Grafikonok, Szenzorok, Külső osztálykönyvtárak, Multimédia alapok

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

hwsw

AutSoft

Tematika

1. Service komponens
2. ContentProvider, Komplex felhasználói felületek
3. Játékfejlesztés
4. Grafikonok, Szenzorok, Külső osztálykönyvtárak, Multimédia alapok
5. Multimédia, további kommunikációs megoldások
6. Biztonságos alkalmazások
7. Andorid TV és Wear fejlesztés
8. Android 9 újdonságok és további helyfüggő szolgáltatások
9. Tesztelési lehetőségek
10. Alkalmazás publikálás, karbantartás (CI/CD)

Tartalom

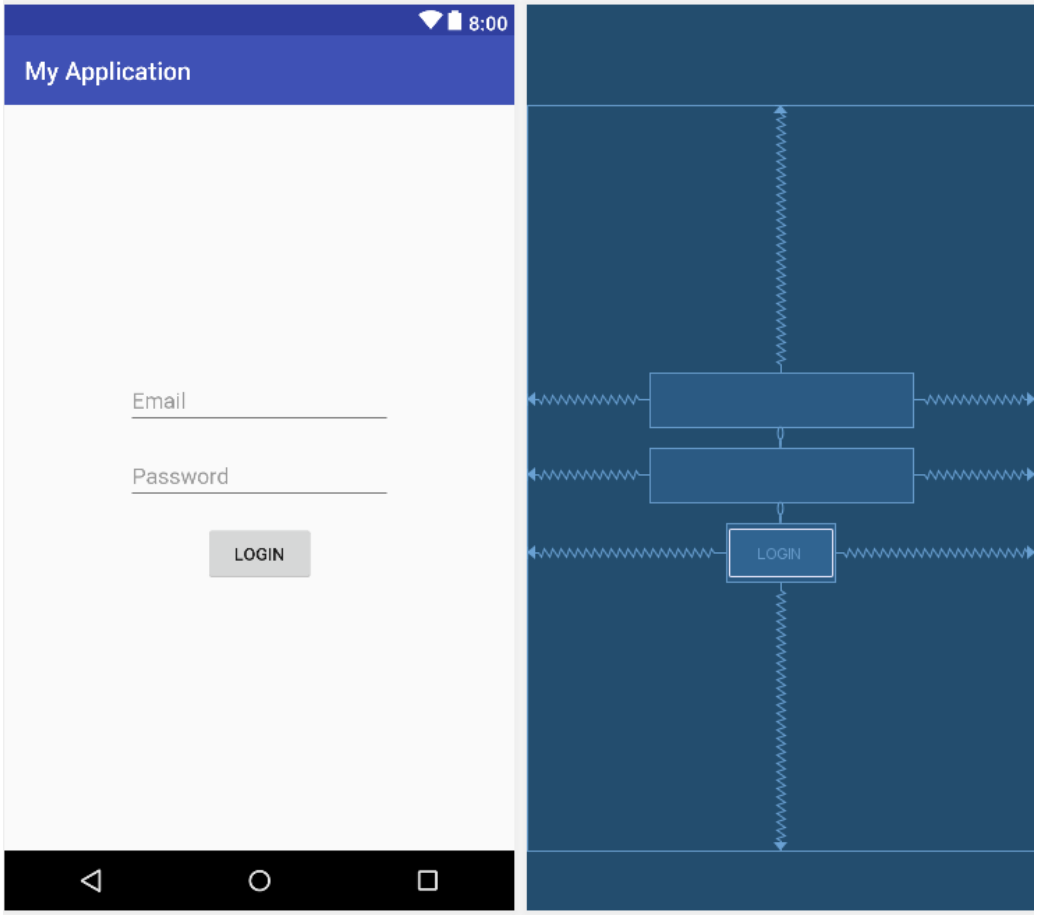
- ConstraintLayout - gyakorlás
- Drag&Drop általános kezelése
- Grafikonok rajzolása
- Gyakran használt külső UI library-k áttekintése
- Szenzorok

ConstraintLayout

Gyakorlás

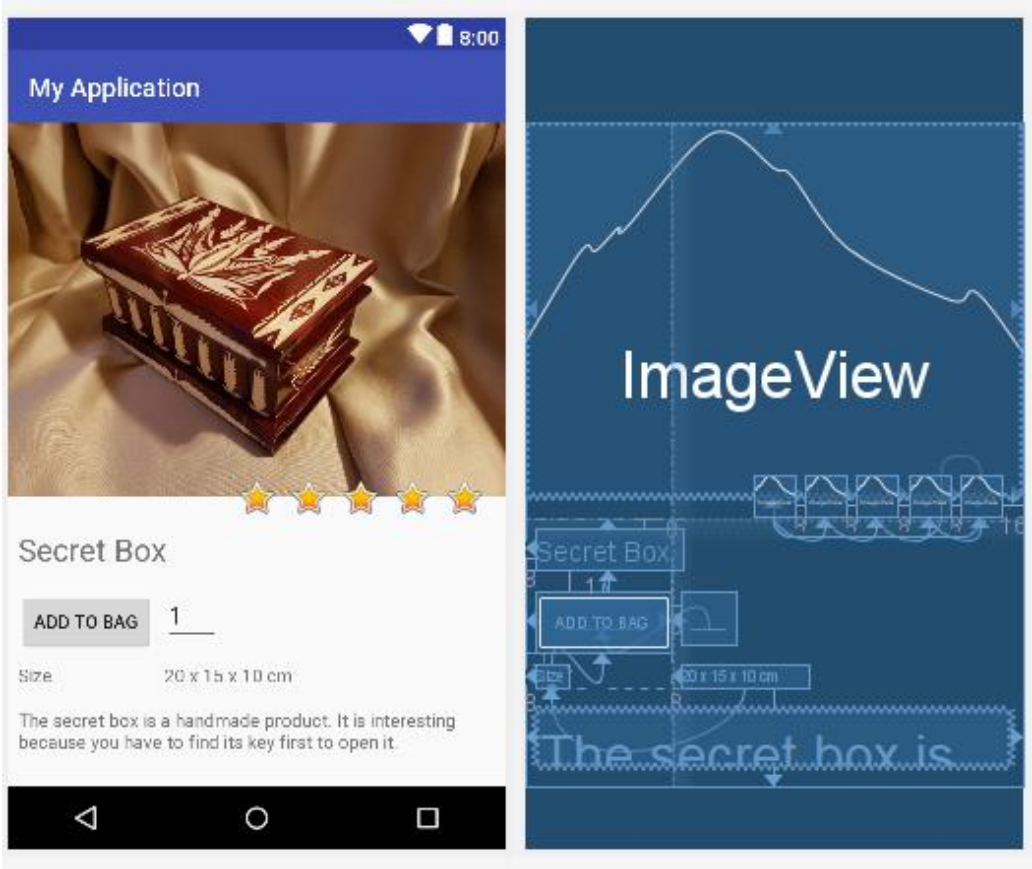
Gyakoroljunk! – 1. Feladat

Készítsük el a felületét egy login activity-nek ConstraintLayout-ot és Layout Editor-t használva!



