ANDROID ALKALMAZÁSFEJLESZTÉS

Felhasználói felület megismerése

Különböző felbontások támogatása

sicz.mj[tekercs]gmail.com

Sicz-Mesziár János

2013. június 13.

Layouts

Ősosztály: ViewGroup

Olyan tárolók, melynek gyermeke lehet:

- View / Widget
- Tároló / Layout

Deprecated

~~TableLayout~~
~~AbsoluteLayout~~
~~TabLayout~~
~~Gallery~~

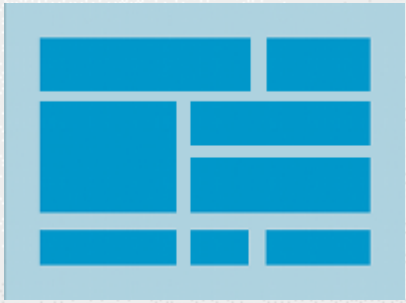
Speciálisak

ScrollView
HorizontalScrollView
ListView
GridView
ViewPager
ViewFlipper
…

LinearLayout
- UI elemek egymás után.
- Horizontal vagy Vertical.
- Méretezés arányokkal.

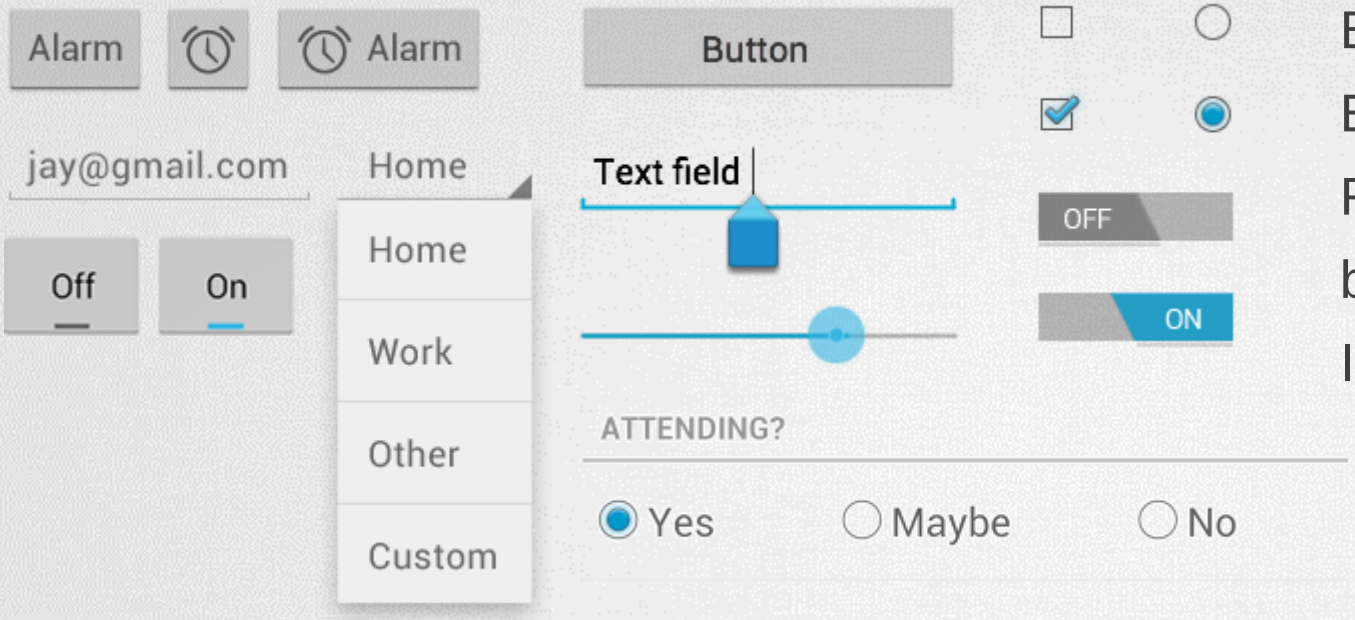
RelativeLayout
- Egymáshoz képest adhatjuk meg pozíciót.
- Leghatékonyabb layout.

FrameLayout
- Minden gyermek a bal felső sarokhoz igazodik



GridLayout
- Mozaik szerű elrendezés
- Android 4.0-tól
- != GridView!
- Bővebben itt!

