# Mobil- és webes szoftverek

Dr. Ekler Péter

ekler.peter@aut.bme.hu

BME AUT
Department of Automation and Applied Informatics

# ZH és PótZH időpontok

- ZH időpont:
  - 2017. 11. 03. Péntek 14-16
  - QI, IB028 és QBF12-es termek (terembeosztás később)
- PótZH időpont:
- 2017. 11. 28. Kedd 8-10
- IB028

# Miről volt szó az előző órán?

- Activity ál
- Activity Ba
- Felhasznál
- Erőforrás
- View/Viev
- Menü kez
- Stílusok, t

# Miről volt szó az előző órán?

- Activity állapot elmentése
- Activity Back Stack
- Felhasználói felület alapok
- Erőforrás típusok
- View/ViewGroup-ok
- Menü kezelés, Toolbar
- Stílusok, témák

# Hogy is volt?

- Magyarázza el a fordítás mechanizmusát!
- Egy Android alkalmazás milyen komponensekből épülhet fel?
- Mi a Service komponens?
- Miket kell tartalmaznia a manifest állománynak?
- Az Activity callback életciklus-függvények felüldefiniálásakor meg kell-e hívni kötelezően az ős osztály implementációját?
- Ha A Activity-ből átváltunk B Activity-re, milyen sorrendben hívódnak meg az életciklus függvények?
- Magyarázza el az Activity Back Stack működési elvét!
- Mit nevezünk ViewGroup-nak?
- Mit nevezünk erőforrás minősítőnek?

# Tartalom

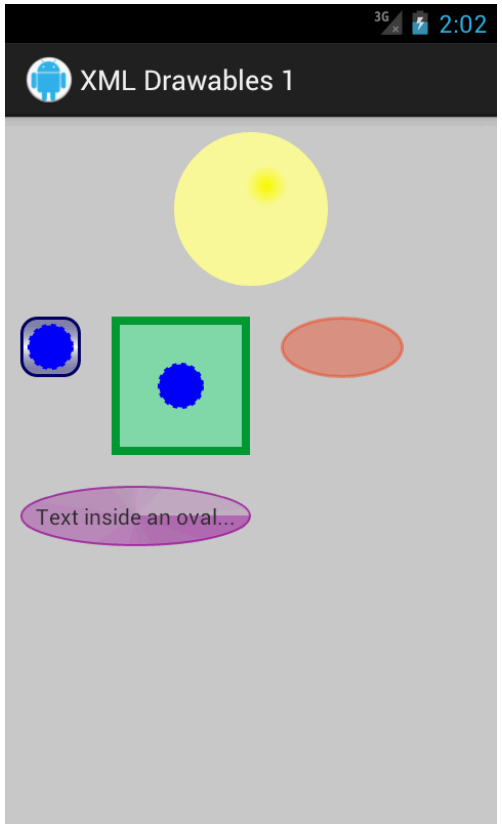
- Grafikai erőforrások
- Animációk
- Élő háttérkép, Widget
- Fragmentek
  - Statikus és dinamikus csatolás
  - Worker fragment
  - DialogFragment

# GRAFIKAI ERŐFORRÁSOK XML-BEN

# Grafikai erőforrások 1/2

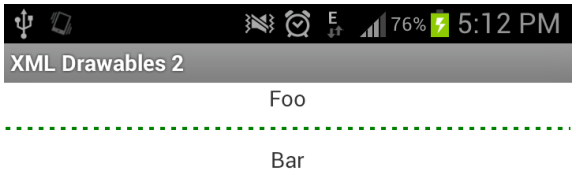
- Grafikai elemek, hátterek is leírhatók XML-ben

```
<?xml version="1.0" encoding="utf-8"?>
<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="oval">

  <gradient
    android:type="radial"
    android:gradientRadius="20"
    android:centerX=".6"
    android:centerY=".35"
    android:startColor="#FFFF00"
    android:endColor="#FFFF99" />

  <size
    android:width="100dp"
    android:height="100dp"/>
</shape>
```



# Grafikai erőforrások 2/2

- Zöld szaggatott vonal:




```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android=
  "http://schemas.android.com/apk/res/android"
    android:shape="line">
    <stroke
        android:width="2dp"
        android:color="#008000"
        android:dashWidth="3dp"
        android:dashGap="4dp"/>

    <size android:height="20dp" />
</shape>
```

# Állapotok leírása grafikai erőforrásokban

- Például gombfelirat színe:
  - res/color/button_text.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
        android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>
```

- Használat:

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:textColor="@color/button_text" />
```

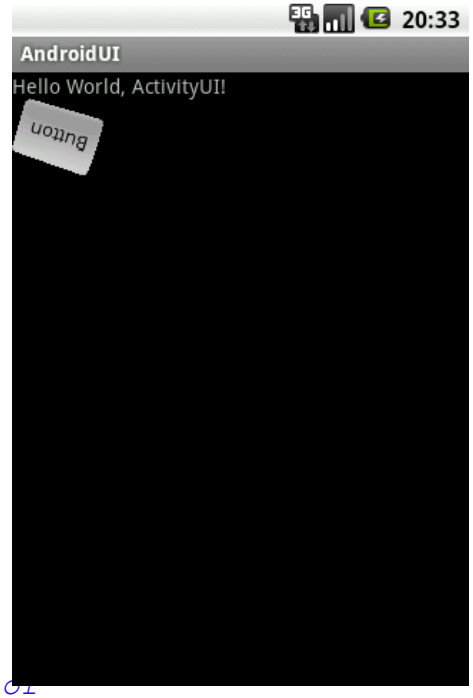
# ANIMÁCIÓK

# Animációk

- Animációk támogatása
  - XML erőforrás (*res/anim*)
  - Programkód
- Layout animáció
  - *Scale*
  - *Rotate*
  - *Translate*
  - *Alpha*
- Három fő típus:
  - Tween animáció
  - Frame animáció
  - Property animator