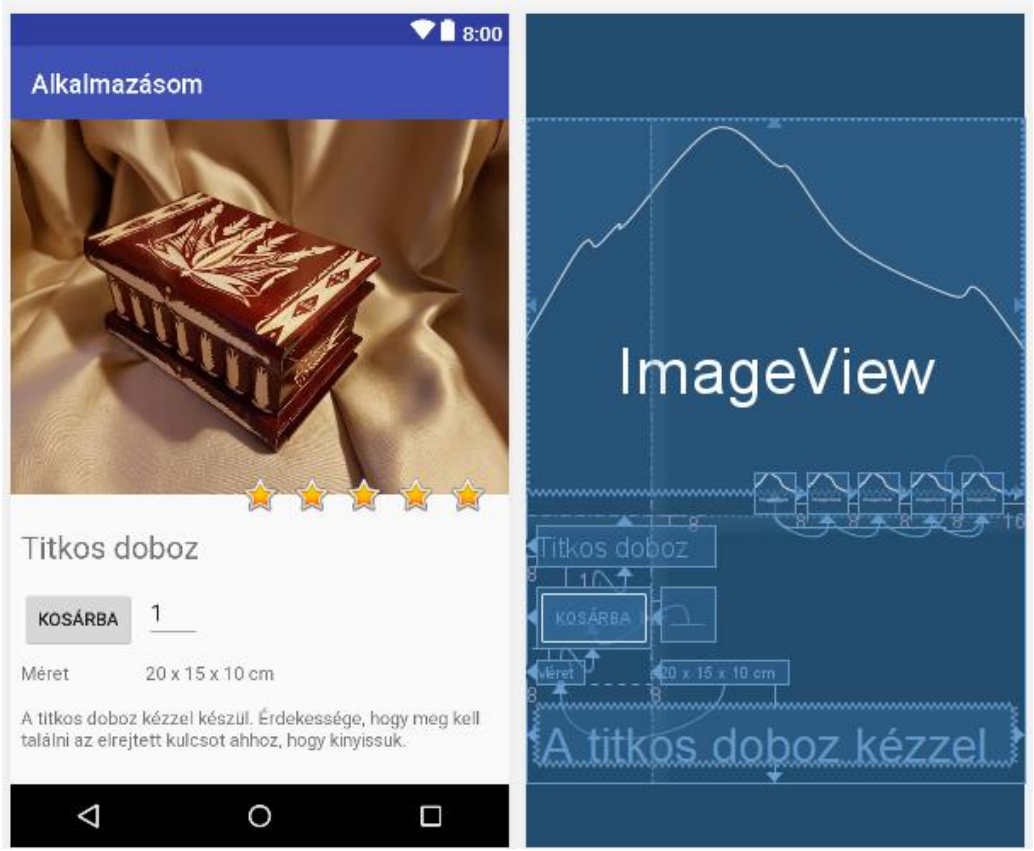
Gyakoroljunk! – 2. Feladat

Készítsük el a felületét egy webáruházban levő termék oldalának ConstraintLayout-ot és Layout Editor-t használva!



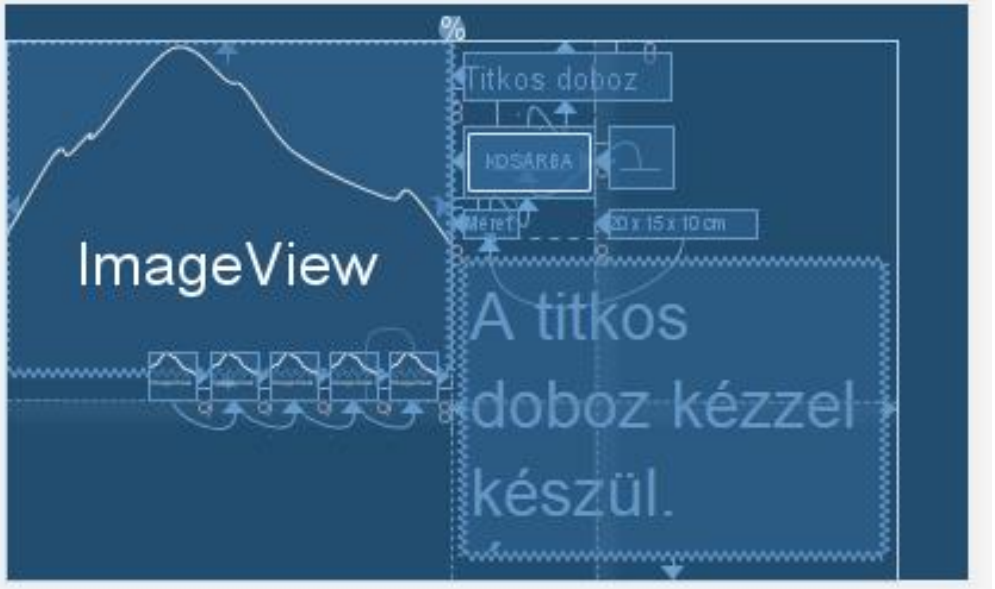
Gyakoroljunk! – 3. Feladat

Az előző feladatban elkészített felülethez adjunk magyar nyelvű fordítást is! Ellenőrizzük, hogy a magyar szövegekkel is megmarad az elrendezés!



Gyakoroljunk! – 4. Feladat

Az előző feladatban elkészített felülethez adjunk fekvő nézetet is!



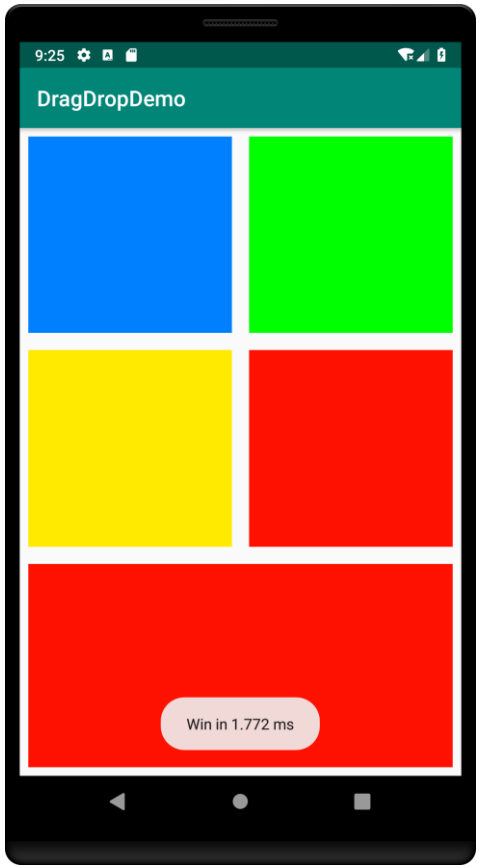
DRAG & DROP ÁLTALÁNOSAN

Drag & Drop támogatás

- Gyakorlatilag bármelyik komponens „drag”-elhető:
  - View. startDrag(...)
- Felületek/komponensek feliratkozhatnak „drag” esemény érzékelésre:
  - setOnDragListener(...)
- Többféle „drag” esemény:
  - ACTION_DRAG_STARTED
  - ACTION_DRAG_ENTERED
  - ACTION_DRAG_EXITED
  - ACTION_DROP

Gyakorljunk!