View / Widget



Button, TextView, EditText, Checkbox, Radio button, Toogle button, Spinner, ImageView, ….

```
<Button
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/txt"
    ...
    />
```

Egyedi azonosító, hivatkozás:

- Java: `R.id.button1`
- XML: `@id/button1`

Méret – szélesség, magasság:

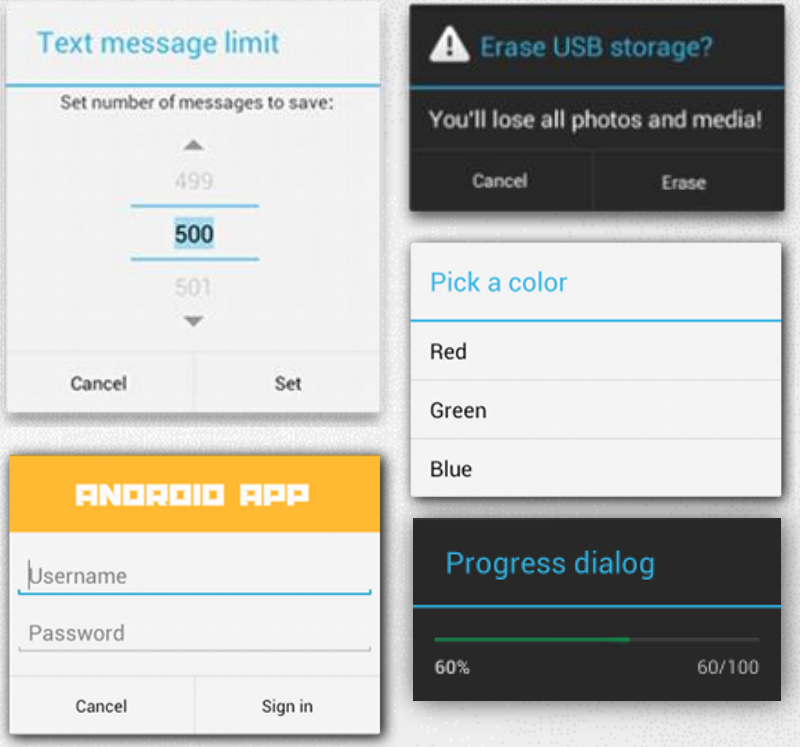
- `MATCH_PARENT`
- `WRAP_CONTENT`
- `[SIZE][dp|px]`

Resource elérés, res/strings.xml:

- `<item name="txt">Gomb</item>`

Dialogs

- Ősosztály: Dialog
- Az aktuális Activity / Fragment előtt jelenik meg
- Beépített gombok:
  - PositiveButton
  - NegativeButton
  - NeutralButton

