Android App forráskód

Eddigi előadások

- https://dl.dropboxusercontent.com/u/93156461/!deleteme/!Tanfolyam/10.zip
- https://dl.dropboxusercontent.com/u/93156461/!deleteme/!Tanfolyam/11.zip
- https://dl.dropboxusercontent.com/u/93156461/!deleteme/!Tanfolyam/12.zip
- https://dl.dropboxusercontent.com/u/93156461/!deleteme/!Tanfolyam/13.zip
- https://dl.dropboxusercontent.com/u/93156461/!deleteme/!Tanfolyam/14.zip

Android telefonunk felismerése

- Az Android telefonunkat fel kell ismertetnünk az Android Studio számára, hogy android USB kábelen keresztül rárakhassuk és kipróbálhassuk az Appunkat a telefonunkon
- Ha automatikusan nem ismeri fel a telefonunkat (ha a zöld háromszögre (Runra) megyünk és nem látjuk a listában), akkor próbáljuk meg letölteni az Android telefonunk gyártójának (márkánknak) az oldaláról a megfelelő programdrivert (OEM driver)
- http://developer.android.com/tools/extras/oem-usb.html#Drivers

Android telefonunk felismerése

- Természetesen be kell kapcsolnunk a telefonunkon az USB debugging (USB hibakeresés) módot (Beállítások->A telefonról->Kernel (Build) verziószámra 7szer klikkeljünk, majd ezek után keressük meg a Fejlesztői lehetőségek menüpontot, ott kapcsoljuk be az USB hibakeresést)
- Ha az előző dián leírt OEM driver telepítése nem sikerült, akkor próbáljuk meg manuálisan telepíteni:
- http://developer.android.com/tools/extras/oem-usb.html#Win7
- Amikor a leírás azt mondja, hogy keressük meg az <sdk>\extras\google\usb_driver\ mappát, akkor ehelyett le is tölthetjük a driver csomagot itt:
- http://developer.android.com/sdk/win-usb.html
- Itt: Click here to download the latest Google USB Driver ZIP file.

Android telefonunk felismerése

- Dugjuk be a gépünkbe az Android telefonunkat
- A driver manuális telepítéséhez:
- Start menüben keressük meg az Eszközkezelő-t (Device Manager)
- Ott az ismeretlen eszközöknél (esetleg máshol) keressük meg a telefonunkra utaló eszközt (általában a telefonunk márkája vagy valamilyen száma szokott ennek az eszköznek a nevében lenni)
- A telefon eszközünkre jobb klikk -> Illesztőprogramok frissítése... -> Illesztőprogramok keresése a számítógépen (manuálisan) -> Választás a számítógépen található illesztőprogram-listából -> tovább -> tallózás... keressük meg a kibontott Google USB Drive mappát -> OK -> tovább -> válasszuk ki az ADB (Composite) drivert -> kész

MainActivity.java

- A fő (induló) Activity java forráskódja megtalálható az app/java/<com.oldalam.tutorial>/MainActivity.java fájlban
- Láthatjuk a forráskódunkban, hogy a MainActivity osztályunkban lesznek az Acitivity életciklusához tartozó eseménykezelések
- Ez a class az AppCompatActivity leszármazottja, amiben az alapértelmezett Android programozáshoz tartozó függvények meg vannak írva, mi ezeket a függvényeket szeretnénk kiegészíteni, módosítani igényeink (alkalmazásunk) szerint

Metódus túlterhelés

- A Javaban tanultak szerint létező programunkba beépített függvényeket (metódusokat) a leszármazott (gyerek) osztályban teljesen átírhatjuk, túlterhelhetjük (@Override) (így a programunkon belül ahol ez a metódus meghívódik, a mi általunk átírt verziója fut le)
- Mivel teljesen átírtuk a metódust, ezért az eddig előre megírt, bevált utasítások is elvesznek, amik az Android alkalmazásunkban fontosak lennének, ezért ahhoz hogy ne kelljen ezekkel is foglalkozni és a lényegre tudjunk koncentrálni, ezért a
- super.túlterheltmetódushívás(hozzátartozó paraméterváltozó)
- Ősosztály metódushívásával az alkalmazáshoz fontos programkódok lefutnak és nekünk ezzel nem kell foglalkoznunk
- A sablonból ezek hívások általában jó helyre kerülnek