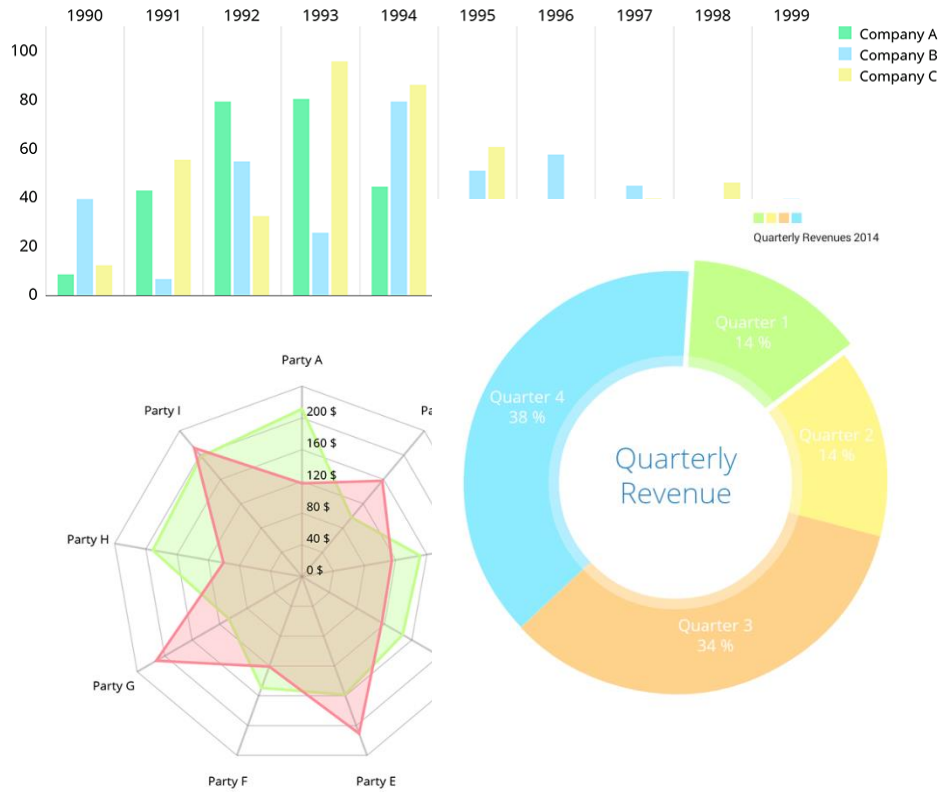
- Készítsünk egy színválasztó Drag&Drop játékot reakció idő mérésre!



GRAFIKONOK RAJZOLÁSA

MPAndroidChart

- Gazdag grafikon rajzoló osztálykönyvtár
- Számtalan grafikon típus
  - LineChart
  - BarChart
  - PieChart
  - CandleStickChart
  - BubbleChart
  - …
- https://github.com/PhilJay/MPAndroidChart



MPAndroidChart használat

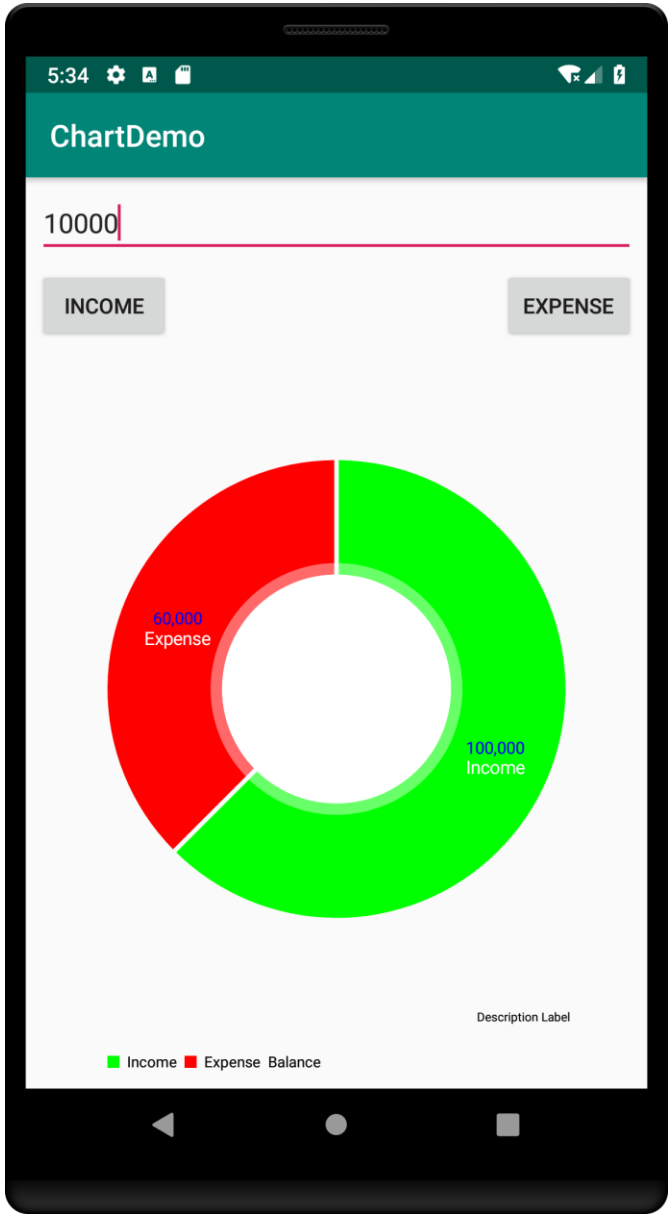
1. Grafikon elhelyezése layout file-ba

```
<com.github.mikephil.charting.charts.PieChart
    android:id="@+id/chartBalance"
    android:layout_width="0dp"
    android:layout_height="0dp"
    …/>
```

2. Grafikon beállítások: elforgatás, gesztusok kezelése, szöveg pozíciók, stb.
3. DataSet és entry-k (értékek) beállítása
4. Színsémák megadása értékekhez
5. Grafikon és DataSet összerendelése

Gyakoroljunk!