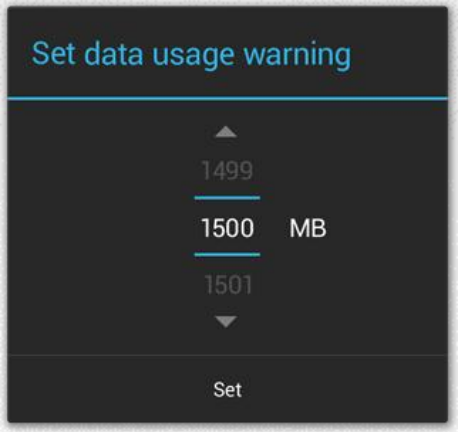


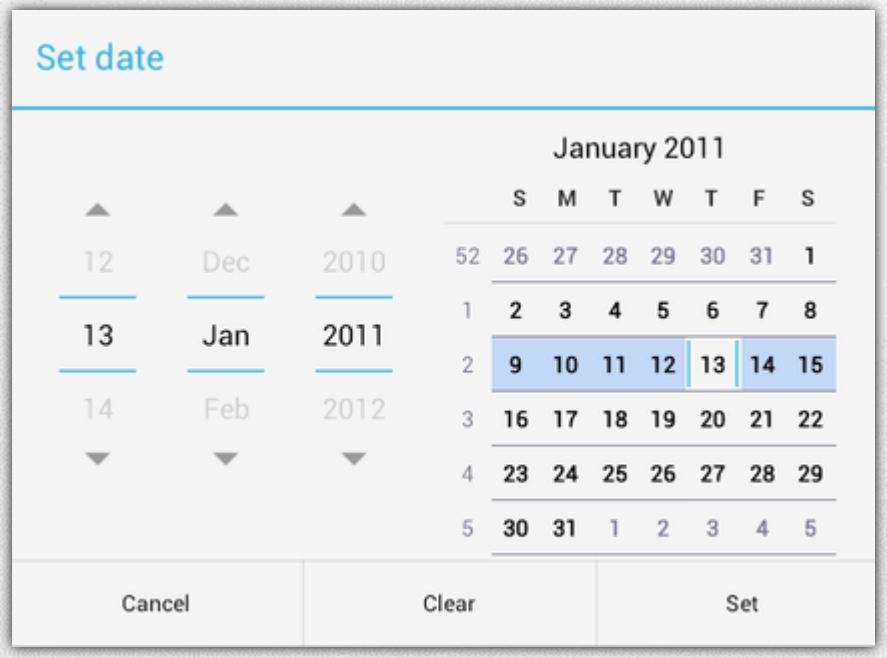

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
builder.setMessage(R.string.dialog_message)
       .setTitle(R.string.dialog_title);
AlertDialog dialog = builder.create();
```

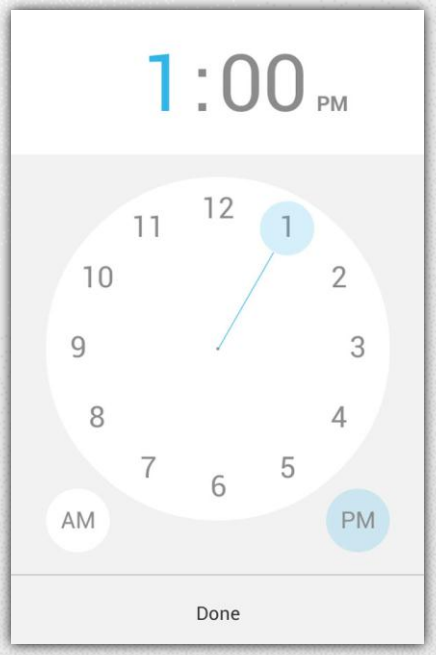
Pickers

Lényegében funkció specifikus Dialog:

- DatePicker
- TimePicker
- Bővebben: http://developer.android.com/guide/topics/ui/controls/pickers.html







Notifications

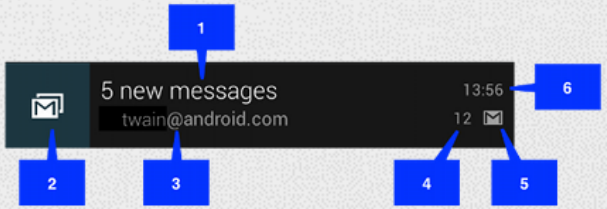
Toast notification

- Egyszerű szöveges mező, felbukkanó buborékban
-
```
Toast.makeText(
        getApplicationContext(),
        "Buborék",
        Toast.LENGTH_SHORT
).show();
```

Status Bar notification

- Egy értesítési terület.
- Jelly Bean óta kibontható.
  - Notifications: Egyszeri értesítés, „Clear” gomb hatására törölhető
  - Ongoing: folyamatban lévőekről értesítés
    Pl.: zenelejátszó. Nem törölhető.

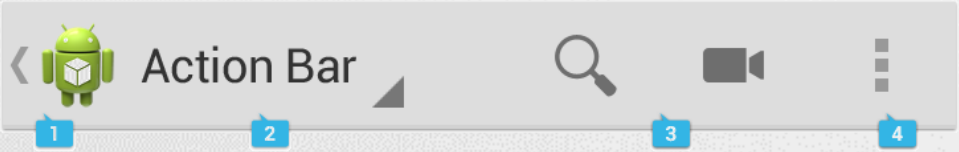
1. Content title
2. Large icon
3. Content text
4. Content info
5. Small icon
6. Time

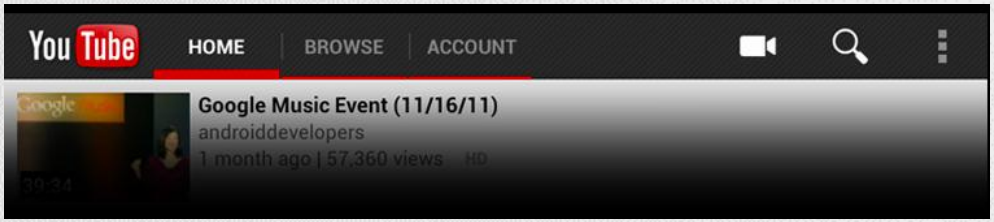