Metódus túlterhelés

- Mivel előre jól megírt beépített metódusokat szeretnénk túlterhelni, ezért jó lenne, ha minden ilyen metódusnak a kerete, sablonja automatikusan beíródna a forráskódunkba (így nem kell kitalálnunk az alapját és igény szerint kiegészíthetjük)
- Erre lehetőségünk van, ha a menüsávon Code->Override Methods…-ra megyünk (vagy kódban jobb klikk->Generate->Override Methods…)

R (Resource) class

- A programkódunkban bárhol hivatkozhatunk bármilyen képernyőnkön levő elemünkre az egyéni android:id (azonosító attribútuma) alapján
- Ezek az azonosítók (ne felejtsük el a + jelet) bekerülnek az R nevű Resource classba, ezen belül az R.id-nél vannak az azonosítóink felsorolva
- Ha egy konkrét típusú elemet módosítani szeretnénk a kódunkban, akkor megtalálhatjuk az elemünket az id-je alapján így:
- EditText editText = (EditText) findViewById(R.id.edit_message);
- Ennek a változónak a típusáról és típuskonverziójáról (kényszerítéséről) nekünk kell gondoskodnunk, értelemszerűen olyan típust kell neki adnunk, amilyen típusú az adott elemünk

Új Activityre adatküldés

- Res/layout/content_main.xml fájlt szerkesszük, adjunk hozzá egy gombot:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

- A megfelelő string name=„button_send” adatforrást készítsük el a values/strings.xml fájlban
- Ha a felhasználó ráklikkel a gombra, akkor a sendMessage saját függvényünk fog meghívódni
- http://developer.android.com/training/basics/firstapp/starting-activity.html

Új Activityre adatküldés

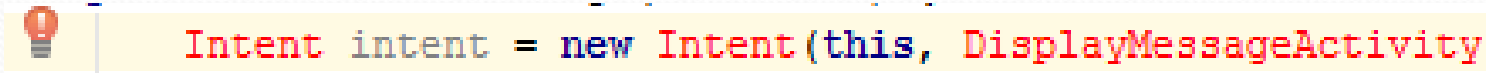
- A MainActivity.java forráskódban hozzunk létre egy új saját függvényt a MainActvity classon belül:

