)
  - `TextView` (width=0dp, height= wrap_content, weight=2)
    - `android:layout_gravity="center_vertical"`
    - `android:textSize="18sp"`
  - `EditText` (width=0dp, height= wrap_content, weight=3)

Marges et remplissage



- **Marges** : espace supplémentaire **autour de la vue**
  - `android:layout_margin` (global)
  - `android:layout_marginTop`, `android:layout_marginBottom`
  - `android:layout_marginLeft`, `android:layout_marginRight`
  - `android:layout_marginStart`, `android:layout_marginEnd`

Marges et remplissage



- **Remplissage** : espace supplémentaire **à l'intérieur de la vue**
  - `android:layout_padding` (global)
  - `android:layout_paddingTop`, `android:layout_paddingBottom`
  - `android:layout_paddingLeft`, `android:layout_paddingRight`

Constraint Layout

Les éléments graphiques doivent posséder au moins deux contraintes :

- Une verticale
- Une horizontale

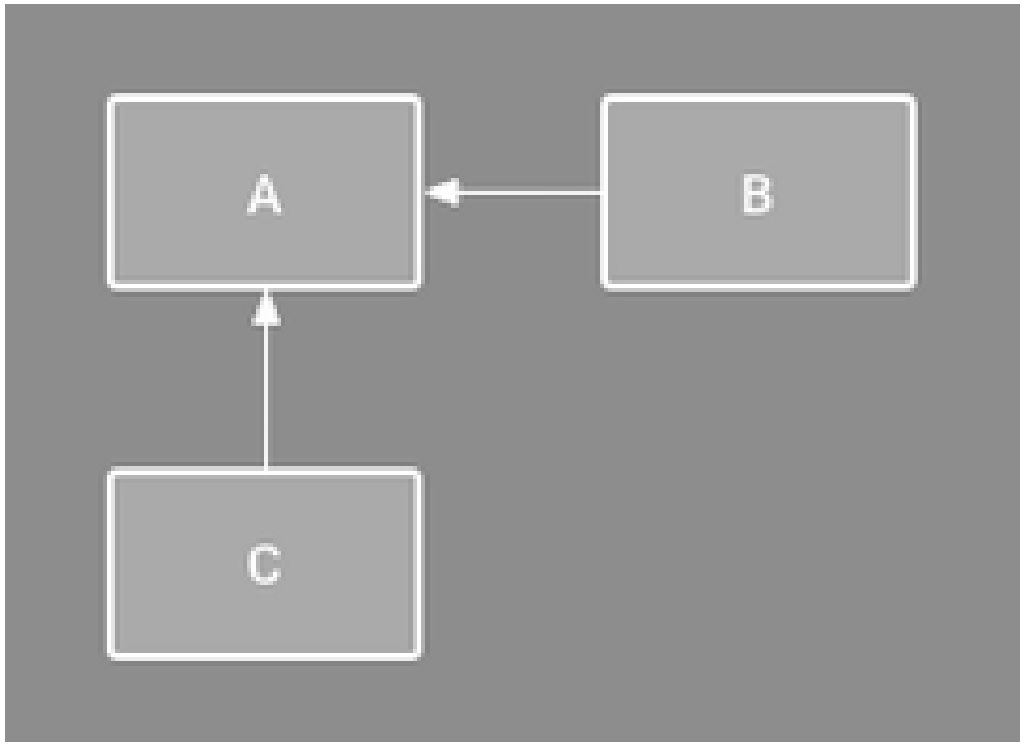
Il est possible de mettre des contraintes entre un élément et :

- Son parent
- D'autres éléments graphiques
  - Nécessité d'identifier les éléments graphiques.