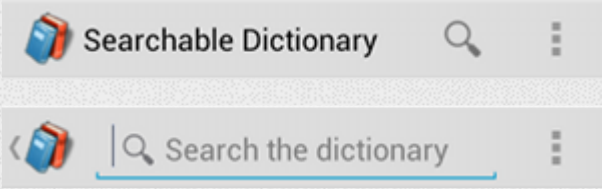
ActionBar

Android 3.0 (API level 11) óta érhető el:

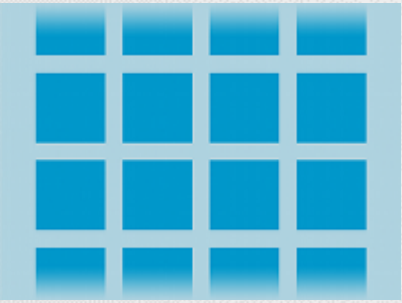
- Kitüntetett hely az alkalmazás nevének és ikonjának - brand
- Navigáció: beépített Tab kontroll a Fragment váltásokhoz
- Jellemző funkciók (Pl.: keresés, megosztás, létrehozás) megszokott helyen
- Bővebben: http://developer.android.com/guide/topics/ui/actionbar.html







Adapter views



ListView, GridView, Spinner, ViewPager, ~~Gallery~~

Az adatokat egy adapteren keresztül biztosítjuk az UI számára.

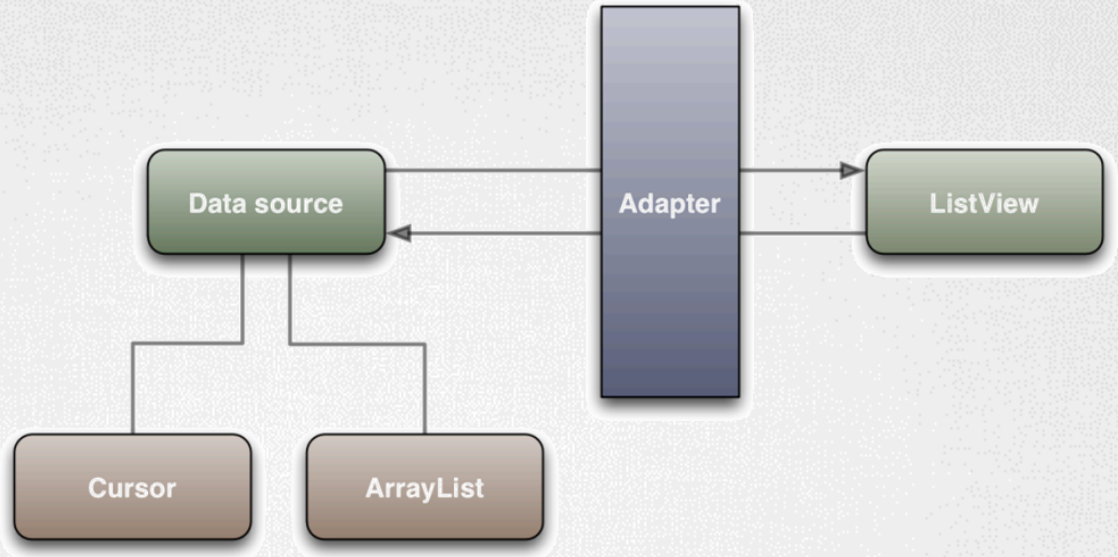
- Előre implementált adapter (Pl.: ArrayAdapter)
- Mi implementáljuk (BaseAdapter leszármazott)

Módszer előnyei

- Az adatok tárolási módja nem meghatározott
  Pl.: SD kártya, SQLite adatbázis, internet, …
  Vagy: lista, tömb, hashmap, …
- UI szétválasztva az adattól
- Optimális erőforrás felhasználás
- Nagy adatmennyiség kezelése
  (akár >10E listaelem kezelése)

Adapter views (2) hogyan működik?

- Ősosztály: BaseAdapter
- Implementációk:
  - ArrayAdapter,
  - SpinnerAdapter,
  - CursorAdapter,
  - …



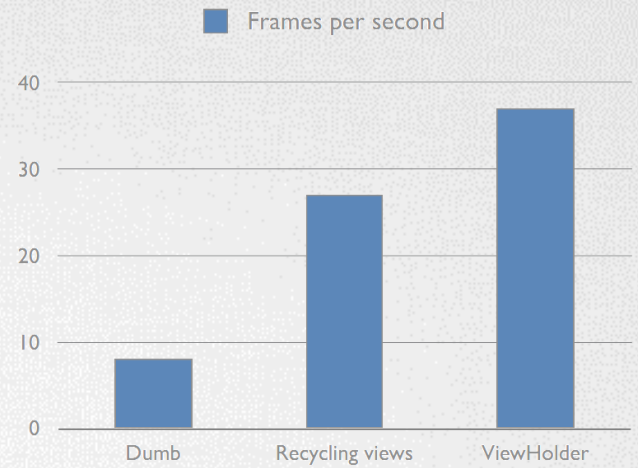