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
}
```

- Az Intent class segítségével azt fejezzük ki, hogy szeretnénk (szándékunkban áll) csinálni valamit egy másik Activityvel, mi most üzenetet szeretnénk neki átadni. Az Intent egyik konstruktora itt meghívódik, első paramétere: this (a mostani Activityt), második paramétere: DisplayMessageActivity.**class** (a másik Activityvel) szeretnénk összekötni, ami még nem létezik

Import

- Ha valamit (classt, metódust, konstansokat, változókat…) szeretnénk használni az API-ból, akkor a szándékunknak megfelelő függvénykönyvtár eléréseket be kell importálni (elérhetővé kell tenni a forráskódunknak), ezt az import kulcsszó segítségével tehetjük meg
- Nincs minden alapértelmezetten importálva, mert így a fordítás során csak azokat fogja a fordító összerakni amiket használunk
- Ha valamilyen import hiányzik, ezt egy piros felkiáltójel jelzi az IDE-n belül a megfelelő helyen, ha erre ráklikkelünk megmondhatjuk az IDE-nek, hogy importálja be a hiányzó részeket, vagy ugyanezt megtehetjük az ALT+Enter kombinációval a hiba helyén



sendMessage()

- A teljes függvényünk így fog kinézni:
- public void sendMessage(View view) {
  Intent intent = new Intent(this, DisplayMessageActivity.class);
  EditText editText = (EditText) findViewById(R.id.edit_message);
  String message = editText.getText().toString();
  intent.putExtra(EXTRA_MESSAGE, message);
  startActivity(intent);
  }
- Szabályok: public void legyen, paraméter: View view
- Természetesen ehhez egy android:id=”edit_message” attribútumú EditText típusú elemre (input text szöveg doboz) van szükségünk a kijelzőnkön
- Még létre kell hoznunk egy statikus konstanst (final) a MainActivity class elején:
- public final static String EXTRA_MESSAGE = "com.mycompany.myfirstapp.MESSAGE";

Magyarázat

- Az EXTRA_MESSAGE segítségével az Appunkhoz köthetünk globális String szövegeket, amiket minden Activitynk elérhet
- Ezt az intent.putExtra(…) metódushívással tettük meg
- Szerencsére a függvény többi része olvasható és érthető, amit beírtunk az EditText szövegdobozunkba, azt a szöveget továbbküldjük egy másik Activitynek, ahol ezt a szöveget feldolgozhatjuk
- Ehhez létre kell hozni a DisplayMessageActivity classt

DisplayMessageActivity

- Új Activityt létrehozhatunk az Android Studioban, ha a java mappára jobb klikk->New->Activity->Blank Activity
- Activity Name: DisplayMessageActivity
- Hierarchical Parent: com.oldalam.myapplication.MainActivity
- Package Name: com.oldalam.myapplication
- ->Finish

DisplayMessageActivity.java

- A content_display_message.xml fájban adjunk egy új attribútumot (azonosító nevet) a RelativeLayout főelemünknek:
- android:id="@+id/content”
- A hiányzó dolgokat importáljuk be, készítsük el
- A java forráskód így nézzen ki:

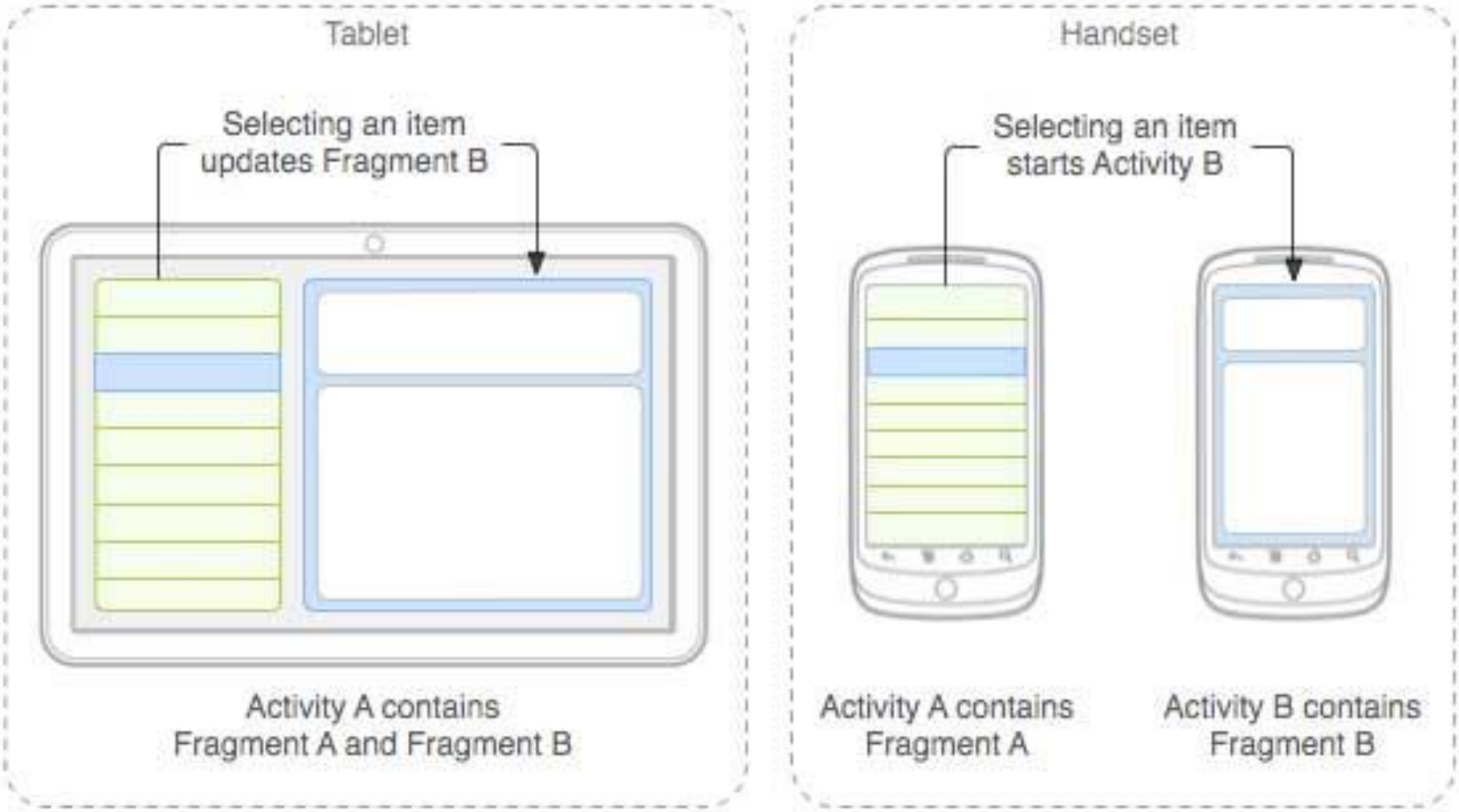
DisplayMessageActivity.java.txt

App Manifest

- manifests/AndroidManifest.xml fájlban találhatóak az Activity tulajdonságok, elnevezések
- Itt lehet beállítani az Alkalmazásunk nevét, ikonnevét
- A package (csomagunk) nevét
- Itt lehet beállítani az alkalmazásunk engedélyeit, hozzáférését a <uses-permission> tag és attribútumok segítségével (pl.: rezgés, kamera, root hozzáférés…)
- http://developer.android.com/guide/topics/manifest/uses-permission-element.html

Fragment