Készítsünk egy kiadás/bevétel kezelő alkalmazást és az állapotot jelenítsük meg egy *PieChart*-on



Szenzorok használata

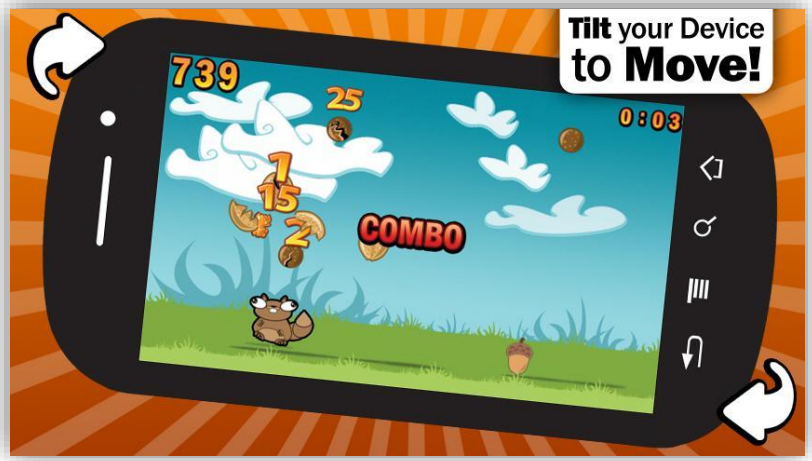
Kilépés a virtuális világból

Bevezetés

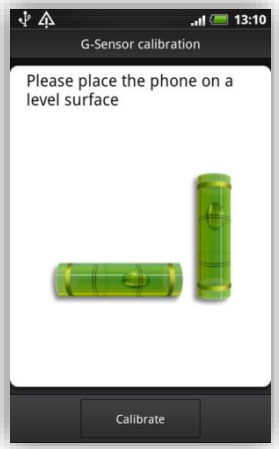
- A mai mobilok többre képesek a telefonálásnál és internet csatlakozásnál
- Rengeteg beépített szenzor
  - Gyorsulásmérő
  - Iránytű
  - Fényerősség érzékelő
  - Hőmérő
  - Stb…

Bevezetés

- Új lehetőségek az interakció megvalósítására
  - > Kiterjesztett valóság
  - > Mozgás alapú vezérlés
  - > Stb...



Támogatott szenzorok



- Az Android absztrakt szenzor típusokat támogat
  - **Sensor.TYPE_ACCELEROMETER**: háromtengelyes gyorsulásmérő, m/s^2-ben adja vissza a pillanatnyi értékeket
  - **Sensor.TYPE_GYROSCOPE**: elforgatás mértékét adja meg fokban, mindhárom tengelyre
  - **Sensor.TYPE_LIGHT**: ambiens megvilágítást méri, egyetlen visszaadott értékének mértékegysége *lux*. Ezt használja az op.rendszer a képernyő fényerősségének automatikus beállításához

Támogatott szenzorok

- **Sensor.TYPE_MAGNETIC_FIELD**: Mágneses erősség mérése három tengely mentén, microtesla egységekben. Iránytű alkalmazáshoz elengedhetetlen
- **Sensor.TYPE_ORIENTATION**: elforgatás szenzor. Közvetlenül nem használjuk, a SensorManager.getOrientation() adja az orientációt
- **Sensor.TYPE_PROXIMITY**: Visszaadja a telefon és a cél tárgy közti távolságot centiméterben. A telefon felvételekor (fülhöz emelés) az Android kikapcsolja a képernyőt, ezen szenzor segítségével

Támogatott szenzorok

- További egzotikus és származtatott szenzorok is támogatottak
  - Hőmérséklet, relatív páratartalom, légköri nyomás, elforgatás vektor, lineáris gyorsulás
- Speciális alkalmazás igények esetén használhatjuk őket
- Lekérdezhető, hogy milyen szenzorok elérhetők a telefonon
  - Egy absztrakt típusból akár több is jelen lehet ugyanazon a készüléken

Szenzorok elérése

- Rendszerszolgáltatás biztosítja a szenzorokkal történő kommunikációt, értékek lekérdezését
  - **`SensorManager`**
- SensorManager elérése

```
val sensorManager =
  getSystemService(SENSOR_SERVICE) as SensorManager
```

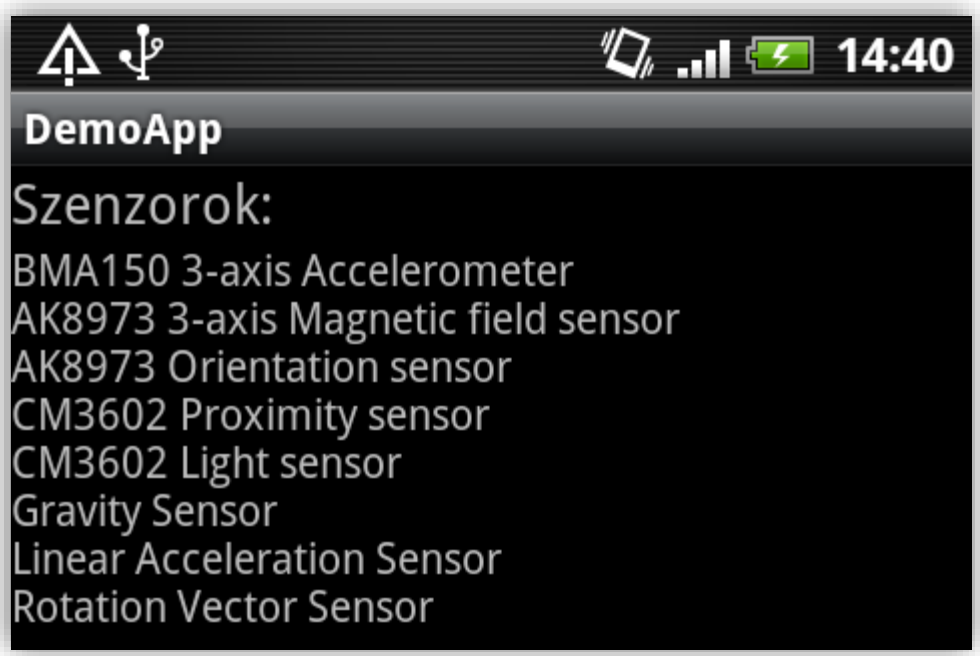
- Szenzor elkérése

```
val sensor = sensorManager.getDefaultSensor(
  Sensor.TYPE_MAGNETIC_FIELD)
sensorManager.registerListener(this, sensor,
        SensorManager.SENSOR_DELAY_NORMAL)
```

Összes szenzor listázása

- Lekérhető a készülék összes szenzora

```
sensorManager.getSensorList(Sensor.TYPE_ALL).forEach {
    tvStatus.append("${it.name}\n")
}
```



Szenzorok használata

- Eseménykezelt módon, **`SensorEventListener`** megvalósításával
  - **onSensorChanged(SensorEvent)**: A szenzor által mért érték változásakor hívódik (új mérés). SensorEvent-ből kinyerhető értékek:
    - Szenzor, amelyik triggerelte
    - Pillanatnyi mérési pontosság (*alacsony*, *közepes*, *magas*, *nem megbízható* – kalibráció szükséges)
    - Mért értékek tömbje (FloatArray). A szenzor típusa határozza meg hogy hogyan kell értelmezni a tartalmát
    - A mérés nanosec pontosságú időbélyege