```
public class CustomAdapter extends BaseAdapter{
        int getCount() {}
        Object getItem(int position) {}
        long getItemId(int position) {}
        View getView(int position, View convertView, ViewGroup parent) {}
}
```

Adapter views (3) hogyan működik jól?

Probléma:

- Minden pozicióban:

  `Adapter.getView();`
- Minden esetben új View objektumot létrehozni költséges!
- Több ezer elem esetén?



Nem látható UI elemek újrahasznosítása:

```
public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;

        if (convertView == null) {
                convertView = mInflater.inflate(R.layout.list_item, null);
                holder = new ViewHolder();
                holder.text = (TextView) convertView.findViewById(R.id.text);
                convertView.setTag(holder);
        } else
                holder = (ViewHolder) convertView.getTag();

        holder.text.setText(DATA[position]);
        return convertView;
}
```

XML drawables

- Egyszerűbb alakzatokat, rajzokat, képeket, viselkedéseket leírhatunk XML-ben is
- Példák
  - Layer-list
    Több kép együttes kezelése rétegekben.

  - State-list
    Különböző állapotokhoz rendelt grafikai elemek.
    Például focused, pressed, hover, stb…
  - Level-list
    Különböző szintekhez rendelt grafikai elemek.
    Például Wi-Fi, fényerő vagy akkumulátor állapotokat ábrázoló képek.

  - Shape
    Egy egyszerű alakzat (rectangle, oval, …) leírása.
    Kitöltési szín, vonal szín, sarkok kerekítése, …

Styles / Themes

- Stílusok alakíthatóak ki, melyeket nagy hatékonysággal lehet újrahasznosítani, és egységesen kezelni.
- res/values/styles.xml

```
<resource>
        <style name="MyStyle" parent="@android:style/Widget.Button">
                <item name="android:background">#556677</item>
                …
        </style>
</resource>
```

- res/layout/activity_main.xml

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gomb"
        style="@style/MyStyle"
        />
```

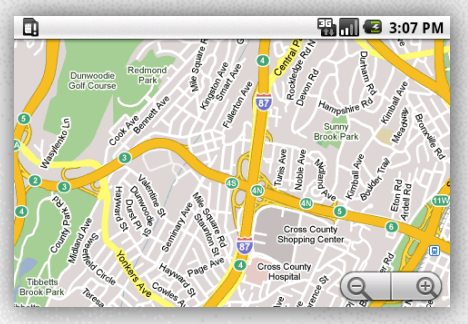
Különböző felbontású készülékek támogatása