- Fragmenteknek hívják a mellék Activityket, ezeknek is lehet saját Layoutjuk, kaphatnak, küldhetnek adatokat és más fő Activityk is meghívhatják, felhasználhatják (pl. ilyen a lebegő menü is)
- A Fragment classt kell leszármaztatni
- Életciklusát hasonlóan képzeljük el mint az Activityknek, a onCreateView(…) metódus hívódik meg először, amikor létrejön a Fragment
- <fragment> tag segítségével definiálhatjuk


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Fragment

- A Fragment mappában levő java forráskódokat másoljuk be a projektünk megfelelő (AndroidStudioProjects\MyApplication\app\src\main\java \com\oldalam\myapplication) helyére
- És a Layoutokat a fragmentekkel meg a (AndroidStudioProjects\MyApplication\app\src\main\res\ layout) mappába

Rezgés bekapcsolása

- Ahhoz, hogy használhassuk a rezgést, az AndroidManifest.xml fájlban engedélyt kell kérni ehhez: a <manifest> főtagen belülre írjuk be:
- **<uses-permission android:name="android.permission.VIBRATE"/>**
- Hozzunk létre egy új gombot a content_mainben, amihez rendeljünk egy saját függvényt, attribútuma: **android:onClick="rezegj"**
- Hozzuk létre a függvényt a MainActiviy classon belül

rezegj.txt

Hang/Zene lejátszása

- A res mappán belül hozzunk létre egy raw mappát
- A hang/zenefájlokat (.mp3) a res/raw mappába rakjuk
- Másoljuk most be a peanready1.mp3 fájlt a res/raw mappába
- Hozzunk létre egy új gombot a content_mainben, amihez rendeljünk egy saját függvényt, attribútuma: **android:onClick="hang"**

- Ha az Android Studio megkérdezi a matching type-t, akkor válasszuk az open matching files in associated application-t
- http://www.tutorialspoint.com/android/android_mediaplayer.htm

Giroszkóp használata

- Hozzunk létre 3 textView1, textView2, textView3 Plain TextView Widgetet
- Ennek a kódnak a segítségével ez a három TextView a háromféle forgatás mértékét fogja kiírni

- http://developer.android.com/guide/topics/sensors/sensors_motion.html

Kamera használata

- AndroidManifest.xml-ben:
- <uses-permission android:name="android.permission.CAMERA" />
- <uses-feature android:name="android.hardware.camera" />
- A Camera importnál az android.hardware.Camera-t válasszuk

Bluetooth használata

- Sajnos a bluetooth részletes bemutatására nincs időnk, mert sok dolgot kell hozzá implementálni, akit érdekel itt tájékozódhat:
- http://developer.android.com/guide/topics/connectivity/bluetooth.html

Eseményvezérelt programozás

- Ugyebár a java forráskódunkban eseményekhez tartozó metódusokat terhelünk túl, ezek a metódusfüggvények akkor futnak le, amikor a megfelelő esemény bekövetkezett (létrejött új Activity, valamit a felhasználó kiválasztott…)
- Saját eseménykezelést kétféleképpen csinálhatunk:
- Vagy létrehozunk egy megnyomható gombot pl és annak az attribútumában (android:onClick=„fuggvenynevem”) megadunk egy függvénynevet amit beleírunk a MainActivity osztályunkba
- Vagy az Activity létrejöttekor (onCreate) definiálunk egy eseményfigyelőt
- (az Android Studio valójában ezt a két esetet ugyanúgy kezeli, ez csak nekünk könnyítés)

Saját eseményfigyelő – FloatingActionButton példáján

- Ha egy Blank Activity sablont használunk (vagy olyat aminek a jobb alsó sarkában van egy körgomb benne egy levéllel)
- Akkor az onCreate metódusban láthatjuk a FloatinActionButton inicializálását, itt történik meg az eseményfigyelő hozzácsatolása