Szenzorok használata

- Az eseménykezelő beállításakor megadhatjuk, hogy milyen gyakran szeretnénk mérni a szenzor értéket

SensorManager.DELAY_(FASTEST|GAME|NORMAL|UI)

- Erőforrás igényes feladat a szenzor folyamatos lekérdezése, válasszuk a célunkhoz megfelelő legalacsonyabbat
- És állítsuk le a frissítést, ha nem szükséges tovább futnia (onPause-ban)

Példa

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private lateinit var sensorManager: SensorManager

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager

        …
    }

    override fun onStop() {
        super.onStop()
        sensorManager.unregisterListener(this);
    }

    private fun listSensors() {
        sensorManager.getSensorList(Sensor.TYPE_ALL).forEach {
            tvStatus.append("${it.name}\n")
        }
    }

    private fun startSensor() {
        val sensor = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
        sensorManager.registerListener(this, sensor,
                SensorManager.SENSOR_DELAY_NORMAL)
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
    }

    override fun onSensorChanged(event: SensorEvent) {
        tvStatus.setText("Magneto: ${event.values[0]}")
    }
}
```

Kilépés a virtuális térből

- Mostanában az a trendi, ha az alkalmazás ki tud lépni a virtuális világból
- Nem (csak) úgy működik, hogy nézzük a kirajzolt pixeleket és nyomkodjuk a képernyőt
- Szenzorok használatával képes kapcsolatot teremteni a külvilággal
- Iránytű, gyorsulásmérő és elforgatás szenzor kombinációival érhető el a legtöbb

Kilépés a virtuális térből

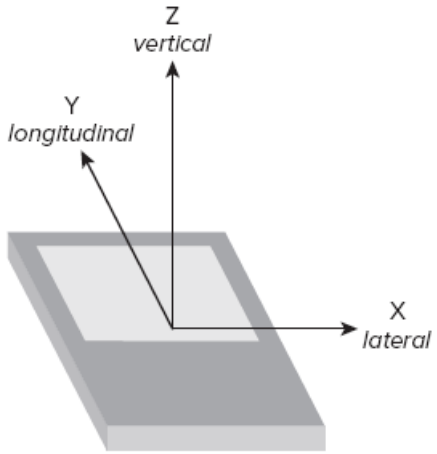
- Ezek segítségével
  - Tudjuk, hogy mi az orientáció
  - Mennyire van elforgatva a készülék
  - Melyik égtáj felé néz a felhasználó
  - Merre mozog a készülék
- További érzékelőkkel kiegészíthető
  - Hol van (GPS)
  - Mit lát a kamera
  - Stb…

Kilépés a virtuális térből

- Rengeteg lehetőség az innovációra, a szenzorokat ügyesen használó alkalmazások általában átütő sikert érnek el, például:
  - Iránytű, gyorsulásmérő, GPS és kamera használatával **kiterjesztett valóság** készíthető
  - Gyorsulásmérő segítségével érzékelhetjük az **ütközéseket** – baleset esetén jelezhetünk
  - Mozgás mint input – **Mobil Wii**

Gyorsulásmérő használata

- Három tengelyen méri a gyorsulást (nem sebességet!)
- A telefon háton fekvő helyzetében
  - X tengely: jobbra-balra
  - Y tengely: előre-hátra
  - Z tengely: fel-le



- Fontos tudni: nyugvó állapotban a Z tengely a gravitációs gyorsulást méri (9.81 m/s^2)

Gyorsulásmérő használata

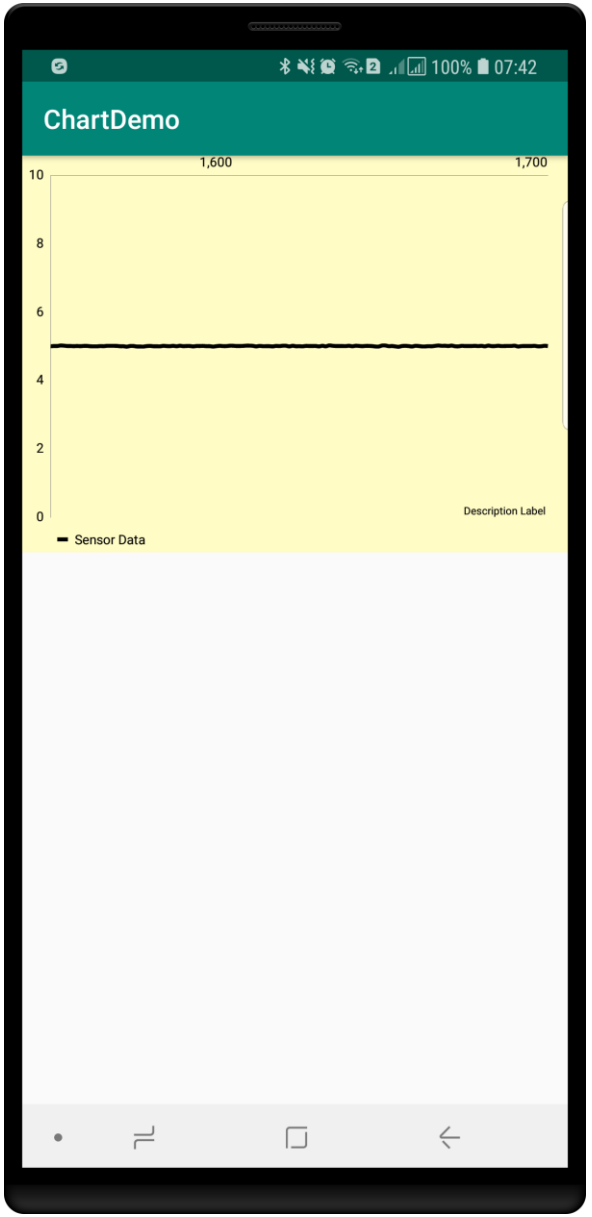
- Mérési adatok értelmezése (G-erő számítása = három gyorsulási érték négyzetösszegének gyöke mínusz gravitáció)