# Tween animáció erőforrás

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator=
          "@android:anim/accelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.0"
        android:fromYScale="0.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000" />

    <alpha android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="5000"/>

    <rotate
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromDegrees="0.0„
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="5000" />
</set>
```



# Tween animáció lejátszása

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);


  Button btn = (Button)findViewById(R.id.button1);
  btn.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
       Toast.makeText(ActivityUI.this,
         getResources().getString(R.string.toast_info),
           Toast.LENGTH_LONG).show();
    }
  });


  Animation showAnim = AnimationUtils.loadAnimation(this,R.anim.btnanim);
  btn.startAnimation(showAnim);
}
```

# ÉLŐ HÁTTÉRKÉP, WIDGET

# Android alkalmazás komponens modell kiegészítése

- Jól ismert Android alkalmazás komponens modell kiegészítése:
  - Minialkalmazások (widget)
  - Élő háttérkép (live wallapaper)
- Látványos és hasznos funkció a felhasználók kezében
- Újszerű lehetőségek a fejlesztők kezében
- Könnyű és gyors fejlesztés
- Meglévő alkalmazások kiegészítése

# Élő háttérkép

- Android 2.1-től elérhető
- Animált, interaktív háttér
- Hozzáférés a platform API-aihoz (2D és 3D rajzolás, GPS, gyorsulásmérő, hálózatkezelés, stb.)
- Például:
  - Animált tájkép, mely a földrajzi pozíció alapján kalkulálja a napnyugtát/napfelkeltét
  - Google térkép

# Élő háttérkép megvalósítása

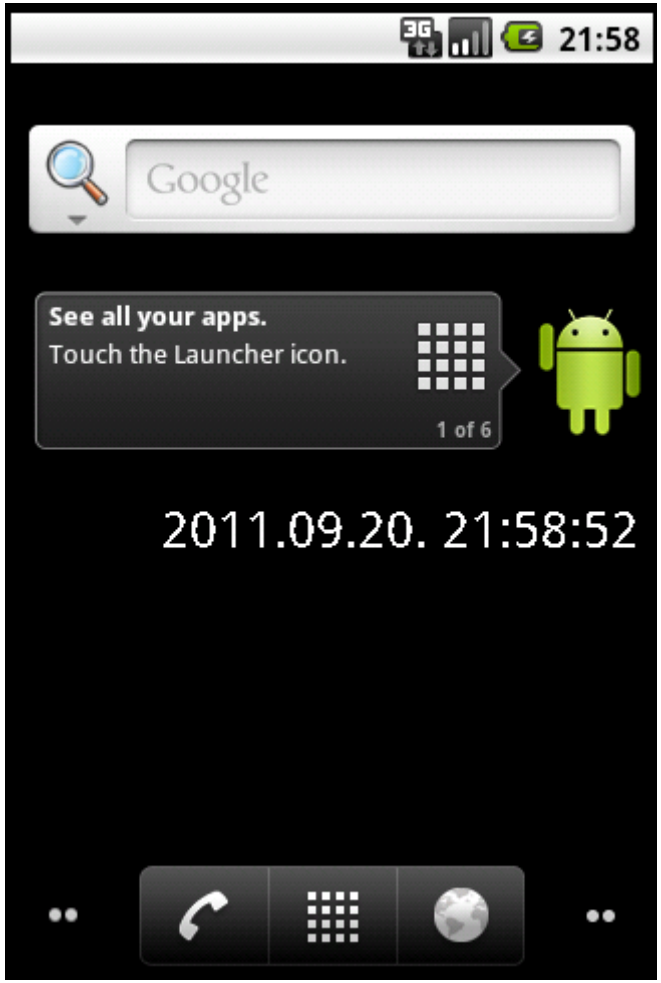
- Leginkább egy Service-hez hasonlítható, de van UI-a
- *onCreateEngine()*:
  - *WallpaperService.Engine* létrehozása
  - Az *Engine* felelős az életciklusért és a rajzolásért
- **Rajzolást nagyon fontos optimalizálni, a CPU és az akkumulátor használat miatt**
- Kezeljük megfelelően az életciklus eseményeket!
- Manifest állományba jelezni kell a Market felé az élő háttérkép képességet:
  - *<uses-feature android:name="android.software.live_wallpaper" />*

# Élő háttérkép eseménykezelés

- *onVisibilityChanged()*:
  - Ha háttérbe kerül, **függesszünk fel** minden tevékenységet!
- *onOffsetsChanged()*:
  - Home screen-ek közti váltás
- *onTouchEvent()*:
  - Érintés esemény kezelése
- *onCommand()*:
  - Más alkalmazás küldhet utasítást a háttérképnek, de jelenleg csak a beépített Home alkalmazás tud