- A fab.setOnClickListener(...); metódushívás paraméterében egy úgynevezett Anonymus (névtelen) objektumot definiálunk (létrehozunk a new kulcsszóval) aminek az egyik metódusát (**public void** onClick(View view)) ott helyben túlterheljük. Ez a metódus akkor fut le, ha a felhasználó ráklikkelt erre a levél körgombra
- A sablon létrejöttekor egyetlen egy dolog történik ráklikkeléskor: egy meghatározott idejű snackbar szövegértesítés jelenik meg alul, értelemszerűen átírhatóak a paraméterek

Switch, radio button elem

- Nemcsak egyszerű gombokat rakhatunk bele az alkalmazásunkba, hanem akár kapcsolókat vagy rádió gombokat is…
- Ezekhez ugyanúgy rendelhetünk eseményfigyelőt az előzőek alapján, de talán egyszerűbb, ha az xml fájlban definiáljuk az andorid:onClick=„sajatfuggveny” attribútumot, így ha valamire ráklikkelünk akkor a saját függvényünk fog lefutni, ennek a megírását láthattuk már
- A rádió gombokat csoportosítsuk a <RadioGroup> taggel (ezen belül legyenek a <RadioButton> tagek)
- Ha rádió gombokat rakunk be és csak egyet szeretnénk ezekből egyszerre bekapcsolni, akkor írjuk meg így a saját függvényünket:

Radio Gomb példa.txt

- Ekkor mindegyik rádió gomb ugyanerre a saját függvényünkre mutasson (mindegyikben ugyanaz az andorid:onClick=„sajatfuggveny” attribútum)

Navigation Drawer

- Navigation Draver Activity sablon használatával a legmodernebb menü forráskód alapja készül el
- A menu/main.xml fájlban a hagyományos menü itemek találhatóak, ezek akkor jelennek meg, ha a telefon beállítás gombjára klikkelünk
- A menu/activity_main_draver.xml fájlban a oldalról behúzható navigációs menü elemei láthatóak, módosíthatóak
- http://developer.android.com/training/implementing-navigation/nav-drawer.html

Navigation Drawer

- Ha egy ilyen sablonnal kezdünk egy új projektet, akkor egy pár új elemet láthatunk
- A res/menu mappában láthatjuk a activity_main_drawer.xml és a main.xml menu resource fájlokat
- A main.xml fájlban az android telefonunk beállítások gombjával előjövő menüt állíthatjuk be, ez a fejléc sávon (action bar) jobb oldalt is megjelenik
- Az activity_main_drawer.xml fájlban az Appunkon belüli menüt definiálhatjuk, ami balról behúzható vagy a fejléc sávon (action bar) a bal oldali menü gombbal megnyitható

res\menu

- Ezekben az xml fájlokban megadhatjuk a menüelemek listáját
- Az egész xml fájl <menu>-vel kezdődik és </menu>-vel záródik
- Láthatatlan csoportokat képezhetünk, ha a <group> taget használjuk, látható csoportokat az <item><menu>….<menu></item> segítségével csinálhatunk
- Menu elemeket az <item> taggel definiálhatunk

Menü item attribútumok

- android:id – új egyedi azonosítója az elemnek, amivel majd hivatkozhatunk rá
- android:icon – a menü elem neve melletti ikon (drawable)
- android:title – a menü elem neve
- A activity_main_drawer.xml és a main.xml menü fájlokban hozzunk létre saját menü elemeket, csoportokat

Menüvel ellátott MainActivity

- A MainActivity.java forráskódban láthatjuk, hogy a főosztályunk az NavigationView.OnNavigationItemSelectedListener osztályt implementálja (implements), így több metódus túlterhelésére van lehetőségünk
- A navigation menü inicializálása és a fejléc sávhoz (action bar) való kötése és a megfelelő eseményfigyelők az onCreate Activity metódusban vannak definiálva (ez ugye akkor fut le amikor az Activity (itt most a főprogramunk) elindul)
- onCreateOptionsMenu metódus a telefonunk beállítások gombjával előjövő menüt „fújja” fel (inflate)
  - Az ezekhez tartozó elemkezelés az onOptionsItemSelected metódusban van megírva (if feltételekkel kell az id-ket összehasonlítani)
- onNavigationItemSelected metódus a navigation (bal oldalról behúzható) menüben található kiválasztott menüelemek kezelését végzi

onNavigationItemSelected(MenuItem item)

- Ez a metódus akkor hívódik meg, amikor a felhasználó valamilyen navigation menüelemet kiválaszt, ekkor a kiválasztott elemhez tartozó id-t megtudhatjuk az
  int id = item.getItemId();
  segítségével.
- Majd megvizsgáljuk, hogy ez az id melyik definiált menü id-vel egyezik meg, ha találtunk egyezést valamelyik feltételben, akkor a megfelelő feltétel blokkon belülre írhatjuk, hogy mi történjen ekkor (kezdődjön új activity esetleg változzon/jelenjen meg valami a képernyőn, fragmentkezelés…)

WebView

- AndroidManifest.xml-ben:
- <uses-permission android:name="android.permission.INTERNET" />
- Hozzunk létre egy WebView widgetet a content_main.xml-ben, ebben jelenik meg a weboldalunk
- Pl onCreate(…) metódusban:
- WebView myWebView = (WebView) findViewById(R.id.webview);
  myWebView.loadUrl("http://www.example.com");
- Ha létrejött a kapcsolat a webszerverrel, akkor a weboldalon definiált (HTML5, PHP) segítségével kommunikálhatunk a szerverrel
- http://developer.android.com/guide/webapps/webview.html

Általános App programozás HTML5 segítségével

- A HTML5 megjelenésével óriási lehetőségek nyíltak meg az általános multiplatformos programozás terén
- Mivel sok a hasonlóság a mobil elemek és HTML elemek leírása között (.xml ~ .html), ezért célszerű lenne, ha a HTML5 nyelvét felhasználhatnánk mobilprogramozásra is
- Ennek a támogatására létrehoztak ún. fejlesztői frameworköket, új cél a programozói iparban az, hogy egy alkalmazást elég legyen egyetlen egy nyelven megírni és ezt bármilyen céleszközre (platformra) lehetőségünk legyen lefordítani kódmódosítás nélkül!
- https://software.intel.com/en-us/intel-xdk

Grafikus felület programozás

- Akár Androidra is létrehozhatunk 3D/2D grafikus Appokat, játékokat
- Ezt az OpenGL segítségével tehetjük meg
- WebGL segítségével akár weboldalakra is van lehetőségünk grafikus felületet programozni (pl.: darkorbit játékot)
- A natív OpenGL API használatát nem ajánlom, mert nehézkes a kezelése, helyette a libGDX függvénykönyvtárat ajánlom, mert segítségével akár PC-re (Windows/Linux…) is és Androidra, IOS-re, weboldalra is programozhatunk grafikus felületet
- https://libgdx.badlogicgames.com/features.html
- https://github.com/libgdx/libgdx/

Játékmotorok

- Játékprogramozásra ajánlom a játékmotorok használatát, ezekben beépített fizikakezelés, modellkezelés, rengeteg programozói könnyítés található, így nem fogunk elveszni a részletekben és koncentrálhatunk a játékfejlesztésre
- Hatalmas előny az is, hogy ha egy motorban létrehozunk egy projektet akkor azt egyből más rendszerekre/platformokra is lefordíthatjuk
- https://unity3d.com/
- https://unity3d.com/unity/multiplatform
- https://www.unrealengine.com/what-is-unreal-engine-4

Android API Dokumentáció

- http://developer.android.com/reference/packages.html
- Itt minden beépített class, függvény, változó, enum… leírása, dokumentációja, magyarázata, használata megtalálható
- Minden függvénykönyvtár dokumentációhoz tartozik ehhez hasonló felépítésű weboldal