```
override fun onSensorChanged(event: SensorEvent) {

    val accX = event.values[0].toDouble()
    val accY = event.values[1].toDouble()
    val accZ = event.values[2].toDouble()

    var origin = Math.sqrt(
            Math.pow(accX, 2.0) +
            Math.pow(accY, 2.0) +
            Math.pow(accZ, 2.0)
    )
    origin = Math.abs(origin - SensorManager.STANDARD_GRAVITY)
}
```

Szenzorok használata

- Absztrakt szenzor típusok
- Mérési eredmények eseménykezelt módon
- Legritkább szükséges frissítést használjuk
- Állítsuk le onPause()-ban
- Nem a szenzor érték kinyerése a nehéz, hanem az értelmes felhasználása
- http://developer.android.com/guide/topics/sensors/sensors_motion.html

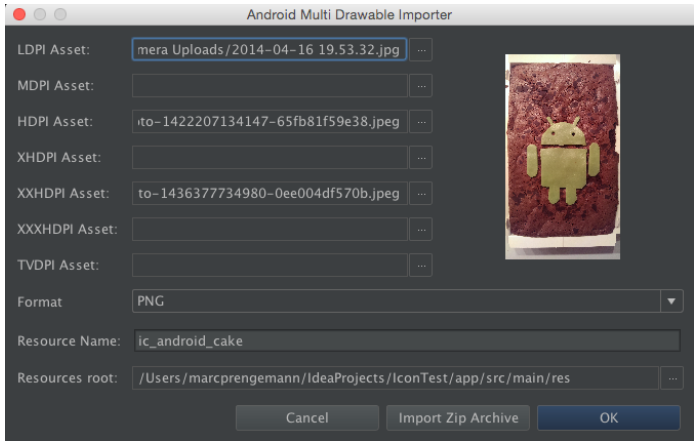
-
-


100% 07:42
ChartDemo
1,600
1,700
10
8
6
4
2
0
Description Label
Sensor Data

NÉPSZERŰ UI LIBRARY-K ÉS PLUGINEK

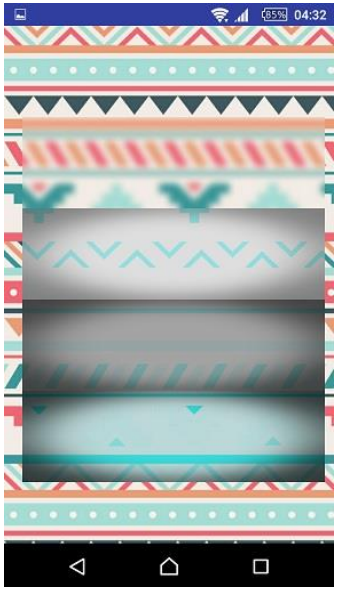
Képek kezelése

- Android Drawable Importer Plugin
  - https://plugins.jetbrains.com/plugin/7658-android-drawable-importer
- Ikonok gyors betöltése
  - Gazdag ikon készlet
  - Különböző méretek és színek
- Képek betöltése több méretben
  - Automatikus átméretezés
- Egyszerű és gyors használat



Blur effekt

- https://github.com/mirrajabi/view-effects

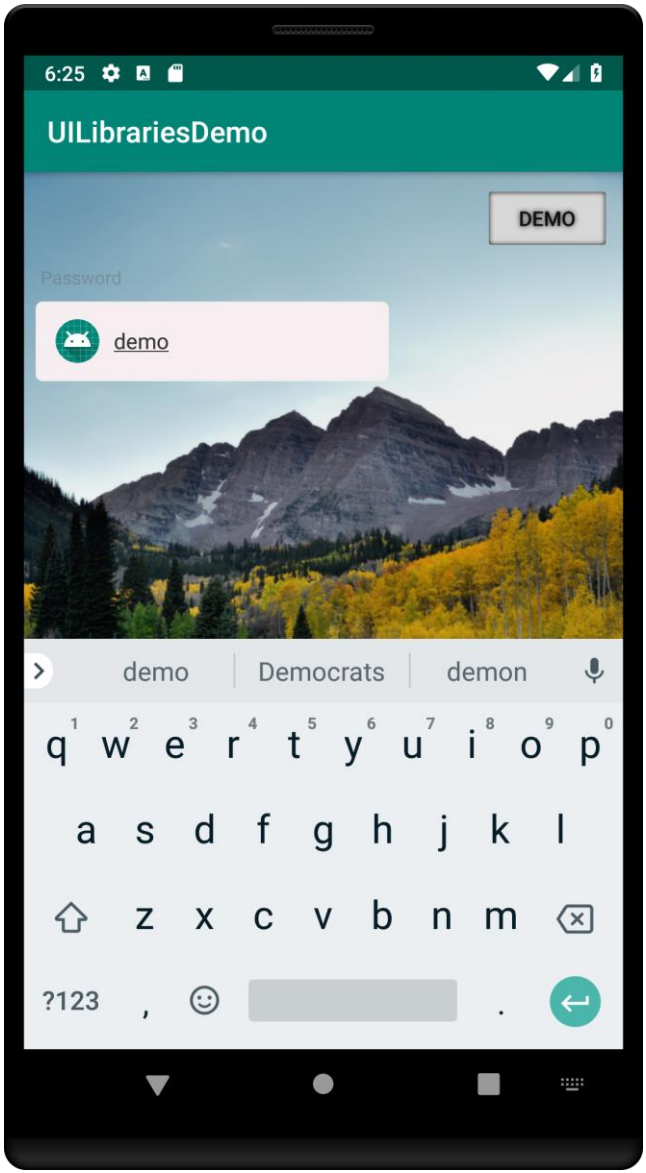



```
ViewFilter.getInstance(this)
    //Use blur effect or implement custom IRenderer
    .setRenderer(BlurRenderer(10))
    .applyFilterOnView(btnDemo,rootView)
```

MaterialTextField

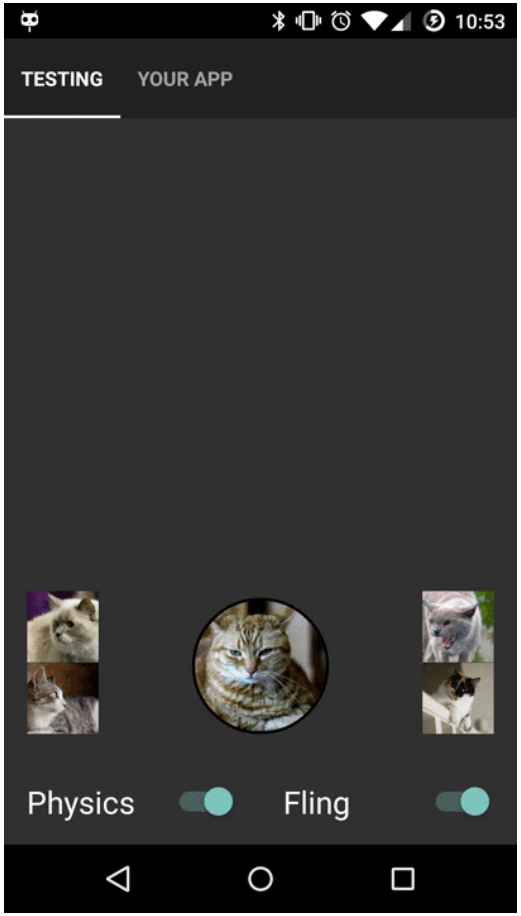
- Látványos beviteli mező ikonnal
- https://github.com/florent37/MaterialTextField