Identification des vues

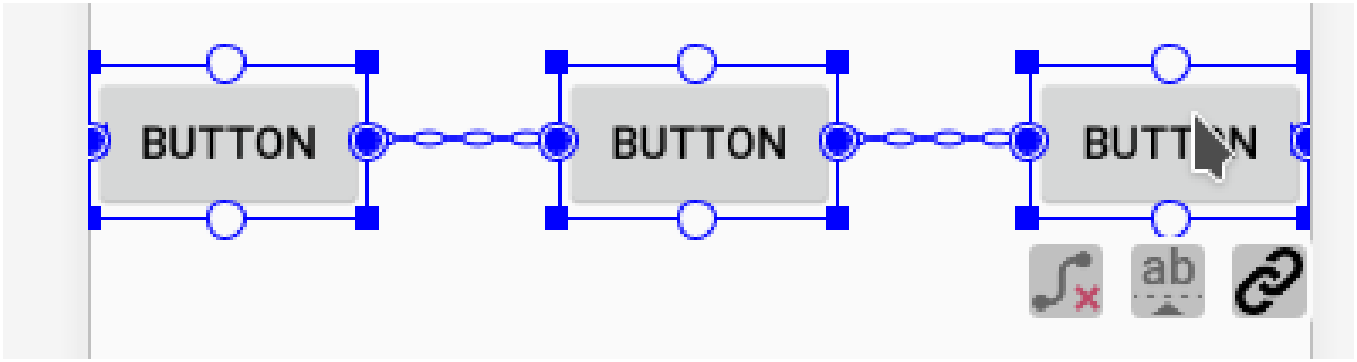
- Placement relatif $\rightarrow$ **mécanisme d'identification des vues**
  - Remarque : ce n'est pas la seule raison
- **Création** de l'identifiant d'une vue
  - `android:id="@+id/nom"`
- **Référence** à une vue via son identifiant
  - `"@id/nom"`
- Remarque : signe + requis à la 1^{re} apparition de l'identifiant
  - En général il s'agit de la création, mais pas toujours ...

Layout invalide car incomplet



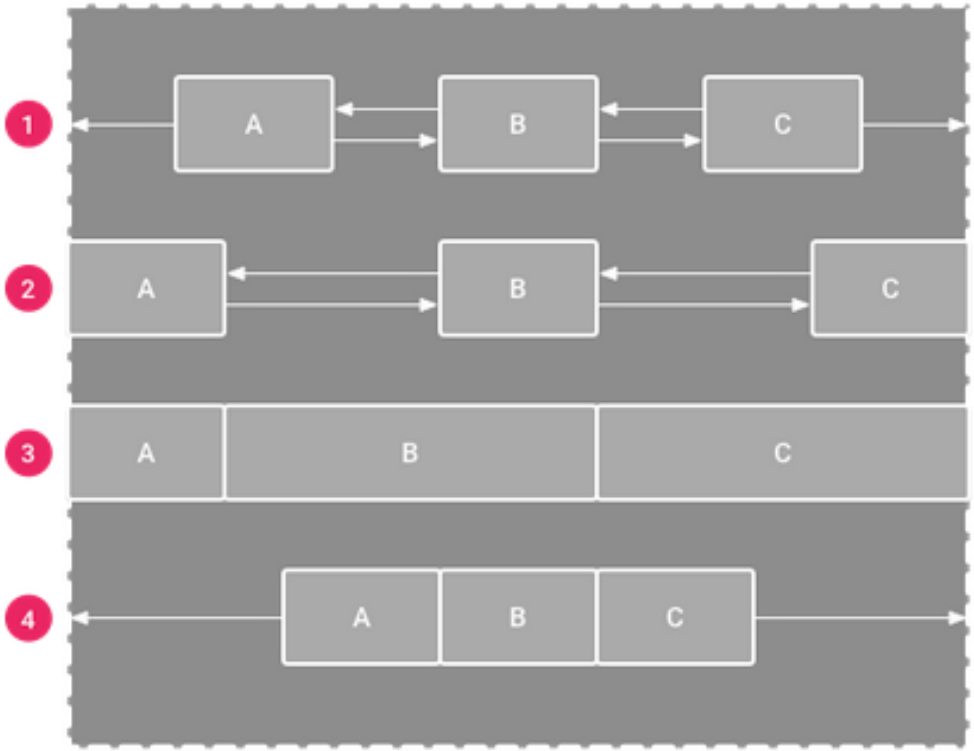
Il est préférable de créer les contraintes en utilisant les outils d'alignement

Ou en créant des chaînes



Chains

Une façon de lier les éléments graphiques et de les répartir



RelativeLayout

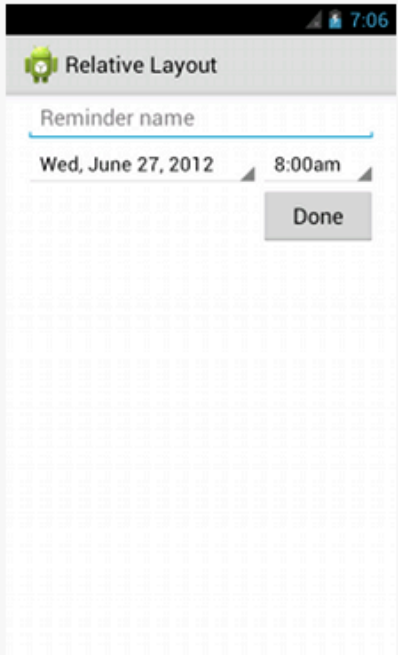
- **Positionnement relatif des vues**
  - **Entre elles**
  - **Par rapport au parent**