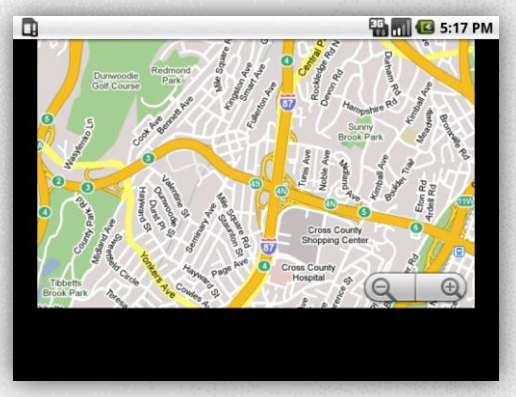
Problémák

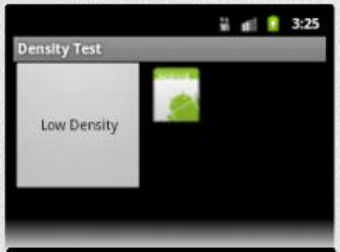
240 dpi (high)

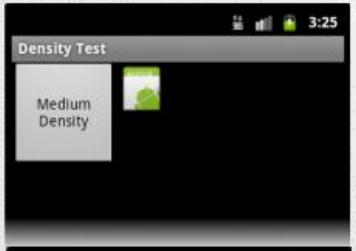
160 dpi (medium)

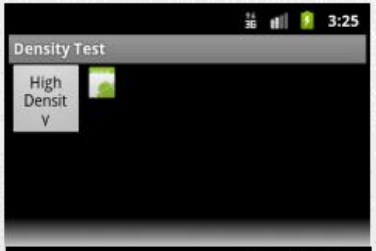
240 dpi, landscape











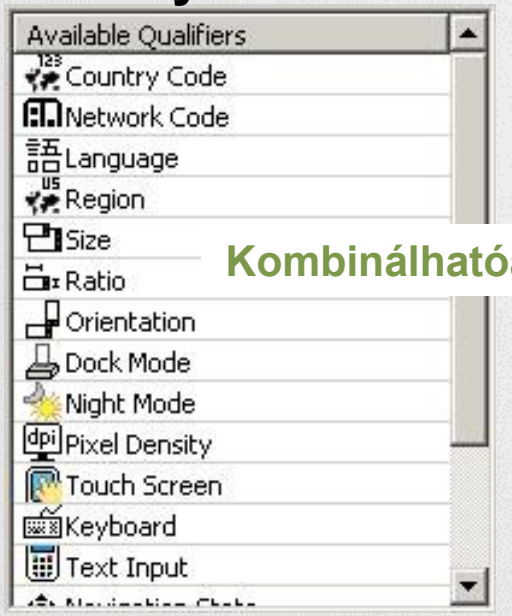
API Level 6+
Launcher Icon

API Level 4+
Launcher Icon

Qualifiers (minősítők)

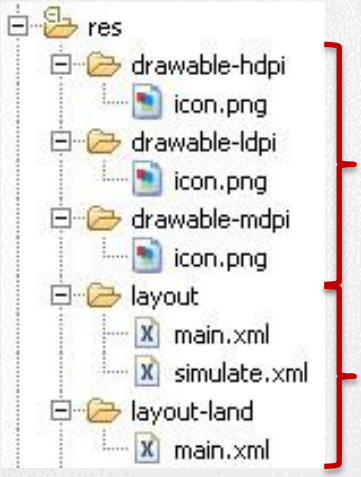
- Különböző esetekre különböző megközelítés szükséges.
- Sok IF és SWITCH helyett minősítőket definiáltak.
- Automatikus kiértékelés, aktuálisan jellemző mappából dolgozik.

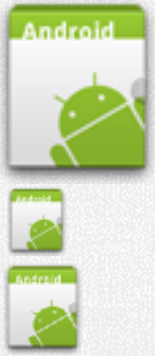
Néhány minősítő:



Kombinálhatóak!

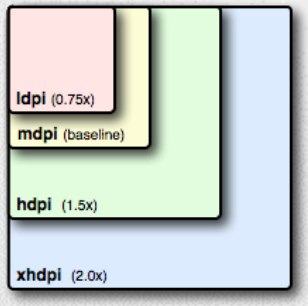
Gyakorlatban:





Ugyanaz a kép 3 különböző méretben

Ugyanazon felület különböző leírása „portrai„ és „landscape”

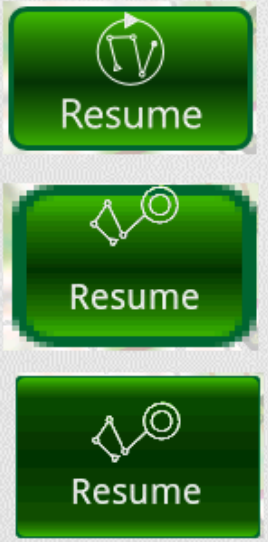


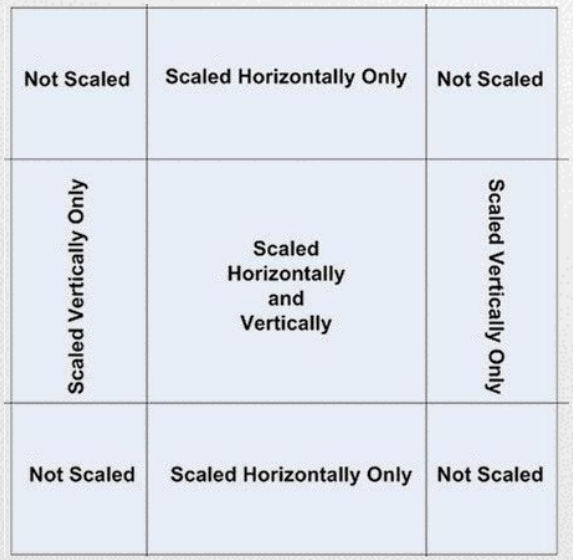
Ikon méretek

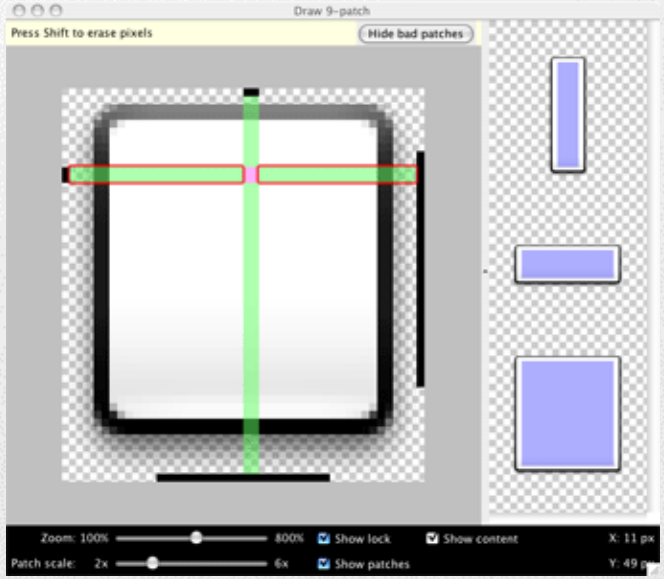
- 36x36 for low-density
- 48x48 for medium-density
- 72x72 for high-density
- 96x96 for extra high-density

9-Patch

- Dinamikus tartalom esetén, ha egyéni hátteret használunk akkor az eltorzulhat. Pl.: egy gombnál
- Speciális PNG fájllal meghatározhatjuk mely részeket nyújthatjuk meg a mintaképen. (bal oldal és fent) Szélén fekete pixelekkel jelöljük meg ezt a nyújtható területet.
- Jobb oldalt és lent a kitöltési területet jelölhetjük meg.
- Mindig ***.9.png** kiterjesztésű
- Van hozzá eszköz: [SDK path] / tools / draw9patch.bat







Density Independent Pixel (DIP, DP)

DP vagy DIP (**D**ensity-**I**ndependent **P**ixel)

- Egy virtuális pixel-egység, sűrűség-független képpont.
- 160dpi felbontású készüléken 1dp = 1px.
- Eltérő pixelsűrűség esetén automatikusan átváltja az alábbi módon:

```
pixels = dips * (density / 160)
```

Ez alapján néhány példa:

- 160dpi felbontás esetén, 10dp = 10px
- 240dpi felbontás esetén, 10dp = 15px

SP vagy SIP (**S**cale-**I**ndependent **P**ixel)
Mint a DP, csak szöveg esetén használatos.

```
android:textSize="16sp"
```

Responsive design