```
<com.github.florent37.materialtextfield.MaterialTextField
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:mtf_animationDuration="500"
    app:mtf_cardCollapsedHeight="4dp"
    app:mtf_image="@mipmap/ic_launcher"
    app:mtf_labelColor="#666"
    app:mtf_openKeyboardOnFocus="true">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:ems="10"
        android:textColor="#333"
        android:textSize="15sp" />
</com.github.florent37.materialtextfield.MaterialTextField>
```



PhysicsLayout

- JBox2D motor által hajtott
- Nem játékra lett tervezve (arra libGdx pl)
- https://github.com/Jawnnypoo/PhysicsLayout



PhysicsLayout példa

```
<?xml version="1.0" encoding="utf-8"?>
<com.jawnnypoo.physicslayout.PhysicsLinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/physics_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:physics="true"
    app:gravityX="0.0"
    app:gravityY="9.8"
    app:bounds="true"
    app:boundsSize="50dp">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"/>

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world, I have physics!"/>

</com.jawnnypoo.physicslayout.PhysicsLinearLayout>
```

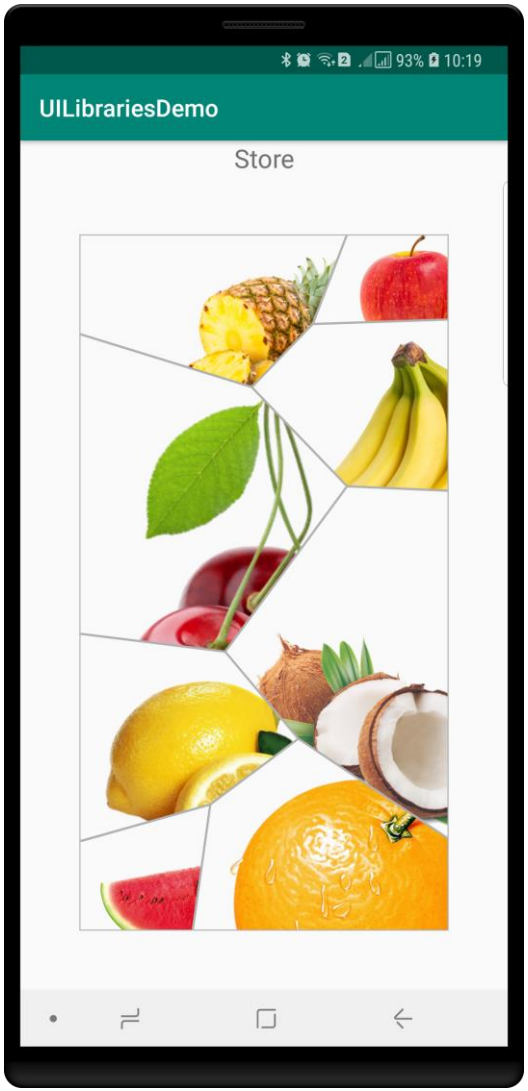
VoronoiView / Vorlay

- Márvány szerű elrendezés
- https://github.com/Quatja/Vorolay

VoronoiView példa

```
<quatja.com.vorolay.VoronoiView
    android:id="@+id/voronoiView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="50dp"
    custom:border_color="#b1b1b1"
    custom:border_round="false"
    custom:border_width="5"
    custom:generation_type="random"/>
```

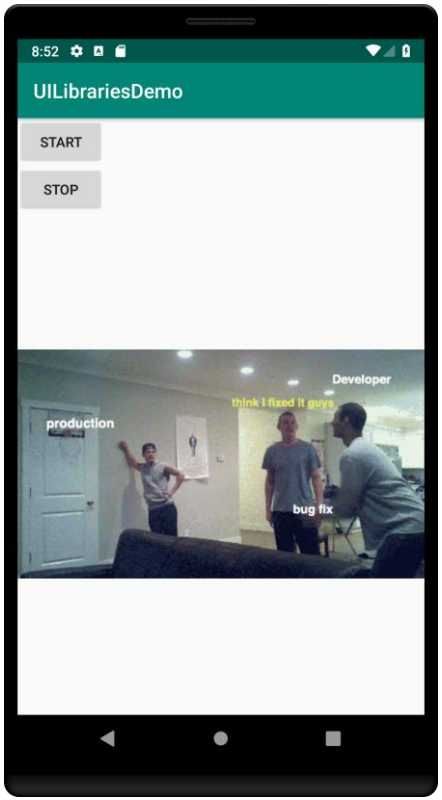
```
class VoronoiActivityActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_voronoi_activity)

        addFruit(R.drawable.ananas)
        addFruit(R.drawable.apple)
        addFruit(R.drawable.banana)
        addFruit(R.drawable.cherry)
        addFruit(R.drawable.coconut)
        addFruit(R.drawable.lemon)
        addFruit(R.drawable.melone)
        addFruit(R.drawable.orange)

    }
    private fun addFruit(fruitDrawableId: Int) {
        val imageView = ImageView(this);
        imageView.setImageResource(fruitDrawableId);
        voronoiView.addView(imageView)
    }
}
```



GifImageView

- GIF-ek lejátszása egyszerűen
- https://github.com/felipecsl/GifImageView

```
<com.felipecsl.gifimageview.library.GifImageView
    android:id="@+id/gifImageView"
    android:layout_gravity="center"
    android:scaleType="fitCenter"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>


btnStart.setOnClickListener {
    gifImageView.setBytes(IOUtils.toByteArray(
      resources.openRawResource(R.raw.developer)))
    gifImageView.startAnimation()
}
```

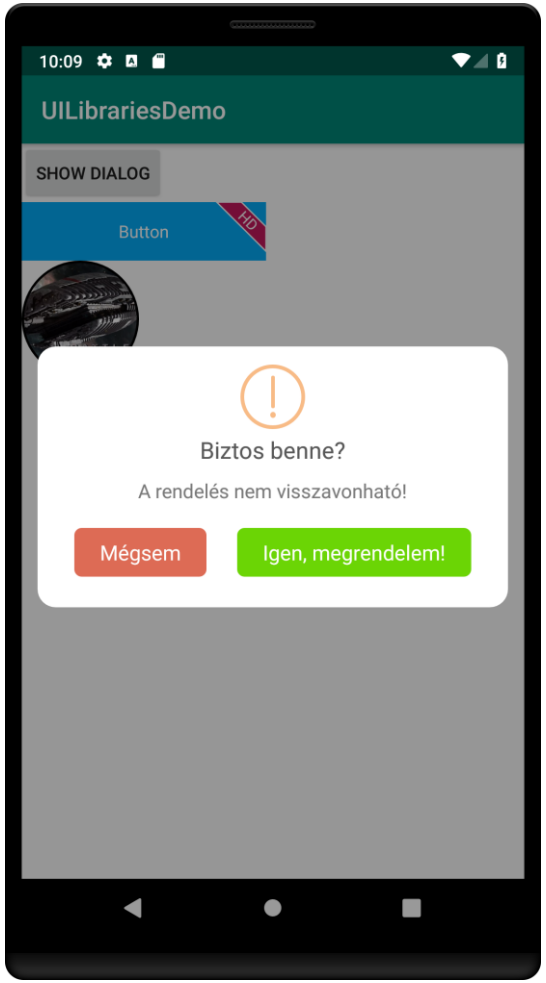


- IOUtils – Apache Common IO