- Source : `http://developer.android.com`

RelativeLayout : exemple

- source : `http://developer.android.com`

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



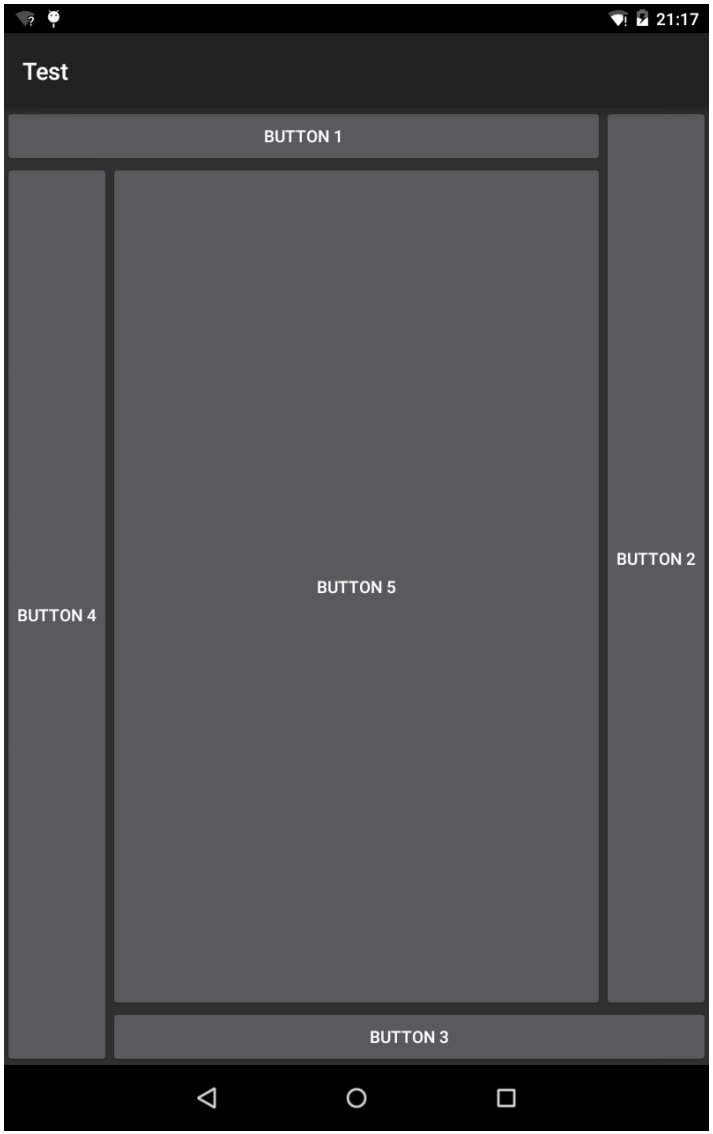
RelativeLayout : principaux attributs

- **Identifiants** comme valeur
  - `android:layout_above`, `android:layout_below`
  - `android:layout_alignLeft`, `android:layout_alignRight`
  - `android:layout_alignTop`, `android:layout_alignBottom`
  - `android:layout_alignStart`, `android:layout_alignEnd`
  - `android:layout_toLeftOf`, `android:layout_toRightOf`
  - `android:layout_toStartOf`, `android:layout_toEndOf`

RelativeLayout : principaux attributs

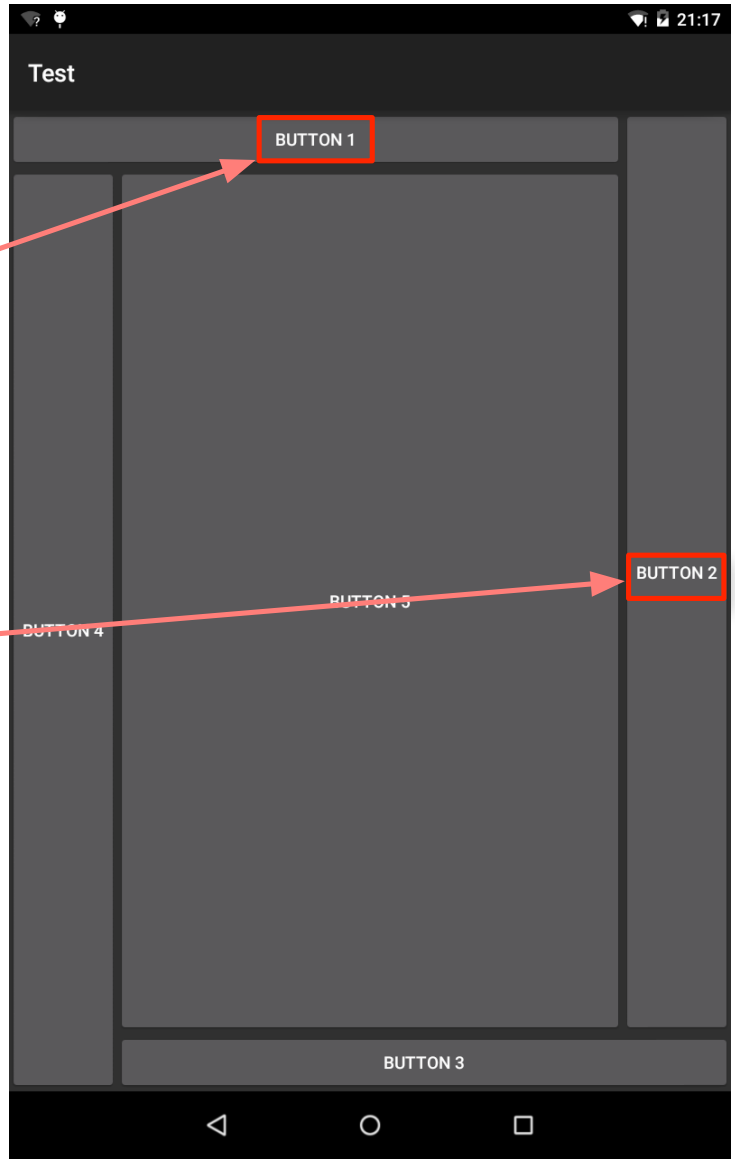
- **Booléen** (à `true` comme valeur)
  - `android:layout_alignParentTop`
  - `android:layout_alignParentBottom`
  - `android:layout_alignParentLeft`
  - `android:layout_alignParentRight`
  - `android:layout_alignParentStart`
  - `android:layout_alignParentEnd`
  - `android:layout_centerHorizontal`
  - `android:layout_centerVertical`
  - `android:layout_centerInParent`

RelativeLayout : un autre exemple



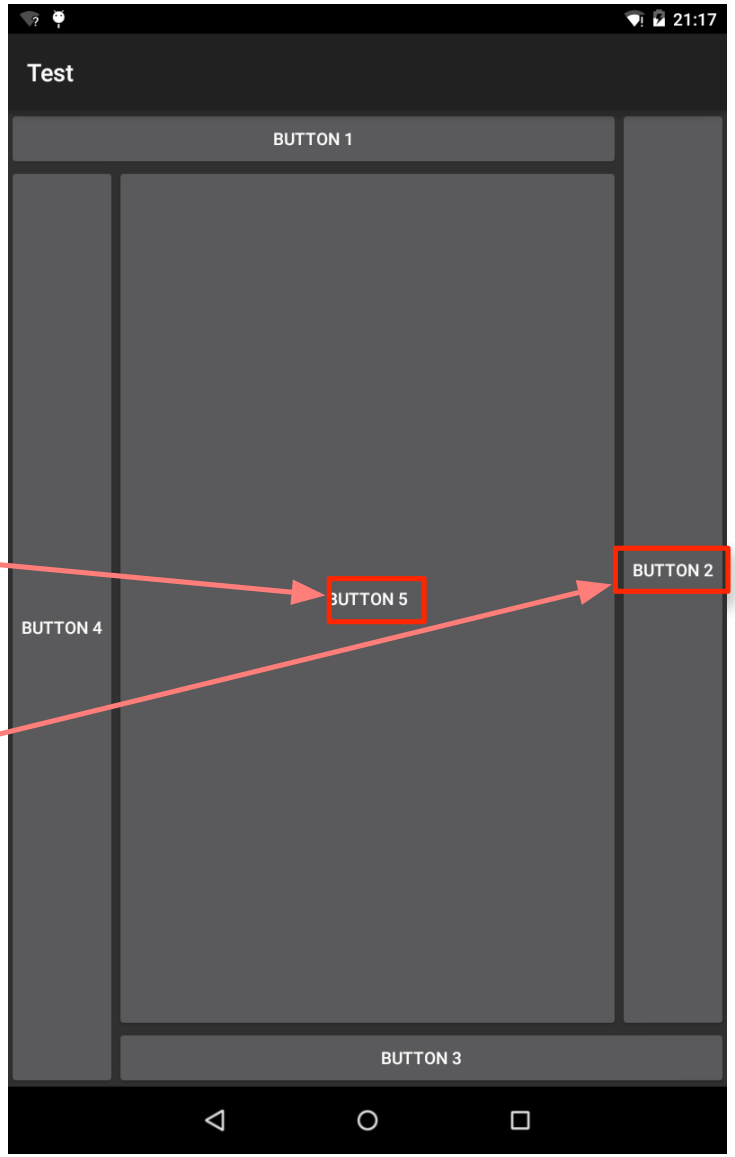
RelativeLayout : un autre exemple

```
<Button
  android:id="@+id/btn1"
  android:text="button 1"
  android:backgroundTint="#FFFFFF"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentTop="true"
  android:layout_toLeftOf="@+id/btn2">
</Button>
```



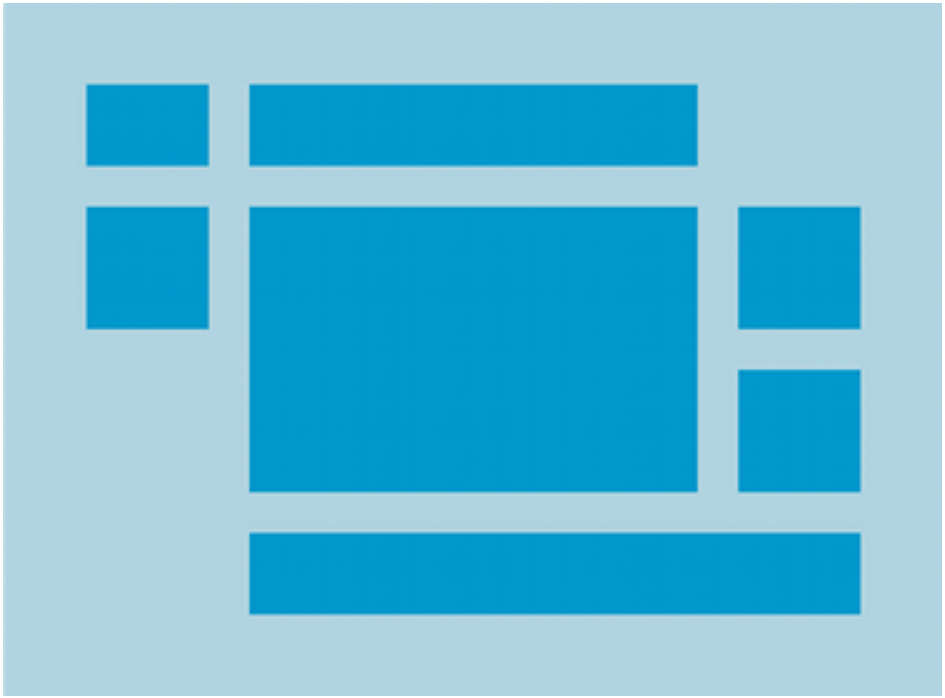
RelativeLayout : un autre exemple

```
<Button
  android:id="@+id/btn5"
  android:text="button 5"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:layout_below="@+id/btn1"
  android:layout_above="@+id/btn3"
  android:layout_toRightOf="@+id/btn4"
  android:layout_toLeftOf="@+id/btn2">
</Button>
```



TableLayout

- **Positionnement des vues en lignes/colonnes**
  - Cellules de **même taille** (la plus grande dimension)
  - **Possibilité de laisser des cellules vides**



- Source : `http://developer.android.com`

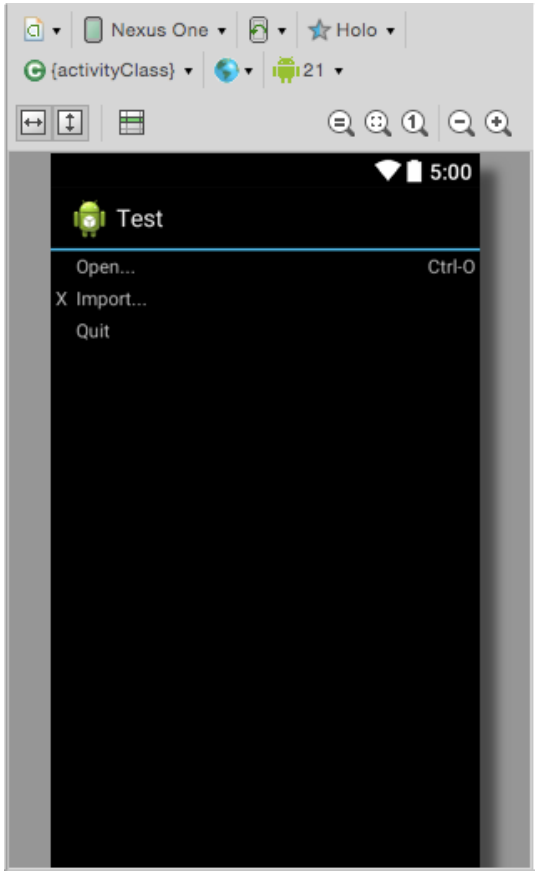
TableLayout : exemple

- source : `http://developer.android.com`

activity_main.x | AndroidManifest | MainActivity.ja | activity_main.x

```xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">

    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Open..."
            android:padding="3dip" />
        <TextView
            android:text="Ctrl-O"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
        <TextView
            android:text="X"
            android:padding="3dip" />
        <TextView
            android:text="Import..."
            android:padding="3dip" />
    </TableRow>


    <TableRow>
        <TextView
            android:layout_column="1"
            android:text="Quit"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

Graphical Layout | activity_main.xml

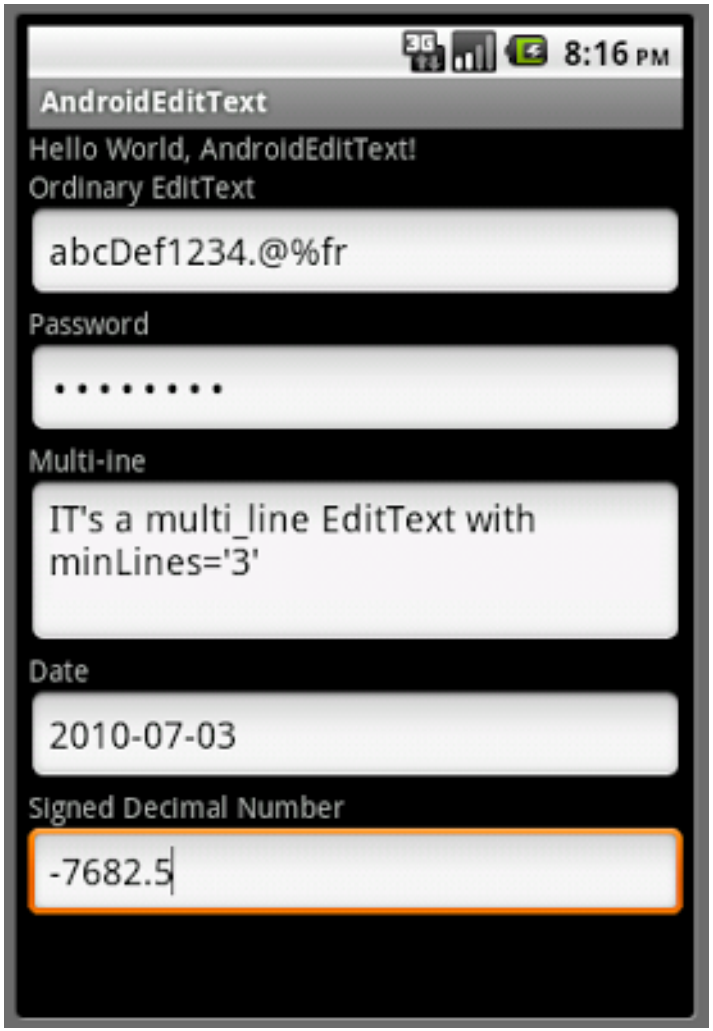


TableLayout/TableRow : attributs

- **TableLayout**
  - `android:collapseColumns`
    - Numéro(s) (≥ 0) de colonne(s) à rendre invisible
  - `android:shrinkColumns`
    - Numéro(s) (≥ 0) de colonne(s) à réduire au maximum
  - `android:stretchColumns`
    - Numéro(s) (≥ 0) de colonne(s) à étirer au maximum
- **TableRow**
  - `android:layout_column`
    - Numéro de colonne où installer la vue
  - `android:layout_span`
    - Nombre de colonnes sur lesquelles la vue doit s'étaler

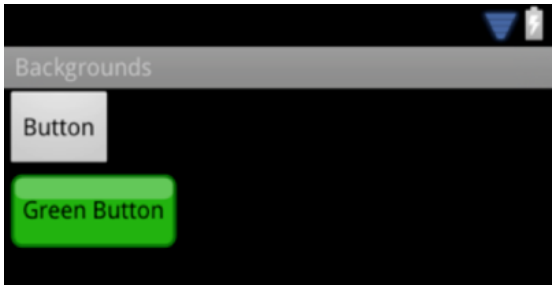
Composants graphiques utiles

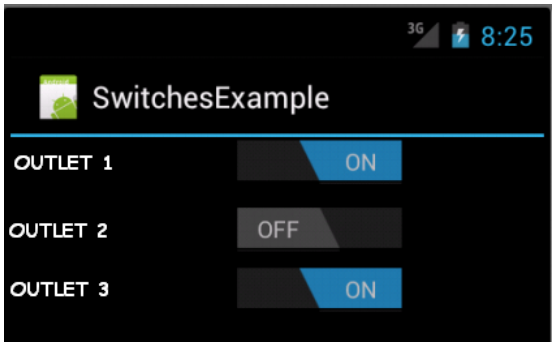
- `TextView`, `EditText`

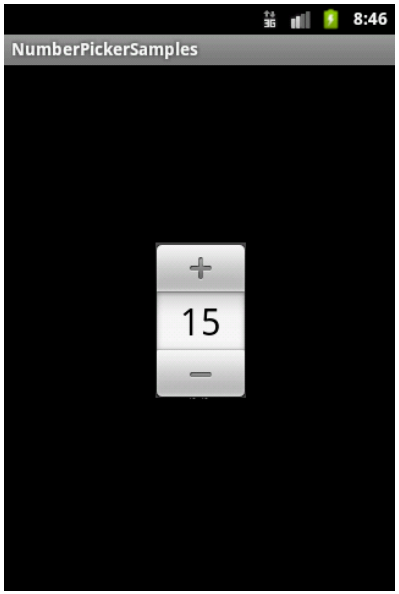


Composants graphiques utiles

- `Button`, `Switch`, `NumberPicker`

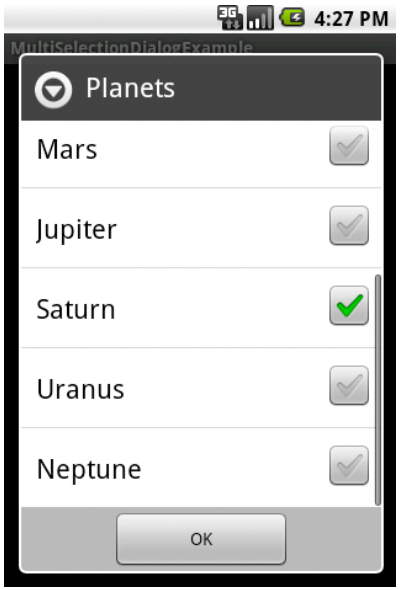


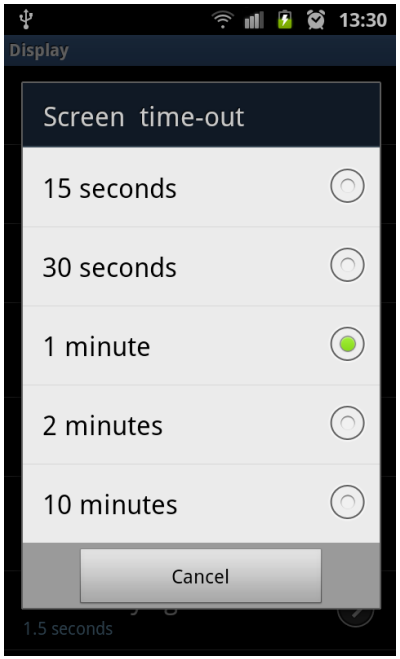


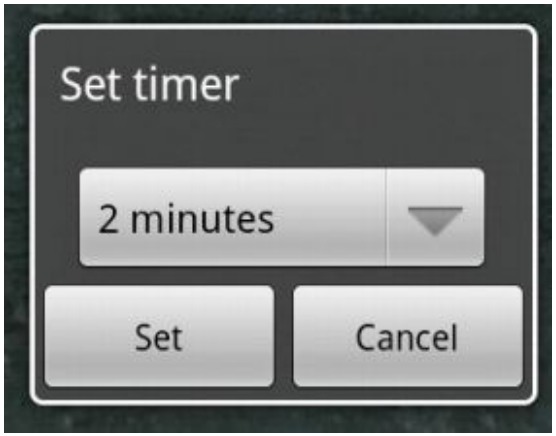


Composants graphiques utiles

- `CheckBox`, `RadioGroup`, `Spinner`

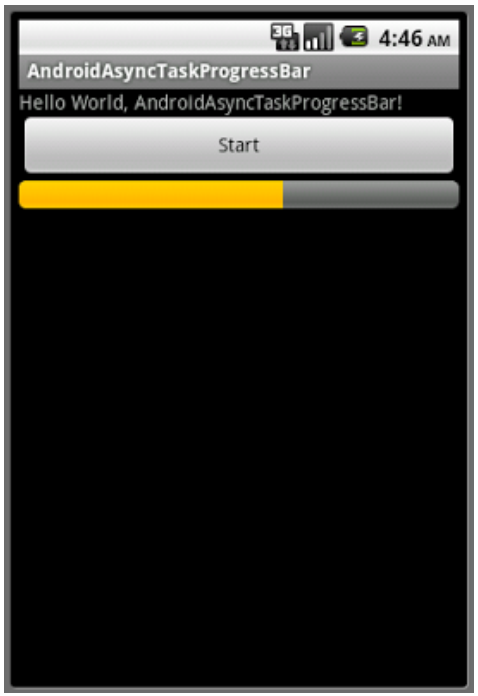


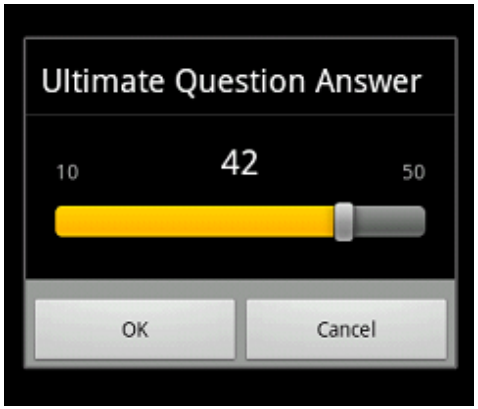


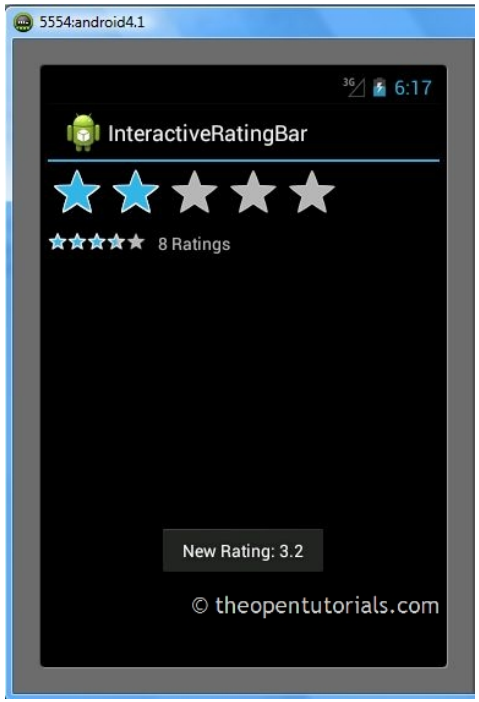


Composants graphiques utiles

- `ProgressBar`, `SeekBar`, `RatingBar`

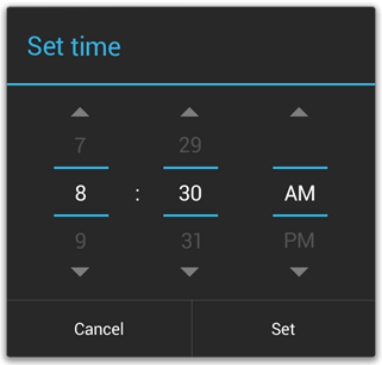


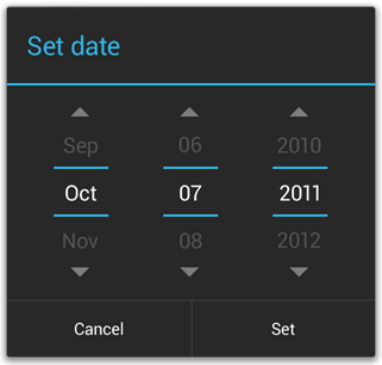


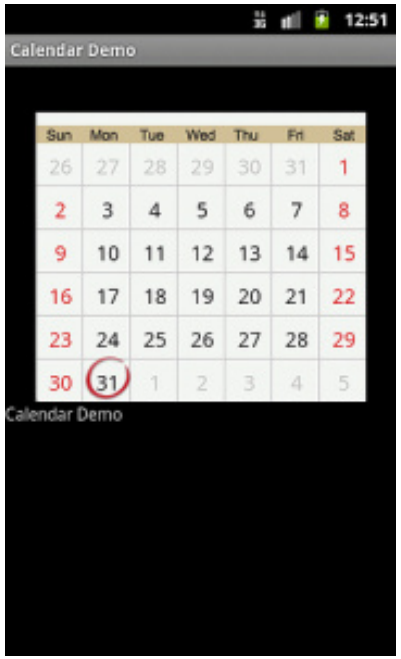


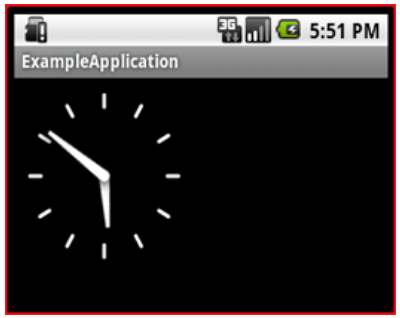
Composants graphiques utiles

- `TimePicker`, `DatePicker`, `Calendar`, `AnalogClock`









Fin !