# Élő háttérkép példa

- Készítsünk egy élő háttérképet, mely megjeleníti a dátumot és az időt az érintés helyén



# Élő háttérkép - manifest

```
<manifest xmlns:android=http://schemas.android.com/apk/res/android
package="hu.bute.daai.amorg.examples">

    <uses-sdk android:minSdkVersion="7" />
    <uses-feature android:name="android.software.live_wallpaper" />

    <application
        android:label="@string/wallpapers"
        android:icon="@drawable/ic_launcher_wallpaper">

        <service
            android:label="@string/my_wallpaper"
            android:name=".MyWallpaper"
            android:permission="android.permission.BIND_WALLPAPER">
            <intent-filter>
                <action android:name="android.service.wallpaper.WallpaperService" />
            </intent-filter>
            <meta-data android:name="android.service.wallpaper"
                android:resource="@xml/mywall" />
        </service>
    </application>
</manifest>
```

# Élő háttérkép – forrás 1/2

```
public class CubeWallpaper1 extends
WallpaperService {

    private final Handler mHandler =

      new Handler();


    @Override

    public void onCreate() {

        super.onCreate();

    }


    @Override

    public void onDestroy() {

        super.onDestroy();

    }


    @Override

    public Engine onCreateEngine() {

        return new CubeEngine();

    }
```

```
class CubeEngine extends Engine {
    private final Paint mPaint = new Paint();
    private float mTouchX = 0;
    private float mTouchY = 0;

    CubeEngine() {
        final Paint paint = mPaint;
        paint.setColor(0xffffffff);
        paint.setTextSize(25);
    }
```

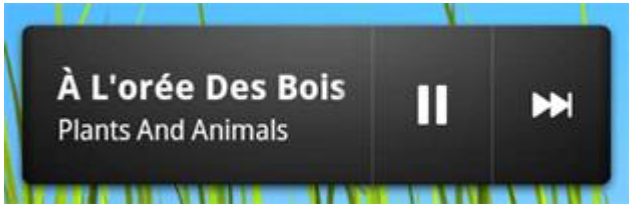
# Élő háttérkép – forrás 2/2

```
@Override
public void onCreate(SurfaceHolder
surfaceHolder) {
  super.onCreate(surfaceHolder);
  // Érintés eseményre kezelés jelzése
  setTouchEventsEnabled(true);
}

@Override
public void onTouchEvent(MotionEvent event)
{
  super.onTouchEvent(event);

  if (event.getAction()==
MotionEvent.ACTION_UP) {
    mTouchX=-1; mTouchY=-1;
  } else {
    mTouchX = event.getX();
    mTouchY = event.getY();
  }
  drawFrame();
}
```

```
void drawFrame() {
  final SurfaceHolder holder =
    getSurfaceHolder();
  Canvas c = null;
  try {
    c = holder.lockCanvas();
    if (c != null) {
      c.save();
      c.drawColor(0xff000000);
      c.drawText(
      new Date(System.currentTimeMillis()).
        toLocaleString(),
        mTouchX, mTouchY, mPaint);
      c.restore();
    }
  } finally {
  if (c != null)
    holder.unlockCanvasAndPost(c);
  }
}}} // osztály bezárása
```

# Widget - minialkalmazások

- Kis alkalmazások, melyek tipikusan a Home Screen-be ágyazódnak
- Periodikusan frissítik az állapotukat
- Például zene lejátszó

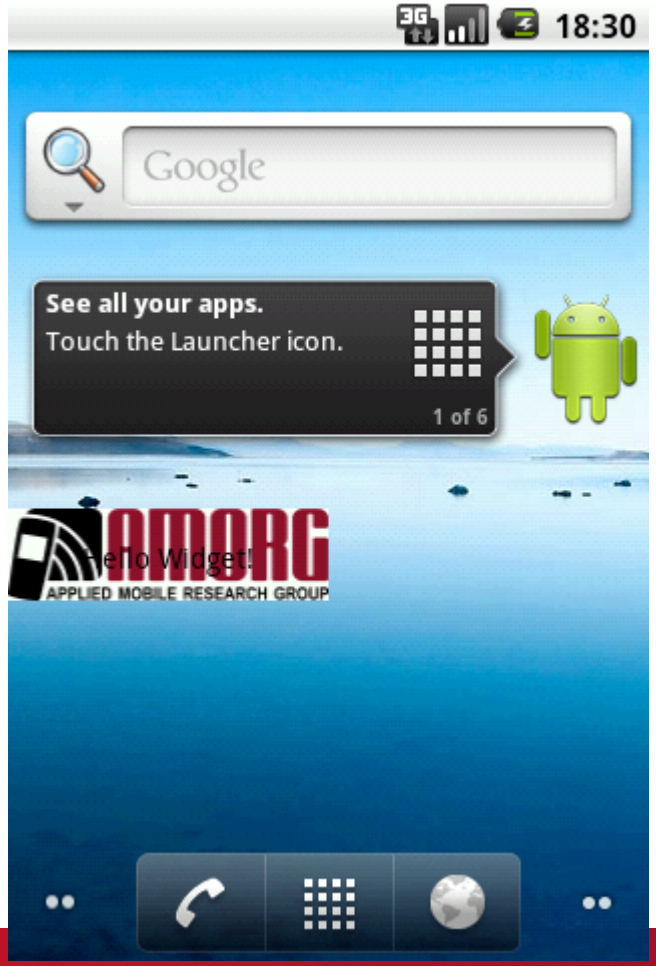


# Widget komponensek

- *AppWidgetProviderInfo*:
  - XML alapú
  - Metaadatok (layout, **frissítés gyakorisága**, AppWidgetProvider osztály)
- *AppWidgetProvider*:
  - Programozói felületet biztosít
  - Broadcast események: updated, enabled, disabled, deleted
- *View layout*:
  - XML-ben a widget felülete
- Konfigurációs Activity:
  - Opcionális
  - Widget hozzáadásakor indul, widgettel kapcsolatos beállítások kezelése

# Widget példa

- Háttérkép és szöveg megjelenítése



# Widget Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="hu.bute.daai.amorg.examples"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
    <!-- Broadcast Receiver fog frissíeni -->
        <receiver android:name=".HelloWidget" android:label="@string/app_name">
            <intent-filter>
              <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
           <meta-data android:name="android.appwidget.provider„
                android:resource="@xml/hello_widget_provider" />
        </receiver>
    </application>
</manifest>
```

# Widget AppWidgetProviderInfo XML

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android=
"http://schemas.android.com/apk/res/android"
android:minWidth="146dip"
android:minHeight="72dip"
android:updatePeriodMillis="500"
android:initialLayout="@layout/main"
/>
```

# Widget Layout

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

<TextView android:id="@+id/widget_textview"
    android:text="@string/hello_text"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_gravity="center_horizontal|center"
    android:layout_marginTop="5dip"
    android:padding="10dip"
    android:textColor="@android:color/black"/>
</LinearLayout>
```

# Widget AppWidgetProvider

```
public class HelloWidget extends AppWidgetProvider {
  @Override
  public void onUpdate(Context context,
    AppWidgetManager appWidgetManager,
    int[] appWidgetIds) {
    // frissítés
  }
}
```

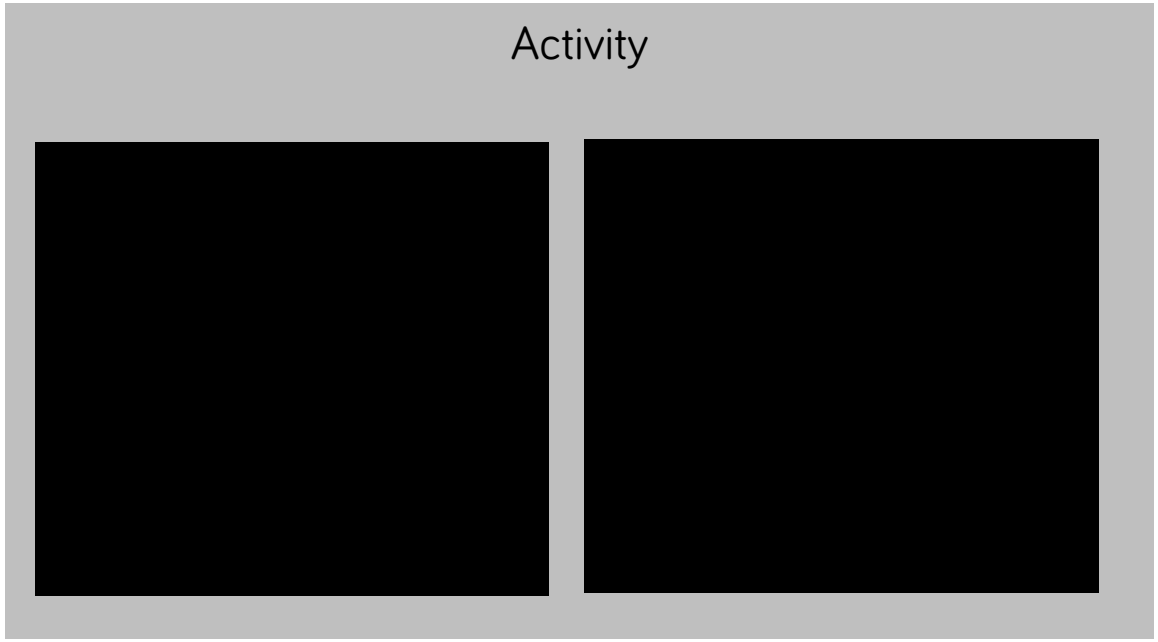
# FRAGMENT-EK

# Fragment API

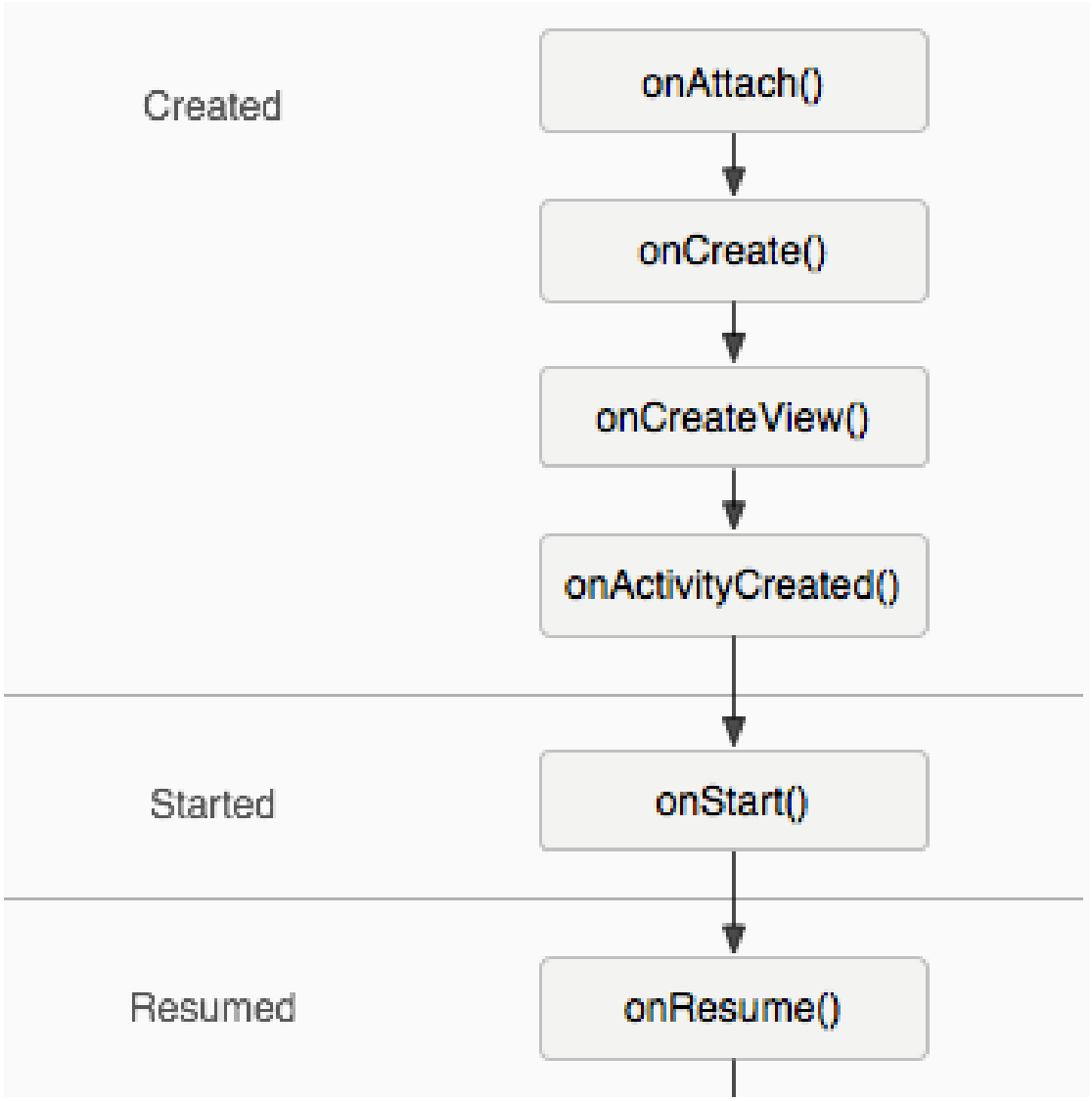
- Mik azok a Fragment-ek?
  - Elsősorban: A képernyő egy nagyobb részéért felelős objektumok
  - Továbbá: A háttérben munkát végző objektumok
- Miért kellenek nekünk?
  - Nagy képernyőméret = több funkció egy képernyőn = bonyolultabb Activity-k
  - Fragment-ekkel modulárisabb, rugalmasabb architektúra építhető

# Fragment és Activity

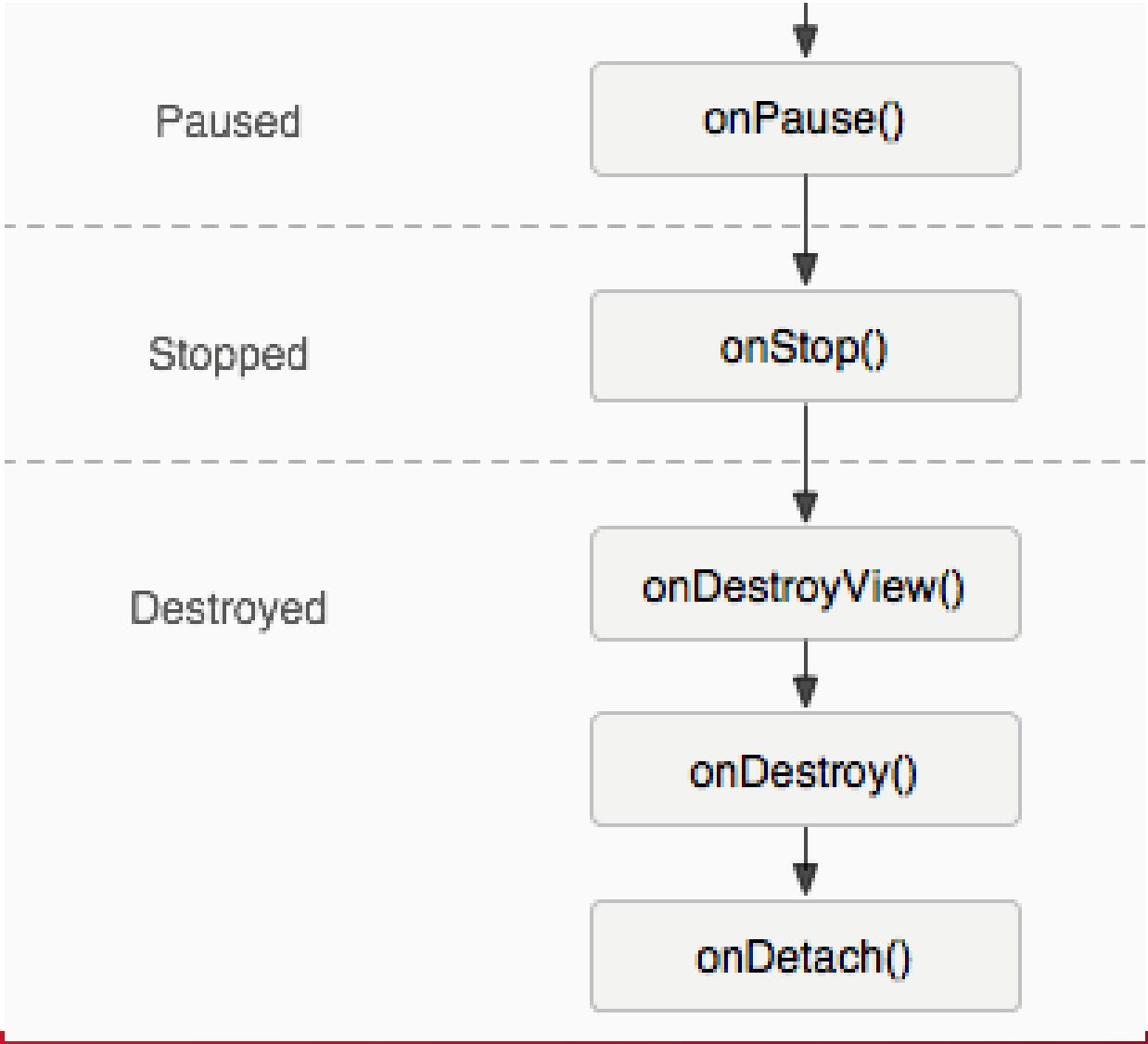
- Egy Fragment mindig egy Activity-hez csatoltan jelenik meg
- Az életciklusaik nagyrészt megegyeznek

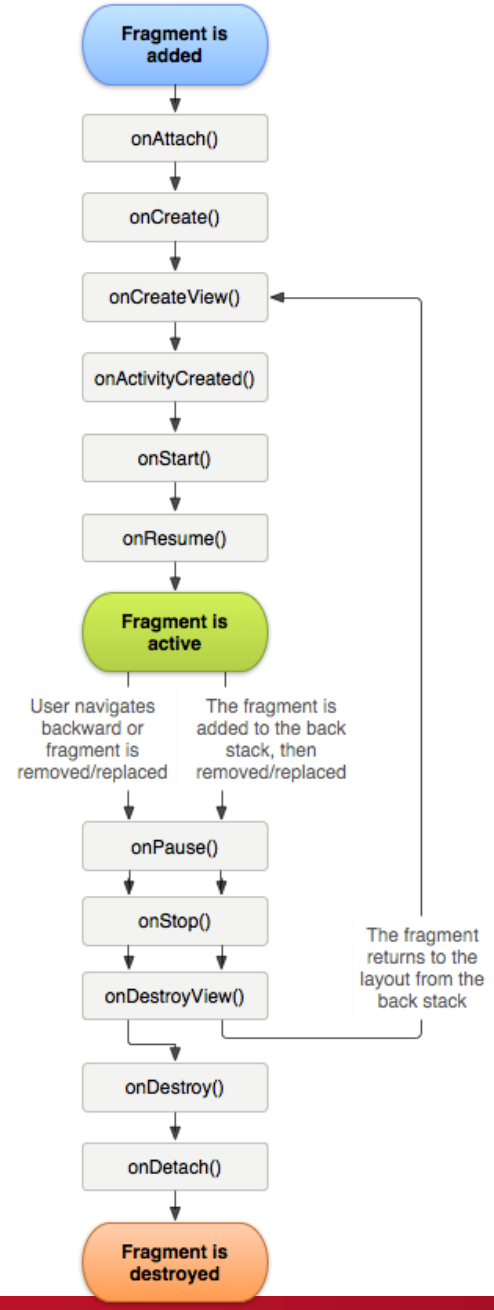


# Fragment életciklus I.

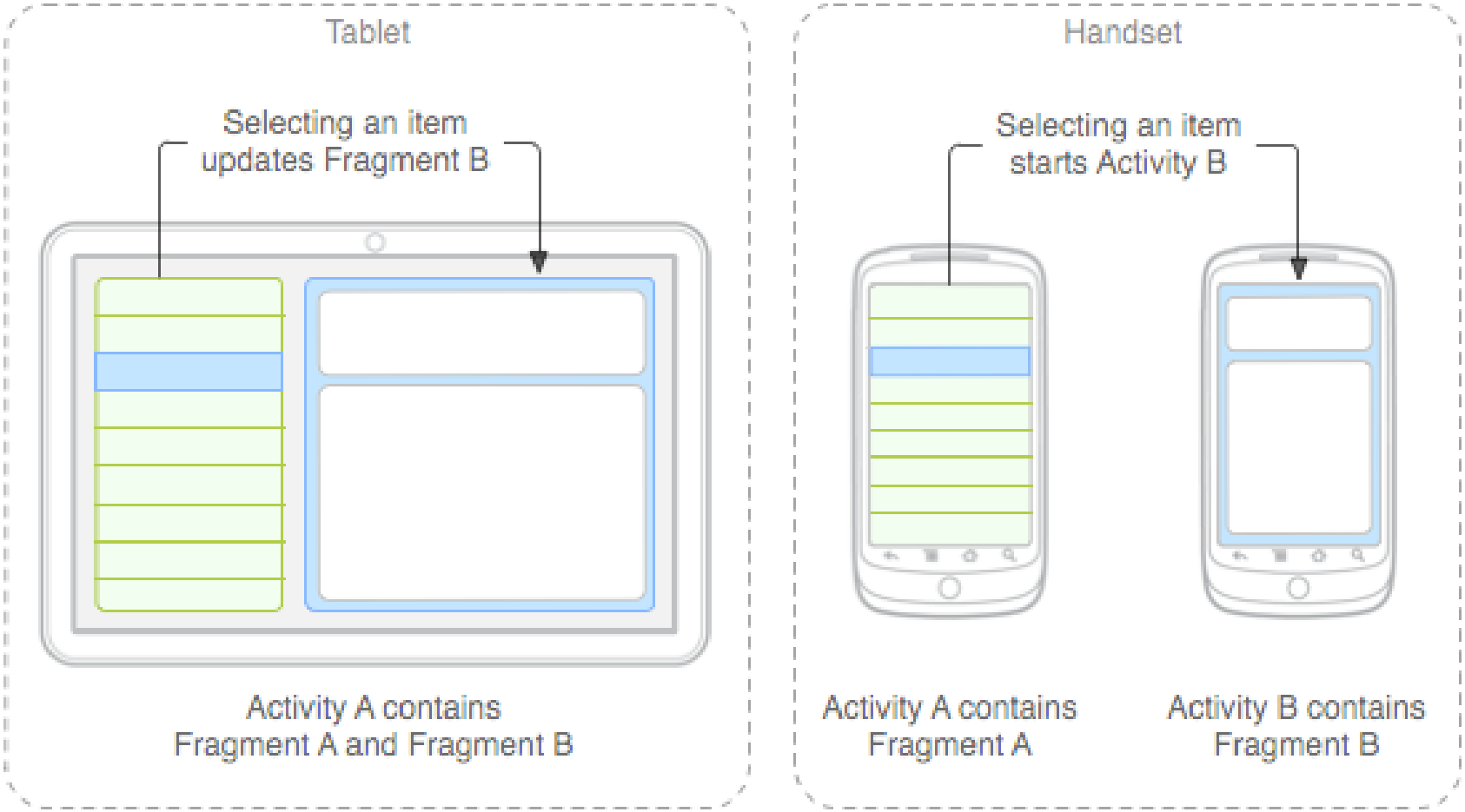


# Fragment életciklus II.




Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

# Eltérő képernyőméretek



# UI Fragment készítése...

- A megjelenítendő View-hierarchiát az .onCreateView() metódusban kell visszaadni

```
public class MyFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View root = View.inflate(getActivity(), R.layout.details, null);

        // Inicializalas

        return root;
    }
}
```

# … és csatolása

- Statikusan
  - Az Activity-hez tartozó layout-ban beégetjük a Fragment-et, nem módosítható később
  - <fragment …/> tag
- Dinamikusan
  - Az Activity futás közben tölti be a megfelelő Fragment-eket, adott ViewGroup-okba
  - Fragment-Tranzakciókkal módosítható

# Statikus csatolás példa

```
<fragment class="hu.bme.aut.fragment.MenuListFragment"
    android:tag="MenuListFragment"
    android:layout_width="0dip"
    android:layout_height="fill_parent"
    android:layout_weight="1"/>
```

# A FragmentManager

- A FragmentManager-el menedzselhetők a Fragment-ek
  - Activity: .getFragmentManager()
  - FragmentTransaction indítása
  - Aktív Fragment-ek közt keres
    - Tag alapján
    - ID alapján
  - **Fragment-stack-et menedzseli**

# FragmentTransaction osztály I.

- Ezen keresztül módosíthatók az aktív Fragment-ek
- A FragmentManager .beginTransaction() metódusával indítható
- Fontosabb műveletek:
  - .add(...) / .remove(...) / .replace(...)
    - Fragment példányok le- és felcsatolása az adott Activity-re
  - .commit()
    - Tranzakció végrehajtása

# FragmentTransaction osztály II.

- .show(...) / .hide(...)
  - Fragment példány elrejtése / újra megjelenítése
- .setTransition(...) / .setCustomAnimations(...)
  - A tranzakció végrehajtásakor lejátszandó animáció beállítása
- .addToBackStack(...)
  - Rákerüljön-e a FragmentTransaction backstack-re a tranzakció?
- .commit()
  - Tranzakció végrehajtása

# FragmentTransaction példa I.

- Fragment kicserélése:

```
FragmentManager fm = getSupportFragmentManager();

FragmentTransaction ft = fm.beginTransaction();
ft.replace(R.id.FragmentContent,
        DetailsFragment.newInstance(selIndex),
        DetailsFragment.TAG);
ft.commit();
```

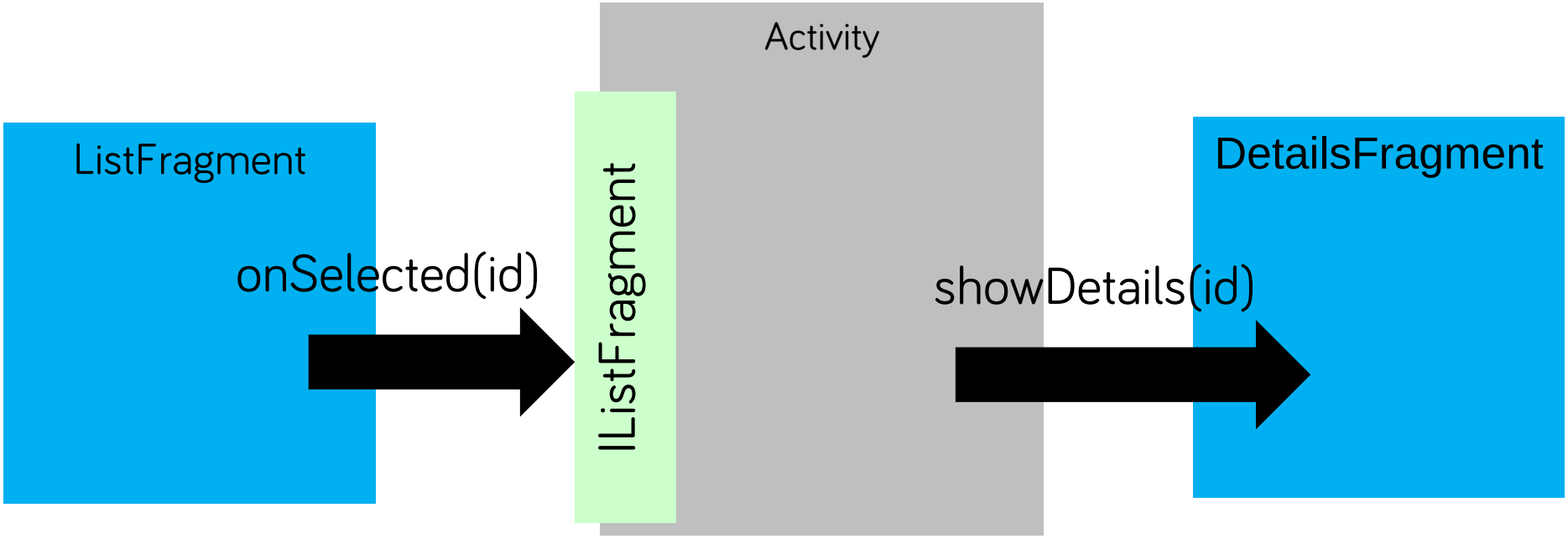
# FragmentTransaction példa II.

- Fragment hozzáadása, a tranzakciót a backstack-re téve:

```
FragmentManager fm = getSupportFragmentManager();
FragmentTransaction ft = fm.beginTransaction();
ft.setCustomAnimations(R.anim.slide_in_top,
        R.anim.slide_out_right);
ft.add(R.id.FragmentContent,
        DetailsFragment.newInstance(position),
        DetailsFragment.TAG);
ft.addToBackStack(null);
ft.commit();
```

# Fragment kommunikáció I.

- Egy Fragment-nek egységbezártnak kell lennie
  => közvetett kommunikáció
  - > Az Activity közvetít



# DialogFragment I.

- Egy Fragment dialógusként is megjelenhet
  - Dialógus egyedi layout-al
  - Az AlertDialog.Builder továbbra is használható
- Így egy dialógus is ugyanolyan életciklussal rendelkezik, mint egy Fragment
- A FragmentDialog-ok is rákerülhetnek a BackStack-re

# DialogFragment II.

- .onCreateDialog()
  - Ez a metódus is visszatérhet a megjelenítendő Dialog-al
- .onCreateView()
  - Ha nem használjuk az .onCreateDialog()-ot
  - Tetszőleges megjeleníthető tartalom
- Egy DialogFragment egyben Fragment is!
  - Ha kell, akár Activity-be ágyazottan is megjeleníthető

# Mi nem igaz a Fragment-ekre?

A. Saját életciklus függvényekkel rendelkeznek.

B. Dialógusként nem jeleníthetők meg.

C. Dinamikusan és statikusan is csatolhatók.

D. A tabletek felhasználói felületének kialakításakor különösen hasznosak.

# Összefoglalás

- Grafikai erőforrások
- Animációk
- Élő háttérkép, Widget
- Fragmentek
  - Használjuk a Fragment keretrendszert!
  - Rugalmasabb, könnyebben konfigurálható architektúrát kapunk, ha jól alkalmazzuk
  - A régi Dialog-ok helyett DialogFragment-et használjunk
  - Tervezzünk egyszerre mindenféle kijelzőméretű hardverre (lehetőleg)

# Kérdések