```
> implementation 'commons-io:commons-io:+'
```

SweetAlertDialog

- Látványos, több funkciós dialógusok megjelenítése egyszerűen
  - > https://jitpack.io/p/Leogiroux/sweet-alert-dialog

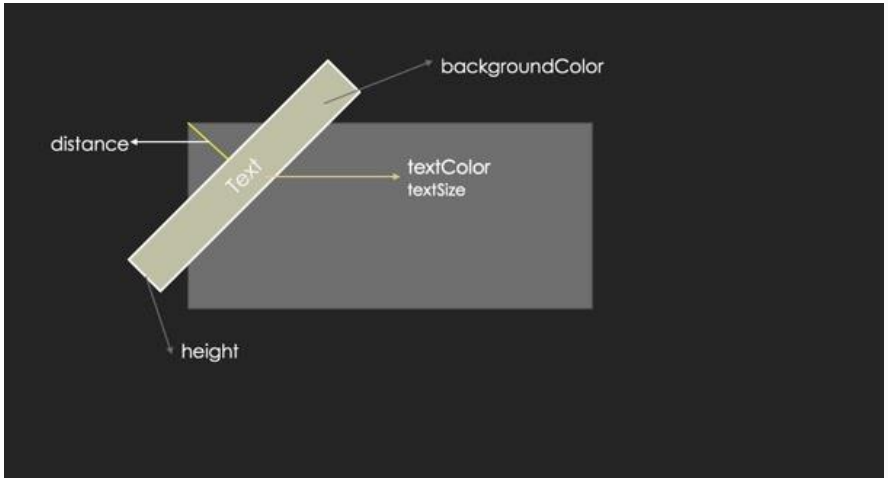


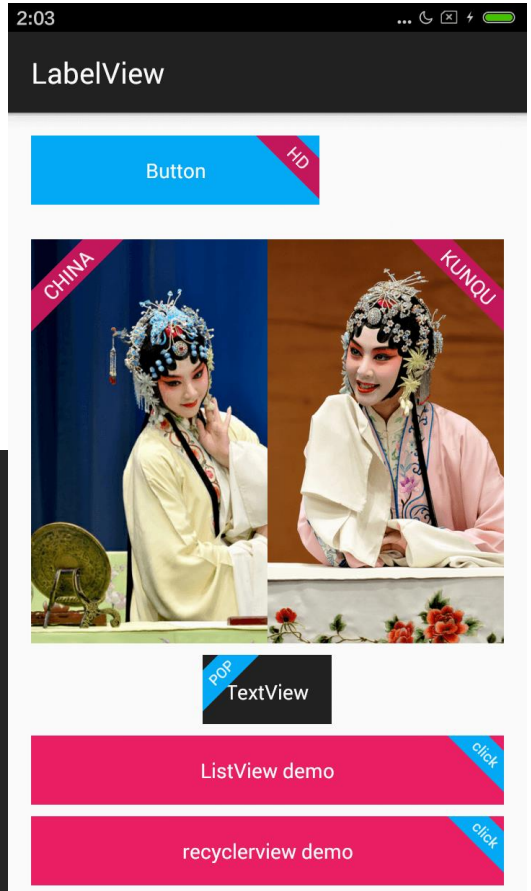

```
SweetAlertDialog(this, SweetAlertDialog.WARNING_TYPE)
        .setTitleText("Biztos benne?")
        .setContentText("A rendelés nem visszavonható!")
        .setConfirmText("Igen, megrendelem!")
        .setConfirmClickListener{
                it.dismissWithAnimation()
        }
        .setCancelButton("Mégsem") {
          sDialog -> sDialog.dismissWithAnimation() }
        .show()
```

LabelView

- Címkék elhelyezése gombon, képen, szövegen

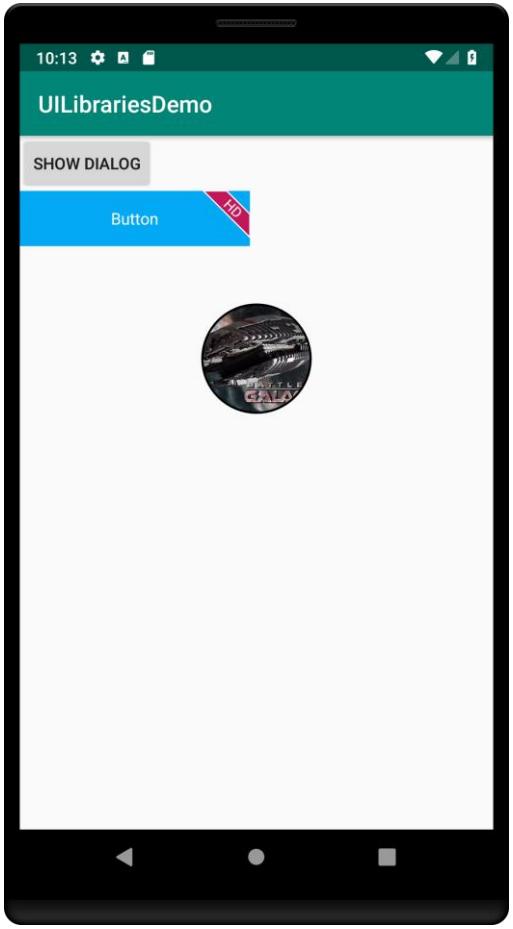
```
<com.lid.lib.LabelButtonView
    android:id="@+id/labelbutton"
    android:layout_width="200dp"
    android:layout_height="48dp"
    android:background="#03a9f4"
    android:gravity="center"
    android:text="Button"
    android:textColor="#ffffff"
    app:label_backgroundColor="#C2185B"
    app:label_distance="20dp"
    app:label_height="20dp"
    app:label_orientation="RIGHT_TOP"
    app:label_text="HD"
    app:label_textSize="12sp" />
```





CircleImageView

- Kör formában kivágott képek (pl. profil képek)
  - https://github.com/hdodenhof/CircleImageView




```
<de.hdodenhof.circleimageview.CircleImageView
    android:id="@+id/circleImage"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:src="@drawable/galactica"
    app:civ_border_width="2dp"
    app:civ_border_color="#FF000000"/>
```

További UI gyűjtemények

- https://github.com/wasabeef/awesome-android-ui
- https://android-arsenal.com/
- https://github.com/JStumpp/awesome-android
- https://www.uplabs.com/collections/library-b9e1d971-50e0-4600-ab9b-5f6b23fe3c0b
- https://medium.freecodecamp.org/25-new-android-libraries-which-you-definitely-want-to-try-at-the-beginning-of-2017-45878d5408c0

Összefoglalás

- ConstraintLayout - gyakorlás
- Drag&Drop általános kezelése
- Grafikonok rajzolása
- Gyakran használt külső UI library-k áttekintése
- Szenzorok

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft