MOBIL KÉPELEMZÉS ÉS GRAFIKA

MSC KURZUS ANYAGA (2020. TAVASZI VERZIÓ)

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

Elektronikusan elérhető anyagok

- **Feladatok és példaprogramok**
  - Az előadás témakörök végén feladatkiírások találhatók
  - Több témakörhöz feladatmegoldások és/vagy példaprogramok érhetők el

- **A kurzus témaköreinek sajátossága miatt a kurzus anyaga folyamatos frissítésen megy keresztül**
  - A legfrissebb előadás anyag és példaprogram gyűjtemény elérhető a kurzus anyagának elektronikus oldalán:
    - http://www.inf.u-szeged.hu/~tanacs/mobilkeg/

MOBIL KÉPELEMZÉS ÉS GRAFIKA

ESZKÖZÖK, ALKALMAZÁSOK ÉS FELADATOK

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

Mobil eszközök

- **Elterjedtség**
  - Több milliárd potenciális tulajdonos
  - Magyarország: több előfizetés, mint lakos
- **Felhasználás**
  - *Kommunikáció*
    - Klasszikus: hang, SMS
    - Internet alapú: Facebook, üzenetküldők, email, …
  - *Információforrás*
    - Időjárás, tőzsde indexek, árfolyamok, hírek, útvonalinformáció, …
  - *Szórakozás*
    - Játék, könyvolvasás, multimédia
- **„Buta” telefon (feature phone)**
  - Alapfeladatokra (hanghívás, SMS)
  - Rendszerint zárt rendszer
  - Fejlesztés korlátozott (pl. Java ME)
- **Okostelefon (smartphone)**
  - Internet elérés, többfajta kommunikációs csatorna
  - Bővíthető „tudás”
  - Fejlesztők számára nyitott fejlesztői környezet
  - Alkalmazásboltok!

Mobil eszközök

Történeti áttekintés dióhéjban

- **Első kézi mobiltelefon** (1973)
  - Motorola (Martin Cooper → Joel S. Engel)

- **Manager kalkulátorok** (1980-as, 1990-es évek), EPOC
  - Monokróm vagy szürke kijelző
  - Digitális titkári funkciók, nincs telefon!
  - Minőségi, strapabíró termékek
  - Erősen limitált darabszám!

- **Apple Newton platform** (1987-1998)
  - Kézírás felismerés, digitális titkári funkciók
  - Többféle gyártótól (pl. Motorola)
  - Technológiai szint még nem megfelelő!

Történeti áttekintés dióhéjban

- **Kommunikátorok (1996-2002)**
  - Nokia Communicator sorozat (9000, 9110, 9210)
    - Telefon, billentyűzet, titkári funkciók, diktafon
    - Email, fax, adatkapcsolt internet,
    - Mobil távközlés még nem elég fejlett

- **PDA korszak (1996—2008)**
  - Digitális titkári funkciók
  - Telefonos funkciók nélküli, PC-hez dokkolt szinkronizálás
  - **Palm OS**: 1996-2007 (Palm, Handspring, Sony, ...)
    - Monokróm, szürke, majd színes kijelzők
    - Hosszú működési idő (több hét), akár ceruzaelemről is
    - Később: nyitás a multimédia felé (későn), integráció telefonnal
  - **Windows CE**: 1996-
    - Asztali Windows-szerű, beágyazott rendszerek
  - **Pocket PC** (2000-2002), **Windows Mobile** (2003-2010)
    - Multimédia előretörése
    - PDA telefon nélkül, smartphone (nem érintőképernyős), PDA telefonnal
    - iPhone megjelenése után erős visszaesés

Történeti áttekintés dióhéjban

- Modern okostelefonok (2007-)
  - PDA-k és mobiltelefonok konvergenciája + mobil/adat infrastruktúra fejlődése
  - RIM Blackberry, Symbian
  - Apple iPhone, Android, Windows Phone, …
    - 2007-től radikális átrendeződés a piacon!

- Táblagépek
  - PC-k visszaszorulása

- Viselhető eszközök
  - Okosórák, fitnesz karkötők
  - Google Glass
  - Bio szenzorok

Jelenlegi jelentősebb okostelefon platformok

- **Apple iPhone, iOS**: 2007-
  - Felhasználói élmény, könnyű kezelhetőség
  - Kevés készüléktípus, zárt ökoszisztéma
    - Original, 3G, 3GS, 4, 4S, 5, 5S és 5C, 6, 6s, ...; iPad sorozat
  - Integrált online alkalmazásbolt megjelenése!

- **Android**: (2005) 2007-
  - Nyílt forráskódú operációs rendszer
  - Kezdetben Google + 79 cég konzorciuma
  - Fragmentáció: Rengeteg gyártó, sokféle hardver
    - Elsők: HTC Dream (2008 október), HTC Magic (2009 április), HTC Hero (2009 június)
    - Tengernyi további készülék rengeteg gyártótól

- **Windows Phone**: 2010-2019
  - A Windows Mobile vonallal radikálisan szakító új Microsoft platform
  - Licenszelhető, de erős nyomás a gyártókra (pl. UI és hardver kapcsán)
  - Megszűnt, a Microsoft inkább Android és iOS platformokra koncentrál

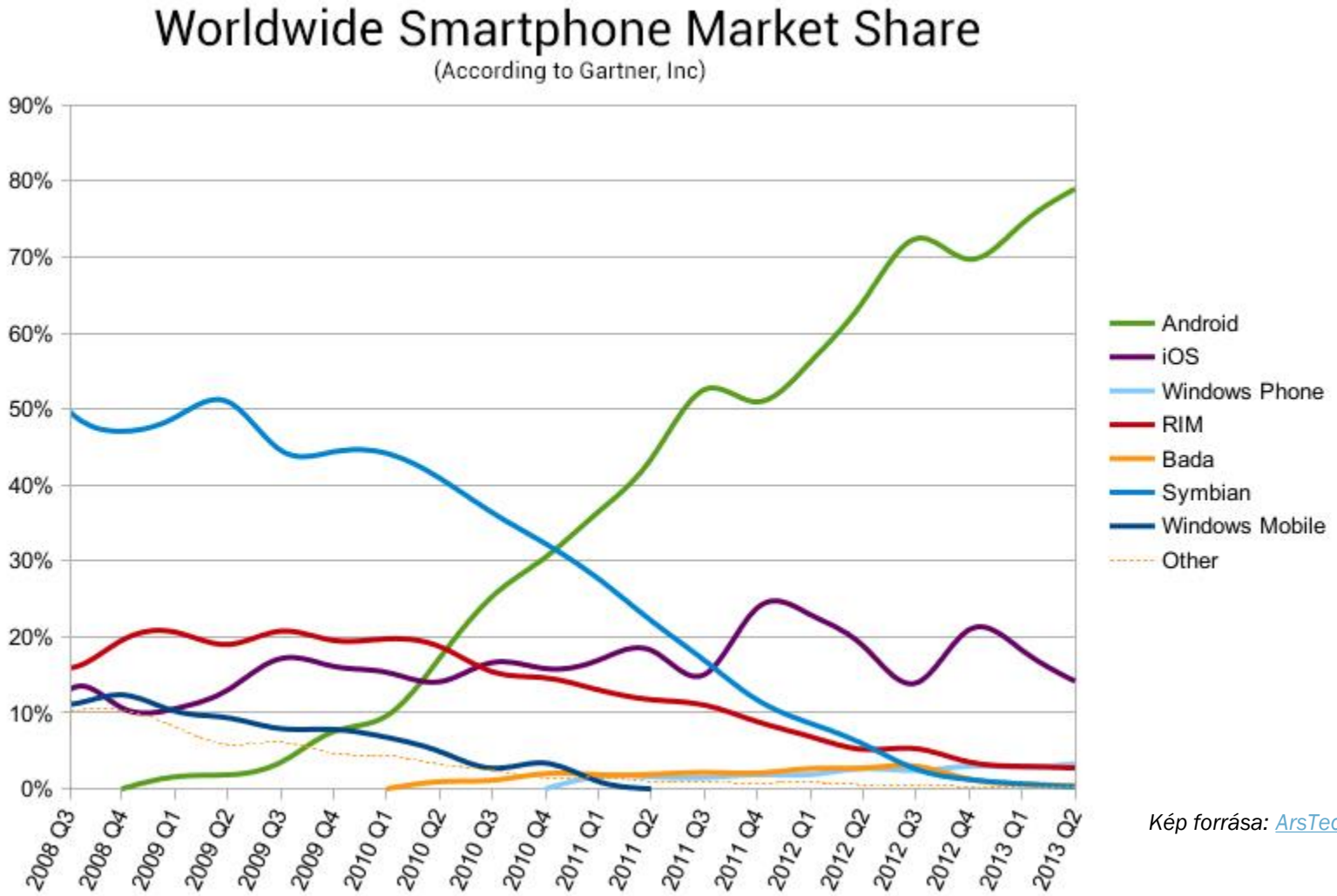
Visszaszorult okostelefon platformok

- **Nokia kommunikátorok**
  - GEOS rendszerű Nokia Communicator 9100, 9110 (1996, 1998)
- **Symbian OS:** 2001 -2011
  - Konzorcium a Windows Mobile ellensúlyozására (Nokia + társai)
  - Okostelefonok, kommunikátorok
  - 2008: Nyílt forráskód, 2011: Nokia WP7-re vált
  - Sokáig piacvezető, de támogatása egyik napról a másikra (még piacvezetőként!) megszűnik
  - Nehézkes programozás, kezelhetőségben súlyos lemaradás
- **Samsung Bada**: 2009-
  - Zárt operációs rendszer, Samsung saját telefonjain → Tizen OS
  - Samsung szánta „menekülési pályának”, ha az Androiddal gond lett volna
- **Meego**: 2010-2011
  - Maemo (Nokia) + Moblin (Intel)
  - 2011: Nokia kísérleti ágra redukálja, Nokia N9 megjelenése
  - A Tizen OS és a Sailfish OS alapja lesz

Egyéb okostelefon platformok

- **RIM Blackberry**: 1999-
  - Biztonság orientált zárt rendszer, **céges környezetben a legnépszerűbb a 2000-es évek közepén**
  - QNX-alapú új rendszerre átállás, de a zuhanást nem állítja meg → Android
  - 2016: hardveres részleg kiszervezése, **szoftveres** biztonsági csomagokra koncentrál a cég
  - **2020: a hardveres rész teljesen eltűnik**
- **HP (Palm) webOS**: 2009-2011
  - iPhone-szerű kezelhetőség, de az áttörés anyagi források hiánya miatt elmaradt
  - 2011. augusztus: a HP leállítja a webOS készülékek gyártását, a platform sorsa bizonytalan
  - **LG okostelevíziókon jelenik meg**, alaposan átalakítva az UI-t (2014)
- **Tizen OS**
  - Meego + Bada ⟶ Tizen OS néven, Samsunggal együttműködve él tovább
  - 2014: **Samsung okosórák** és kamerák rendszere, okosTV-k felé nyitás
  - 2015: első Samsung telefon megjelenik Indiában (ultraolcsó szegmens), elhal
- **Firefox OS:** 2013-2015
  - Ultraolcsó szegmens
  - Böngésző-alapú mobil operációs rendszer
  - 2013. nyara: első készülékek megjelenése; 2015: készülék forgalmazás leállítása
- **Ubuntu for Mobile**
  - 2013. január: bejelentés, 2015. február: első készülék megjelenése
  - Megszűnt

Okostelefon platformok



Kép forrása: ArsTechnica

Okostelefon márkák részesedése

Forrás: TrendForce

Apple: iOS
Összes többi: Android

Táblagépek / tabletek

- A hardver fejlődése lehetővé teszi
  - nagy méretű (10'') kijelzők használatát,
  - akár 8-10 órán keresztül.

- Kategóriák
  - Ebook olvasók
    - Kifejezetten olvasáshoz, szürkeárnyalatos
    - E-ink kijelzők: lassú frissítési idő, de olvasáshoz szemkímélőbb, hosszú üzemidő (statikus megjelenítéskor nem fogyaszt áramot)
    - Amazon Kindle, Barnes&Noble Nook, ...
  - Táblagépek
    - Apple iPad, iPad2, új iPad dominancia
    - Android Honeycomb (csak tábla), Ice Cream Sandwich (egyesített), kevés figyelmet kap a fejlesztése, nem eléggé kiforrott
    - webOS táblákra, PC-kre (megszűnik)
    - RIM Playbook (Adobe AIR alapú fejlesztések, QNX-alapú operációs rendszer, újracsomagolt Android alkalmazások futtatása) – elhal

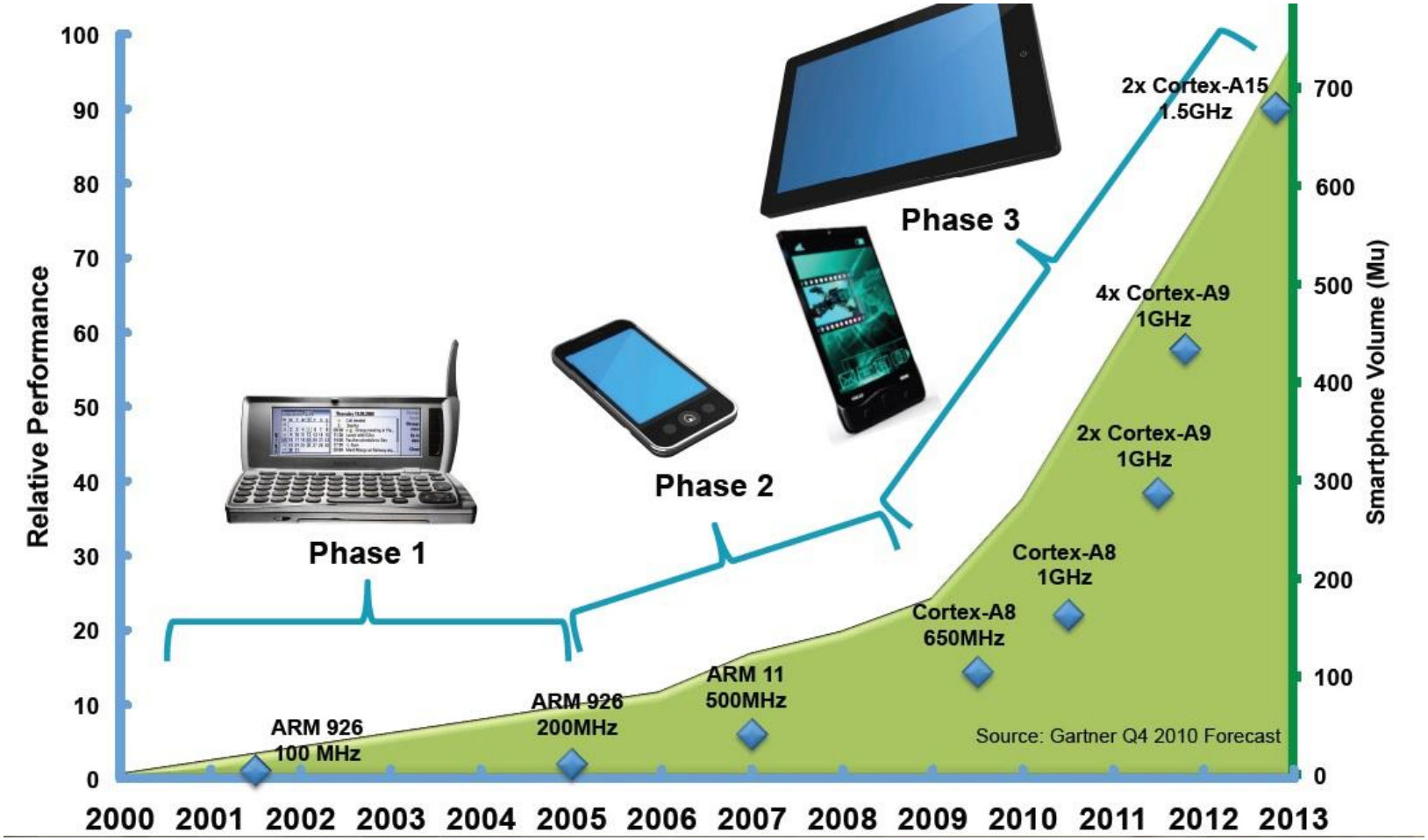
Okostelefon platformok

- Viharos és gyors fejlődés az utóbb 20 évben
  - Korábbi „egyeduralkodó” platformok tűnnek el szinte nyom nélkül (pl. Palm, Windows Mobile, Symbian)
  - **A felhasználói igényeket időben le kell reagálni! Sőt, ki kell találni azokat!**
- *Készülékek csatája* helyett *ökoszisztémák háborúja*! (S. Elop, Nokia)
  - Az üzlet a hozzáadott (fizetős) tartalomszolgáltatásban van
    - Szoftver, e-könyv, film, zene, …
  - Az egyik ökoszisztémában megvásárolt tartalmak nem mindig vihetők át másikra!
  - 2015: a tartalomszolgáltatók kezdenek a riválisplatformokon is megjelenni
    - Google termékpaletta iOS-en
    - Apple Music Android-on
    - Microsoft szoftverek Android-on és iOS-en (Office, Outlook, …)

Hardver fejlődése

- CPU+GPU
  - Rendszerint alacsony fogyasztású ARM architektúra, az Intel próbálkozásai elhalnak
  - 16 MHz ⟶ 2.3 GHz, 4-8 mag
  - 32, manapság egyre inkább 64 bites architektúra
  - **Dedikált GPU megjelenése**
- Memória
  - 1 MB ⟶ Sok GB
- Kijelző
  - Rezisztív: stylus (ceruza-alapú vezérlés) – régi technológia!
  - Kapacitív: vezérlés tapintással, egyszerre több ujj követésével
  - Felbontás: 160x160 fekete-fehér ⟶ FHD, 4K, 2.55''–7''
    - „Retina" kijelző: képpont méret annyira kicsi, hogy szabad szemmel nem látszik
  - Méretben egyre kisebb, aztán egyre nagyobb ☺
  - HDR10+, Dolby Vision támogatás
  - Autosztereoszkóp 3D kijelzők megjelenése és eltűnése
- Új érzékelők, szenzorok
  - **kamera, GPS, gyorsulásmérő, iránytű, giroszkóp**, mikrofon, mágneses mező, fénymérő, légnyomás, hőmérséklet, ...
  - Ujjlenyomat ellenőrző, retina szkenner, arc szkenner, ToF kamera, mozgásérzékelő radar
  - Nyomásérzékeny panel („force touch")

Hardver fejlődése



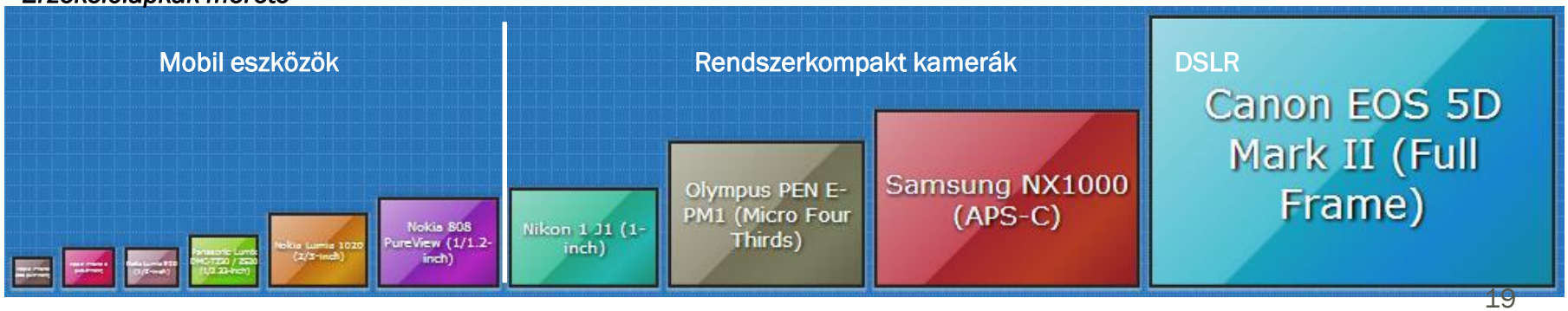
Képalkotás mobil eszközökön

- **1997: Sharp, Kyocera: kezdeti kísérletek kamerás mobillal**
- **2000: Első a piacon**
  - Sharp J-SH04, 0.1 MP felbontás
- 2000: Eyemodule Digital Camera Springboard Module for Handspring Visor
  - 320x240 felbontás
- 2002: Sony Ericsson T68i
  - Első MMS képes telefon, külön csatolható kamerával
- 2002: Sanyo SCP-5300
  - Első kamerás mobil az USÁ-ban, VGA (0.3 MP) felbontás
- **2003: több kamerás telefon, mint digitális fényképező**
- **2004: Nokia a legtöbb eladott digitális kamera gyártó**
- 2006: a mobilok fele kamerás
- **2008: Nokia legnagyobb kameragyártó (digitális és filmes együttvéve)**
- 2010: szinte minden mobilban van kamera
- 2017: többkamerás mobil képalkotás (2, 3, 5, 7)
- 2018: szoftveres mobil képalkotás előretörése

Képalkotás jellemzői mobil eszközön

- Felbontás
  - 5-8-12-20-41-108 Mpixel állókép
  - HD videó (720p, 1080p, 4K, 8K)
  - Akár 3D üzemmód is (sztereó kamerapárral)
- Jellemzők
  - A jobb mobil kamerák egy közepes kompakt kamerát kiválthatnak.
  - 2012-ben a gyártók elkezdtek a jobb kamera hardverre összpontosítani
    - Nokia, HTC, Samsung, Sony
- Előnyök
  - Mindig kéznél van
    - „ez a legjobb kamera”
  - Mobilinterneten képek rögtön megoszthatók
- Hátrányok
  - Gyenge CMOS szenzor
    - olcsó legyen
  - Gyenge optika
    - vékony, kicsi maradjon
  - Vaku, segédfény nincs vagy gyenge
  - Sok automatikus beállítás, lassú!
  - Mpixel növelése nem jó megoldás!

Érzékelőlapkák mérete



Kép forrása: Camera Image Sensor Comparison

Kamera alkalmazások

- Óriási szociális befolyás!
  - Családi
  - Politikai
    - Közvetítés, fotók eseményekről
    - Hírtelevíziók alkalmazzák
  - Tudományos
    - Remek lehetőség a szökőárak tanulmányozására „2011. márc. 11. 12:00 Ausztrália A BBC tudományos rovata szerint eddig sosem látott felvételek készültek a szökőárról, amit a jövőben a tudósok elemezhetnek, hogy jobban kiismerjék a cunamik természetét.”
    - Meteorológiai jelenségek megfigyelése (Több tornádó van-e manapság, mint régen?)
    - …
- Jegyzetelés
  - Plakátok, nyitva tartás, …
- Kódok olvasása
  - Vonalkód, QR kód
- Képek készítése, feldolgozása, szerkesztése, javítása
  - Képmontázsok, HDR, time lapse, mozgó objektumok eltüntetése, …
- Képek szerverre feltöltése, megosztása
  - Instagram, Twitter
- Kép-alapú keresés, kiterjesztett valóság
  - Kliens-szerver alkalmazások
- Képelemzés
  - Arcfelismerés, detekciók, …
- Videótelefonálás

Kamera alkalmazások

- QR kód, vonalkód
  - Termékazonosítás, visszakeresés
  - Névjegykártya információk
  - URL
  - Tetszőleges szöveg

- Komplex fotózási funkciók
  - Mosoly detektálás, pislogás detektálás
  - Utófeldolgozás
  - HDR képalkotás
  - Panoráma kép készítése (akár szenzoros támogatással)
  - 3D kép készítése

Kép alapú keresés (kliens-szerver)

- Google Lens (Android)
  - Épületek, festmények
  - Vonalkódok, QR kódok
  - Névjegykártya felismerés
  - Könyvek, termékek
  - Sudoku megoldása

- Szakdolgozat témák
  - Óraállás bejelentő
  - Kép-alapú Sudoku megoldó
  - Gépkocsi-típus felismerés

Kép forrása:Google System BlogSpot

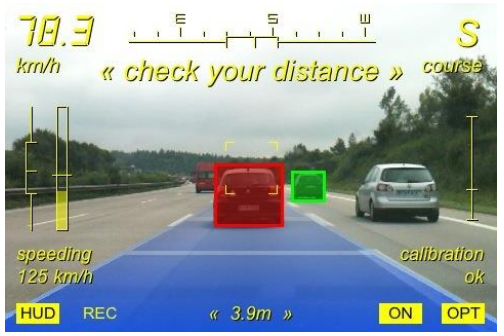
Kiterjesztett valóság

- Okostelefonok „legtöbb tudását” kihasználja
  - Kamera élőkép + pozíció információ + szenzor információk + helyfüggő adatok hálózatról + 2D/3D számítógépes grafika

- Alkalmazások
  - Layar (Android, iPhone)
  - Wikitude World Browser
  - Word Lens (iPhone)
    - Spanyol-angol fordító
  - Játékok
  - ARCore, AR Kit

Számítógépes látás

- Alkalmazások
  - OCR (karakterfelismerés)
    - Google Docs: lefotózott szöveg felismerése (angol)
  - Word Lens (iOS), CamTranslator (Android)
    - Kamera élőképen szavak felismerése, fordítása
  - Augmented Driving (iPhone)
    - Sávok, sávváltások detektálása, közeli jármű érzékelése
  - Mozgásdetekció
  - Sarokpont keresés, egyenes detektálás



- Szakdolgozat/diplomamunka téma
  - 3D objektumok röntgen nézete
  - Épületek 2D/3D tervrajzának felépítése szenzor információk alapján
  - Megfeleltetések keresése széles bázisvonalú sztereo képek között

Mobil 3D számítógépes grafika

- Hardver
  - Erős, többmagos GPU-k megjelenése
    - Pl. Nvidia Tegra platform, Qualcomm Adreno
  - Android-alapú kézi játékkonzolok
    - Nvidia Shield, Ouya, …
  - Játék stream-elés hálózaton keresztül
    - Google Stadia, xCloud, …

- Szoftver API: OpenGL ES
  - ES = Embedded System
    - Kisebb funkcióveszteség a standard verzióhoz képest
  - Nyílt szabvány 3D grafikához
  - Szinte minden platform ezt támogatja (iOS, Android, Java ME)
    - Microsoft kivételével
  - OpenGL ES 1.1: Régi rögzített csővezeték
  - OpenGL ES 2.0: Csúcspont és fregmens árnyalók, programozható csővezeték
  - OpenGL ES 3.0: Bevezetés alatt (Android 4.4 Kit Kat)

- Újabb irányvonal: Vulkan
  - Energiatakarékosabb és hatékonyabb hardver kezelés

Mobil képelemzés és grafika MSc kurzus

- **Képfeldolgozó ismeret szükséges**
  - Referencia platform: Android
- **Főbb témakörök**
  - Mobil digitális képalkotás
  - Képkészítés + kamera élőkép
  - OpenCV könyvtár használata
  - Szenzorok + grafika (OpenGL ES); Kiterjesztett és virtuális valóság
  - Detektálása, felismerése, osztályozás (képelemzés)
  - Képtartalom-alapú keresés
- **Félévközi projektmunka**

Gyakorlati feladatok

- Régi telefonos platformok kipróbálása
  - Rendelkezésre álló készülékek
    - **Handspring Visor Neo (2001):** Palm OS. Érdekességek: bővítőporton keresztül, GPS, modem, memóriakártya, GSM telefon modulok; játékvezérlő előlap; bölcső.
    - **HP iPAQ 2210 (2003):** Windows Mobile 2003 OS. CF kártya foglalat, GPS modul, ceruzás vezérlés, bölcső.
    - **HTC HD2 (2009):** Windows Mobile OS. Korának óriási méretű telefonja.
    - **Sony Ericsson Satio (2009):** Symbian OS. Nagy méretű kamera modullal.
    - **HTC Hero (2009), HTC Legend (2010):** A 2. és 3. Android rendszerű telefon.
    - **HTC EVO 3D és LG Optimus 3D (2011):** sztereó kamerapárral szerelt Android készülékek
    - **Acer Iconia (2003):** Android rendszerű, nagyméretű tablet.
    - **Samsung S4 és Note 3 (2003)**: Korának csúcskategóriás Android készülékei

Gyakorlati feladatok

- Régi telefonos platformok kipróbálása
  - Feladatok
    - Próbáljunk ki többféle készüléket!
    - Hasonlítsuk össze a kijelzők felbontását!
    - Hasonlítsuk össze a ceruzás és az ujjal érintős vezérlést!
    - Vizsgáljuk meg a készülékek rendszerének alapszolgáltatásait!
    - Figyeljük meg a készülékek méretváltozásait, a képernyő méretét a készülék méretéhez képest!
    - Hasonlítsuk össze a rendelkezésre álló memória méretét, a CPU képességeit!

MOBIL KÉPELEMZÉS ÉS GRAFIKA

MOBIL DIGITÁLIS KÉPALKOTÁS

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Felhasznált irodalom

- Péterfy Kristóf: Digitális fényképezés és eszközei. INOK Kft., 2006.
- Enczi Zoltán, Richard Keating: A digitális fotózás műhelytitkai kezdőknek. Rainbow-Slide Bt., 2011.
- Joseph Ciaglia, Barbara London, John Upton, Peter Kuhns: Absolute Beginner's Guide to Digital Photography. QUE, 2004.
- Czúni László, Tanács Attila: Képi információ mérése. Typotex kiadó, 2011.

Webes anyagok

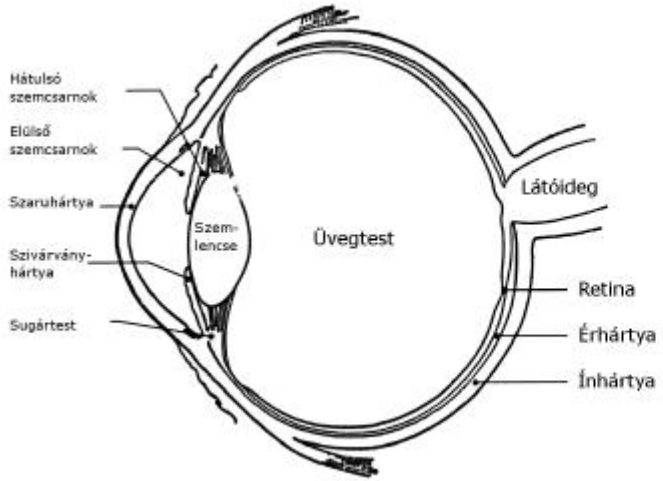
- Pixinfo Fotóelmélet cikksorozata
  - Optika, érzékelők, zárak, élességállítás, …
  - Nagyon jó animációkkal
- PHYS 6.3: Optical elements: prisms, lenses and spherical mirrors
  - Angol nyelvű leírás az optikai rendszerekről
- Free Photography Classes & Lessons online for beginner DSLR users
  - Apertúra, fókusztávolság, … (nagyon jó ábrákkal)
- Lens Focal Length Explained
  - Fókusztávolság magyarázat
- Lens Sense
  - Alapfogalmak
- Basic photography tutorial
  - Történeti áttekintéssel
- Camera Lens Types
  - Objektív típusok ismertetése

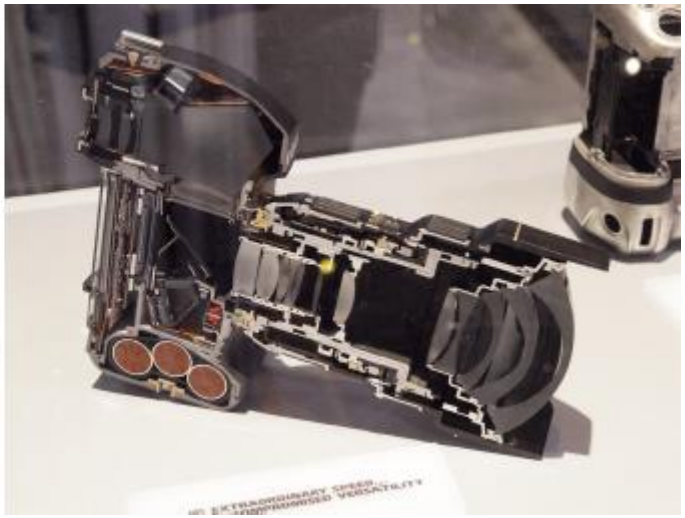
Alapelv

- Természetes és mesterséges *fényforrások*
  - Nap, közvilágítás, vaku, …
- A tárgyakról *fény verődik vissza*
  - Anyagára jellemző mértékben a fény egyes komponenseit elnyeli (szín),
  - a többit közvetlenül vagy szórva visszatükrözi (csillogó, matt, …).
- A fénysugarak *lencsék* segítségével az *érzékelőre* jutnak
  - Közeghatáron a fény útja megtörik, így irányítható a kis méretű érzékelőre
- Az érzékelő *digitális jellé* alakítja az impulzusokat
  - Mintavételezés és kvantálás

Látás és leképezés

- Emberi szem és egy DSLR kamera keresztmetszete
  - „*Képi információ mérése*” jegyzet alapján





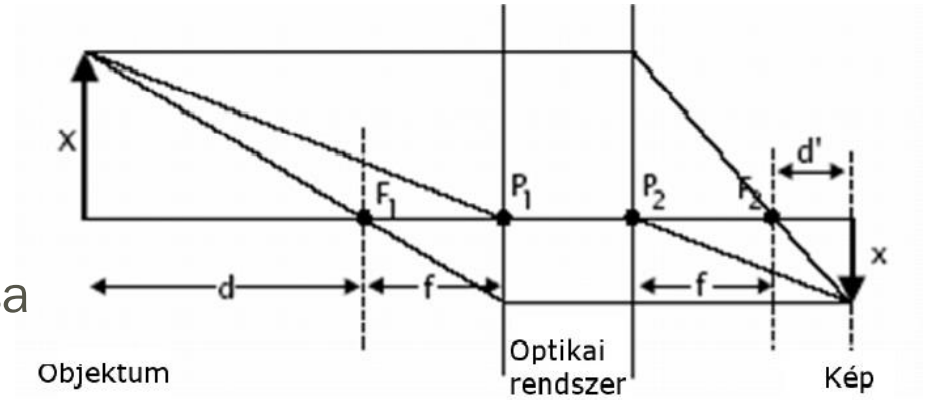
Látás és leképezés

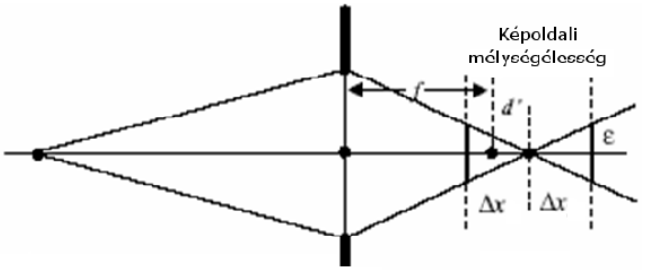
- Lyukkamera
  - Fény kis lyukon, a vetítési középponton keresztül vetül a képalkotó síkra
  - „Fejenálló” kép
  - „Camera obscura”
    - National Geographic 2013. novemberi száma,
    - Eger, Varázstorony
  - Egyszerű matematikai modell

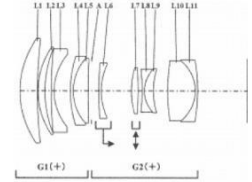
Fotó: Szász Péter – http://bp.underground.hu

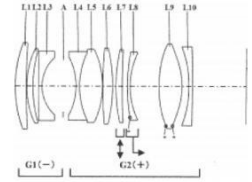
Látás és leképezés

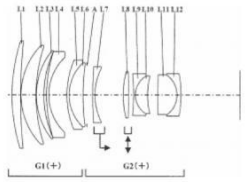
- Gyűjtőlencse
  - Éles kép: X-ből induló sugarak a képsíkon 1 pontban találkoznak
  - Apertúra (nyílás nagyság) hatása
    - Mélységélesség
    - Adott tartományon belül még elfogadható mértékű az elmosódás

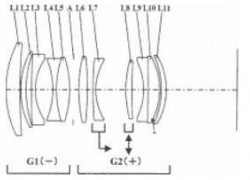


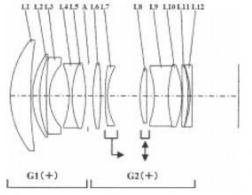


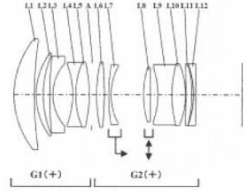

patent embodiment1: 65mm f/1.8


patent embodiment2: 25mm f/1.4


patent embodiment3: 100mm f/2


patent embodiment4: 35mm f/1.8

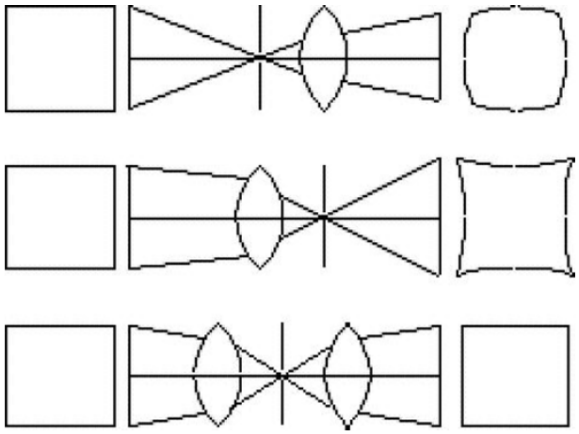

patent embodiment5: 43mm f/1.4


patent embodiment5: 43mm f/1.4

Panasonic lencse szabadalmak

Látás és leképezés

- Lencse torzítások
  - Szoftveresen korrigálható
  - Készülékek rendszerint ezt alkalmazzák is



2.12. ábra. Geometriai torzítás a rekeszszerkezet függvényében: Első sor – hordó torzítás. Második sor – párna torzítás. Harmadik sor – torzításmentes kép.

Képalkotás főbb paraméterei

- Érzékelő
  - Érzékenység (ISO)
  - Felbontás (MP)
  - Méret, típus
- Objektív
  - Zársebesség
  - Fókusztávolság
    - Zoom-átfogás
  - Rekeszérték
- Megvilágítás
  - Mennyisége, egyenletessége
    - → HDR technika (High Dynamic Range – több fotó felhasználásával)
  - Színhőmérséklet

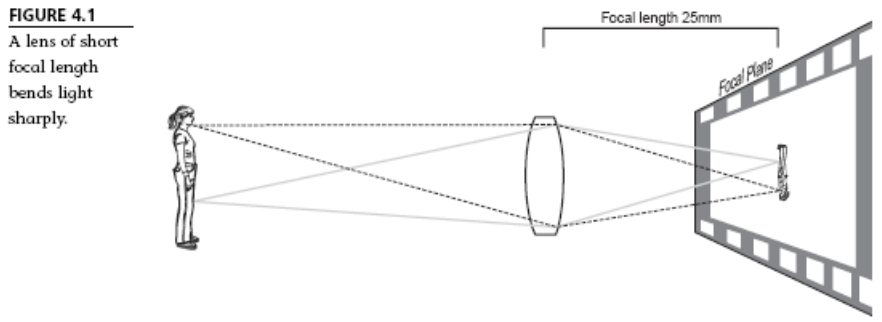


FIGURE 4.1
A lens of short focal length bends light sharply.

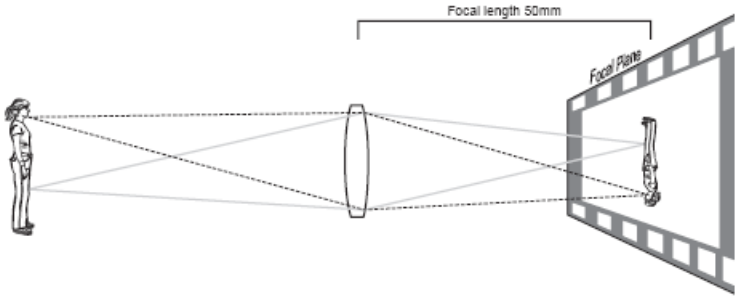


FIGURE 4.2
A lens of longer focal length bends light rays less than a short lens does.

Joseph Ciaglia, Barbara London, John Upton, Peter Kuhns: Absolute Beginner's Guide to Digital Photography

Digitális szenzorok

- CCD
  - Charge-Coupled Device; töltéscsatolt eszköz
  - 1969-ben készült az első
  - Jó minőségű, alacsony zajterhelésű képek
  - Drága előállítás, nagy áramigény!
- CMOS
  - Complementary Metal Oxide Semiconductor
  - Közepes képminőség, nagyobb zajérzékenység
  - Olcsóbb technológia
  - Jelentősen kisebb áramigény
- Összehasonlító videó a működésükről
  - Látványos animáció: így dolgozik a CMOS és a CCD érzékelő

CCD szenzor

- Tulajdonságok
  - Fém-oxid félvezető alapú kondenzátorok
    - Fényenergiából elektromos energia
  - Monokromatikus
    - Feszültséget mérik, nincs szín információ!
    - Bayer-szűrő (interpoláció szükséges!)
  - Mátrix-szerű (vagy hatszöges) elrendezés
  - Kiolvasó áramkör
- CMOS és egyebek
  - → Képi információ mérése jegyzet

Forrás: WikiMedia Commons

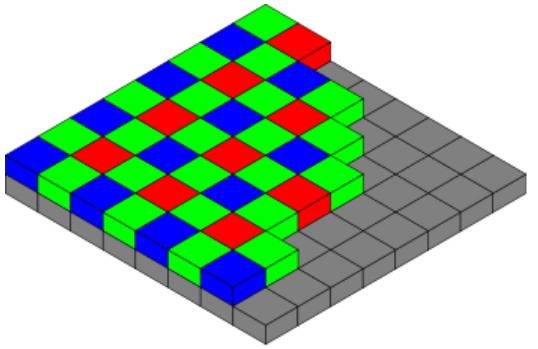
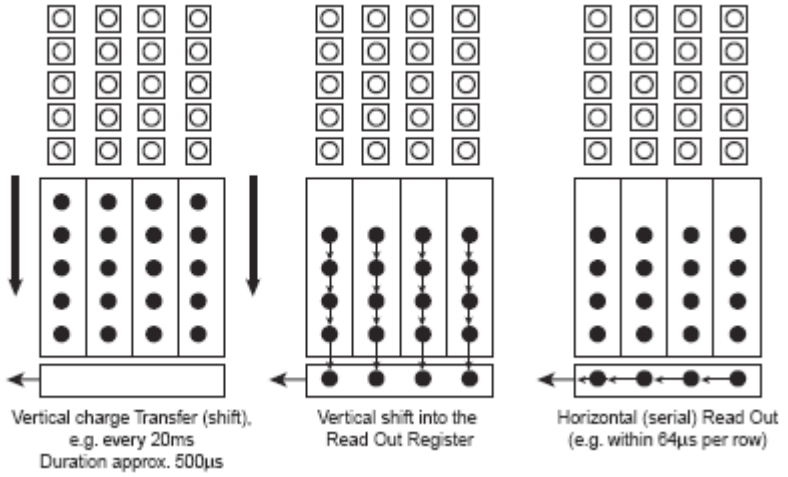


FIGURE 2.2
A light-sensitive image sensor shifts pixel values along rows to the processor.



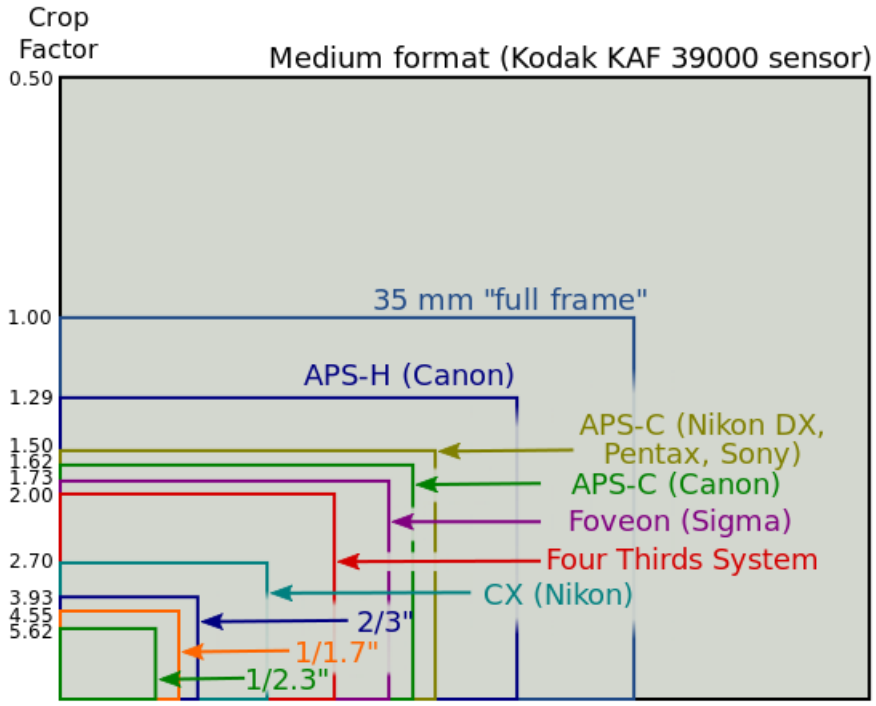
Joseph Ciaglia, Barbara London, John Upton, Peter Kuhns:
Absolute Beginner's Guide to Digital Photography

Szenzor megapixel

- Érzékelők száma a szenzoron
  - Hány millió pixelt (MP) érzékel
  - Gyakran (különösen régen) ezzel reklámozzák a készülékeket
  - Önmagában ez kevés, az érzékelő mérete is számít
- Dedikált kamerák
  - 20-30 MP, de nagy méreten
- Okostelefonok
  - Sokáig 4-8-12 MP, kis méreten
  - 2019-tól: 48, 64, 108 MP is van
  - Nagyobb MP számnál leginkább képpont összevonással dolgoznak
    - Pl. 12 MP kép készül, kisebb zajjal gyenge fényviszonyok között
- Hány megapixel kell?
  - Attól függ :D
  - Egy Full HD kijelző kb. 1 millió képpontot jelenít meg (1 MP!)
  - 10x15 cm méretű papírképhez elég kb. 12 MP

Szenzor méret

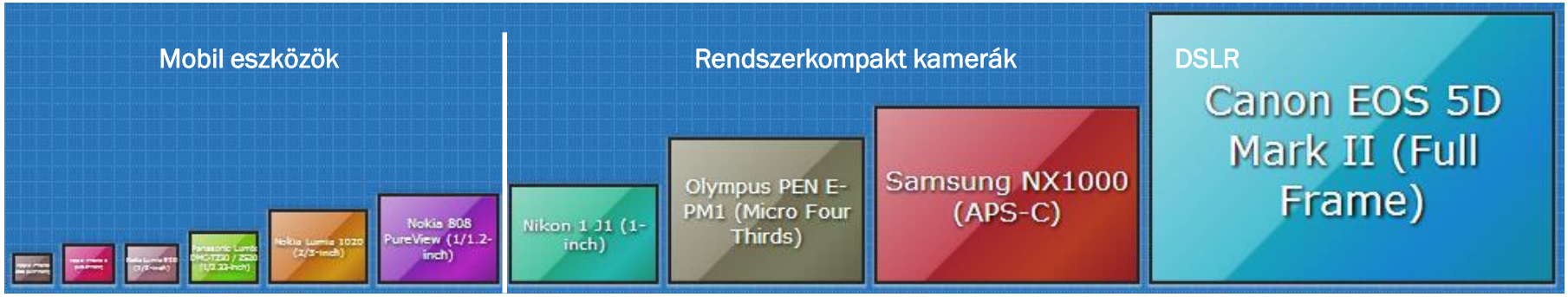
- Nagyobb szenzor méret →
  - kisebb zaj
  - jobb mélységélesség beállítási lehetőség
  - jobb fotók
  - (de nagyobb méretű objektív kell!)
- Néhány típus
  - 35 mm: Profi DSLR
  - APS-C: DSLR, rendszerkompakt
  - 4/3'': rendszerkompakt
  - 1/1.8'': felső kategóriás kompakt
  - 1/2.5'': kompakt kamerák, jobb mobil eszközök
  - 1/3.2'': átlagos mobil eszközök



Kép forrása: Engadget

Szenzor méret

- Mobil eszközökben
  - Rendszerint kisméretű érzékelő
  - Nokia két kiugróan nagyméretű mobil szenzora



Képek forrása: Camera Image Sensor Comparison

Szenzor méret

- Mobil eszközökben
  - Szenzor mérete összevethető a P&S kamerákkal
  - P&S kamerák objektíve jellemzően jobb

Kép forrása: [Camera Image Sensor Comparison]

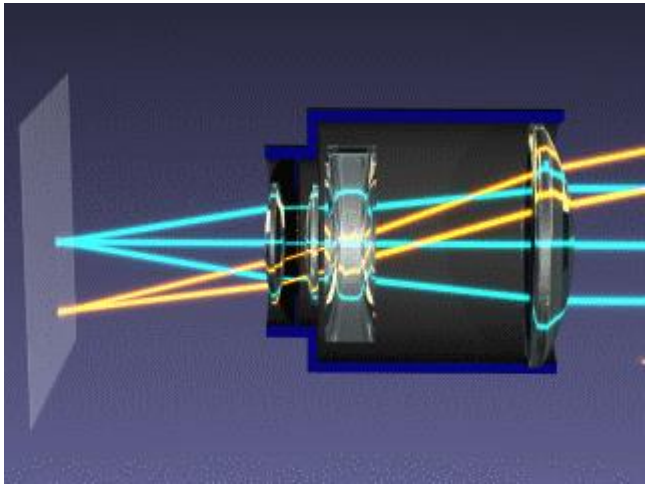
Típus	Szenzor méret
Nokia 808	1/1.2''
Nokia 1020	1/1.5''
Nokia N8	1/1.83''
Sony Xperia Z1	1/2.3''
HTC One	1/3''
iPhone 5	1/3.2''

Objektív

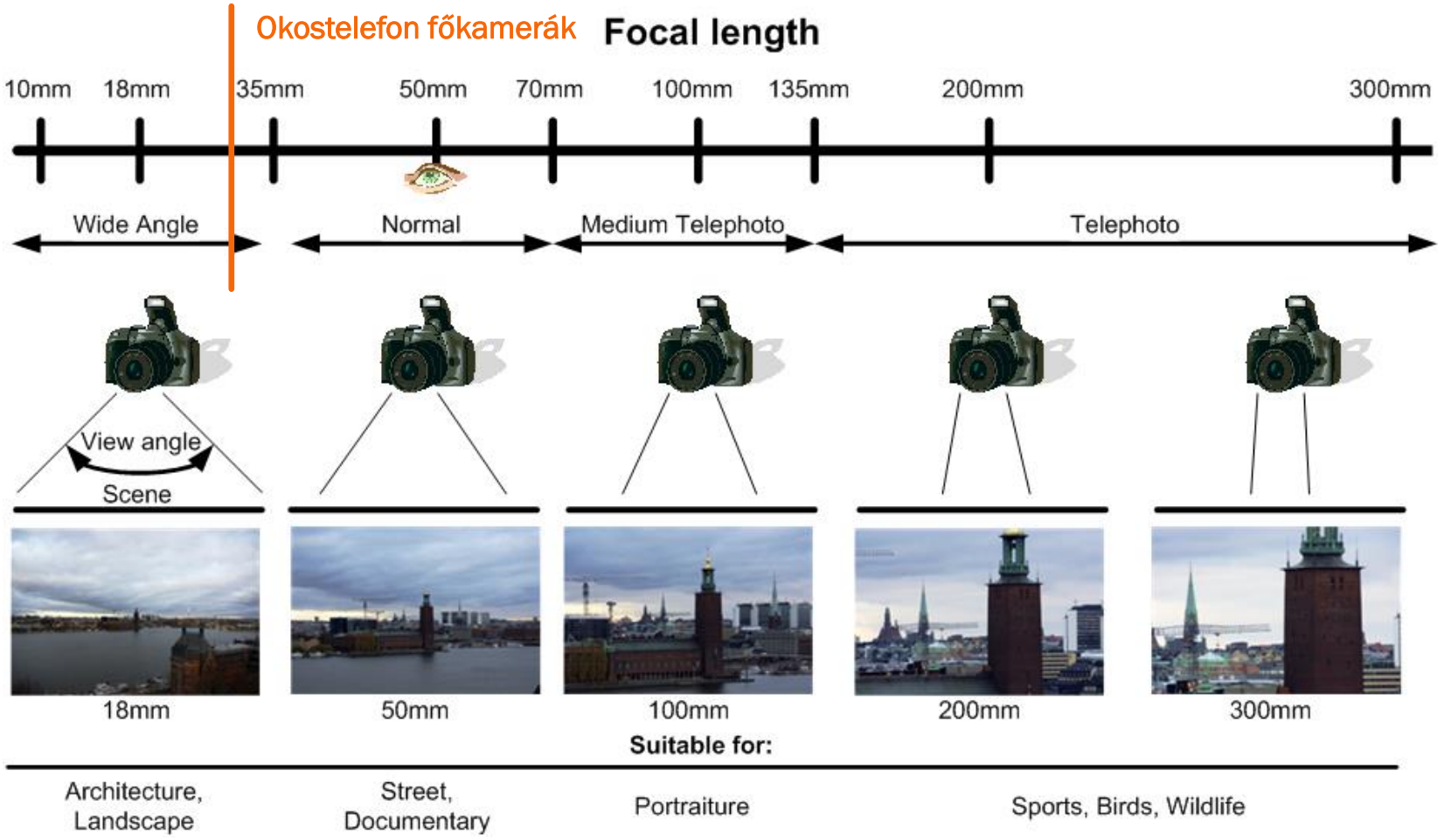
- **Fényerő**
  - Adott megvilágítás mellett átjutó fény mennyisége
  - Rekeszértékből és zársebességből számítható

- **Fókusztávolság**
  - Látószög, legkisebb tárgytávolság
  - Rögzített
    - Mobilok vagy „pancake” objektívek (rendszerkompakt, DSLR)
    - Egyszerűbb felépítés → kisebb fizikai méret
  - Zoom (állítható): motoros vagy kézi
    - 3x-5x: kompakt gépek, rendszerkompakt és DSLR objektívek
    - 5x-40x: ultrazoom (kompakt) gépek
    - **Nem csak a szorzó, hanem a befogott tartomány is érdekes!**

Forrás: Camera Lenses Research Project



Objektív



Forrás: http://www.digital-photography-student.com/lens-focal-length-explained/

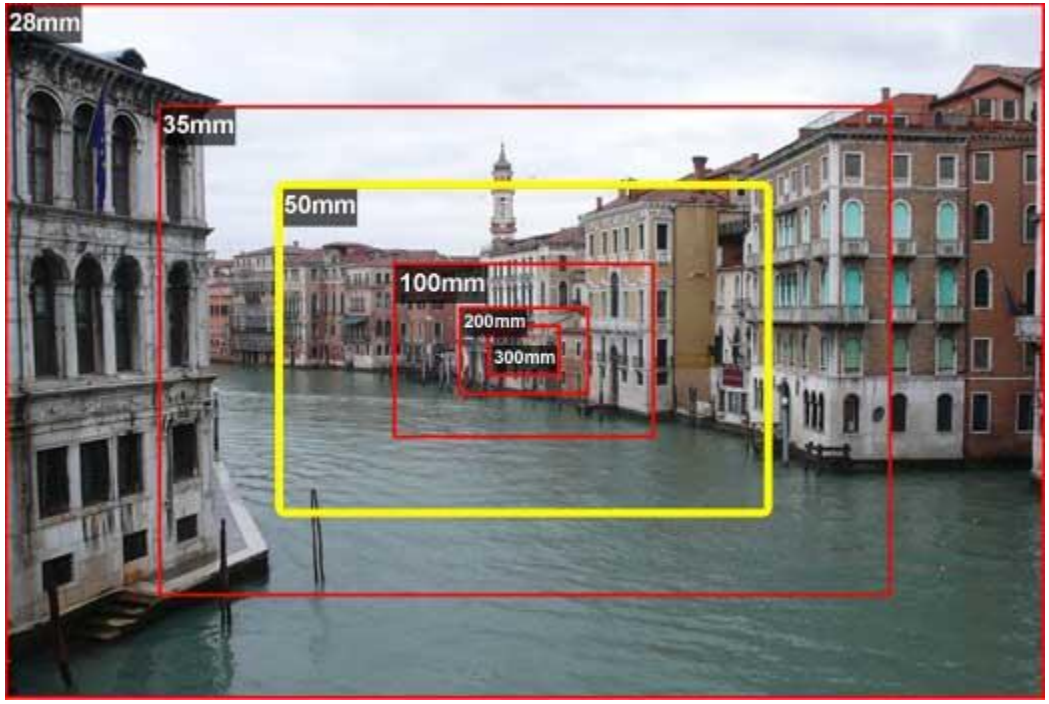
Objektív

- Normál objektív
  - Emberi szemnek megfelelő
  - 35 mm-es érzékelő esetén 50 mm fókusztávolság
- Teleobjektív
  - Fókusztávolság > 50 mm
  - Perspektívát csökkentik
  - „Zoom”
- Széleslátószög
  - Fókusztávolság < 50 mm
  - Kissé távolabb viszi a tárgyakat
  - Perspektívát növel
  - Mai olcsóbb objektívek jellemzően 28 mm-ről (ekvivalens) indulnak
- Halszem optikák
  - ~180 fokos látószög, 10 mm körüli fókusztávolság, erős geometriai torzulás
- Makró optikák
  - Tárgy ~1-1 méretű leképezése az érzékelőre

Nikkor 35mm f/2 AF-D objektív
(*Forrás: Mark Sweep, WikiMedia Commons*)

Ekvivalens (ekv.) érték: a könnyebb összehasonlíthatóság végett az objektív paraméter-értékeit a 35 mm-es, klasszikus filmes méretekre skálázva adják meg a gyártók!

Objektív



Forrás: http://www.wexphotographic.com/?/guides/lens-sense.html

Objektív

- iPhone 6 lencserendszere

Objektív

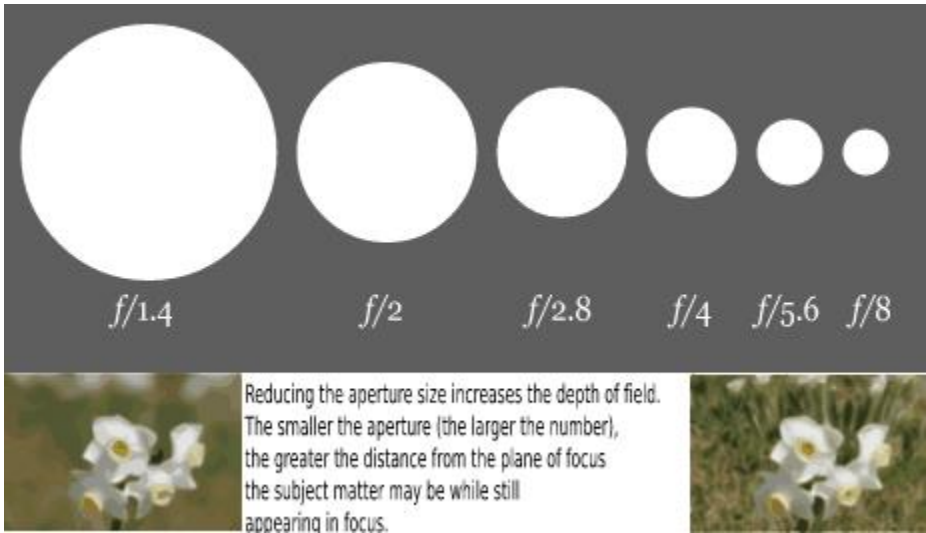
- Canon EF-1200mm F5.6
  - Magyar nyelvű leírás
  - YouTube

Mobil objektívek

- Jellemzően fix fókusztávolságú kamerák
- Manapság (2018-tól) inkább több kamerát használnak
  - Más fókusztávolsággal
  - Ezek eredményeit kombinálják
  - Vagy a zoom ezek között vált
- Periszkópos zoom
  - 2019: Oppo, Huawei P30 Pro
  - 5x, 10x (a főkamerához képest)
  - Hibrid zoom (digitálissal kombinálva)
  - Videó: erre képes a Samsung Galaxy S20 Ultra 100x-os zoomja
- Digitális zoom
  - Nem igazi zoom!
  - Kivág egy képrészletet és felnagyítja!
  - Inkább utófeldolgozással dolgozzunk helyette!

Rekesz

- Lencserendszer elemei között helyezkedik el
- Áteresztett fény mennyiségének és a mélységélesség szabályozására
  - Nagyobb rekesz, több fény, rövidebb expozíciós idő
  - Objektív sebessége: átengedett fény maximális mennyisége
- Profi gépeken állítható, olcsóbbakon automatikus vagy rögzített
- Szabványos értékek:
  - F-szám
  - Az átengedett fény mennyisége feleződik
  - Fókusztávolsághoz viszonyítva adják meg



Kép forrása: WikiMedia Commons

Rekesz

- Objektív „sebessége”
  - F-szám értelmezhető így is
  - Gyors objektív: nagy nyílás esetén elég, ha kevés ideig van nyitva a rekesz
  - Különösen gyenge fényviszonyok között fontos
    - Esti, beltéri fotók
  - Optikai stabilizálással akár 1 FÉ nyerhető

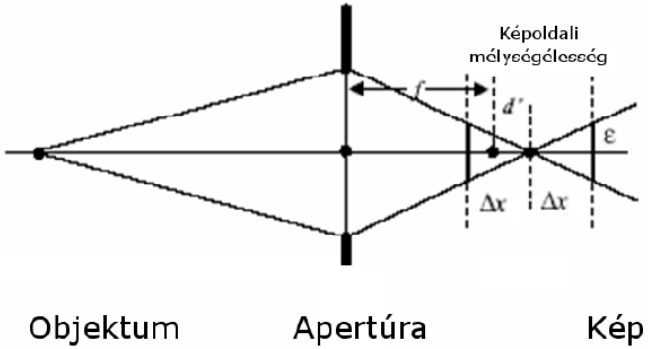
- Mobil kamerák
  - F-2 körüli értékek
    - Ez elég jó, de mégis gyengék a fényhiányos fotók!
  - Rögzített rekeszérték!
    - A Samsung és a Honor használ dual apertúrát (kétféle rekeszértékből választhatunk)
  - Optikai stabilizálás megjelenése bizonyos kamera modulokban

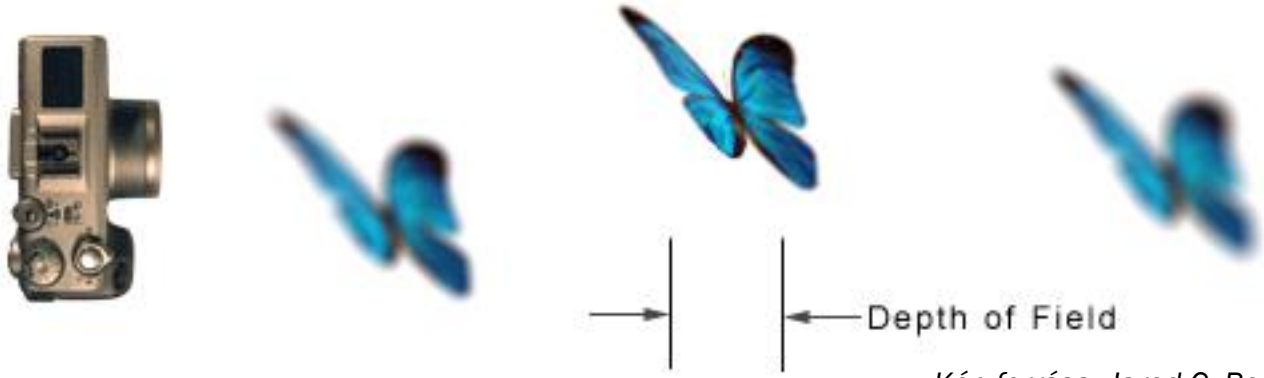
Xiaomi Note 10 kamerák

Részletes leírás: https://pixinfo.com/cikkek/kiprobaltuk-xiaomi-mi-note-10/

Mélységélesség

- A fókusztávolságban lévő tárgyak jelennek meg élesen a képen, a többi homályos: lényegkiemelés!
- Egy tartományon belül a homályosodás mértéke elfogadható/nem látható
- Mélységélesség: mekkora ez a tartomány
  - Alacsony mélységélesség: szűk tartomány
    - Pl. portréfotók esetén (homályos a háttér)
  - Nagy mélységélesség: széles tartomány
    - Tájképek (éles tartomány vége a végtelenben)
- Beállítása
  - Nagyobb rekesznyílás ⟶ kisebb mélységélesség
  - Nagyobb szenzorméret ⟶ jobb alacsonyabb mélységélesség képesség
  - Lyukkamera: mélységélesség „végtelen”

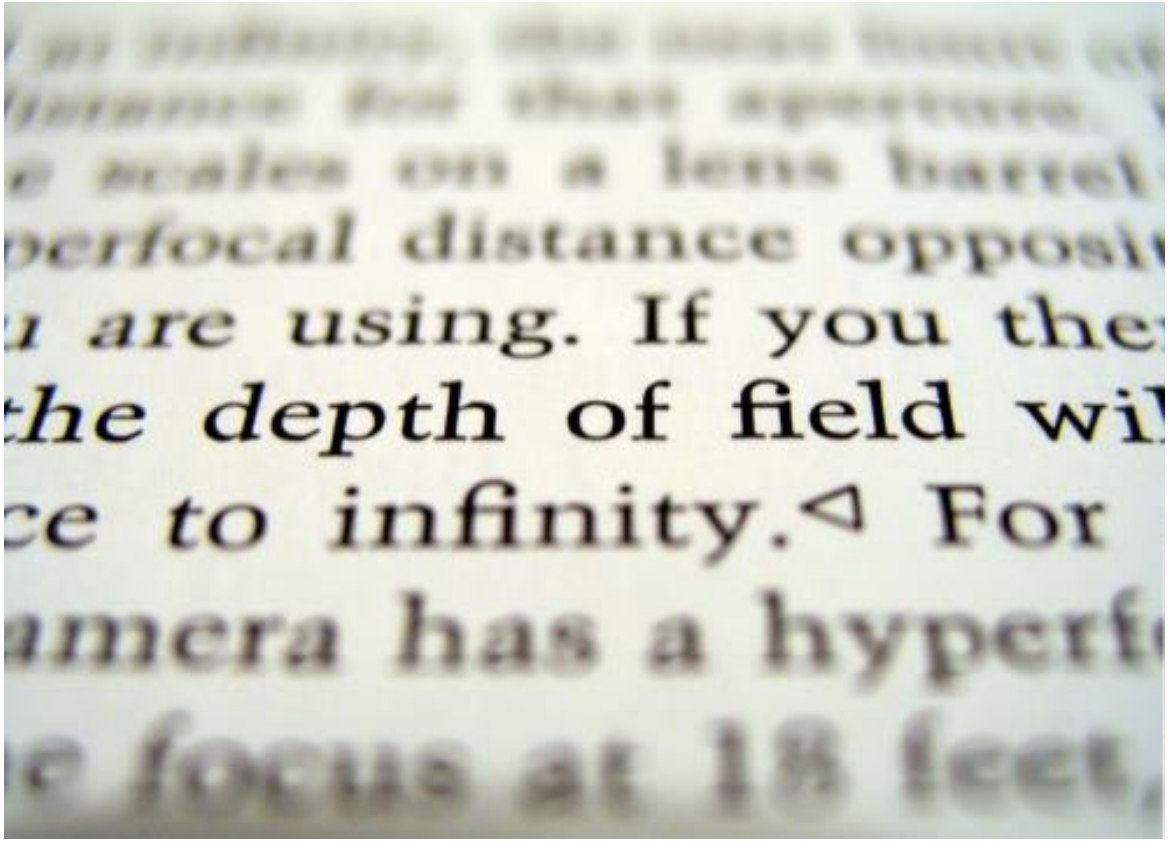




Kép forrása: Jared C. Benedict, WikiMedia Commons

Mélységélesség

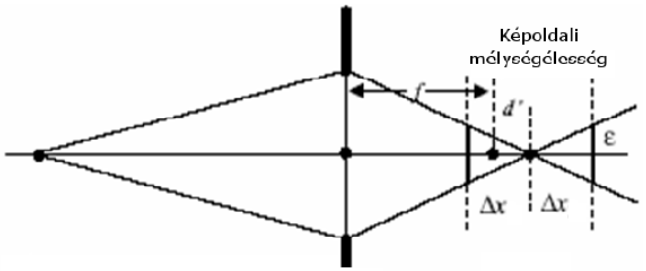
- Nagyon alacsony mélységélesség (makró fotó)



Kép forrása: WikiMedia Commons

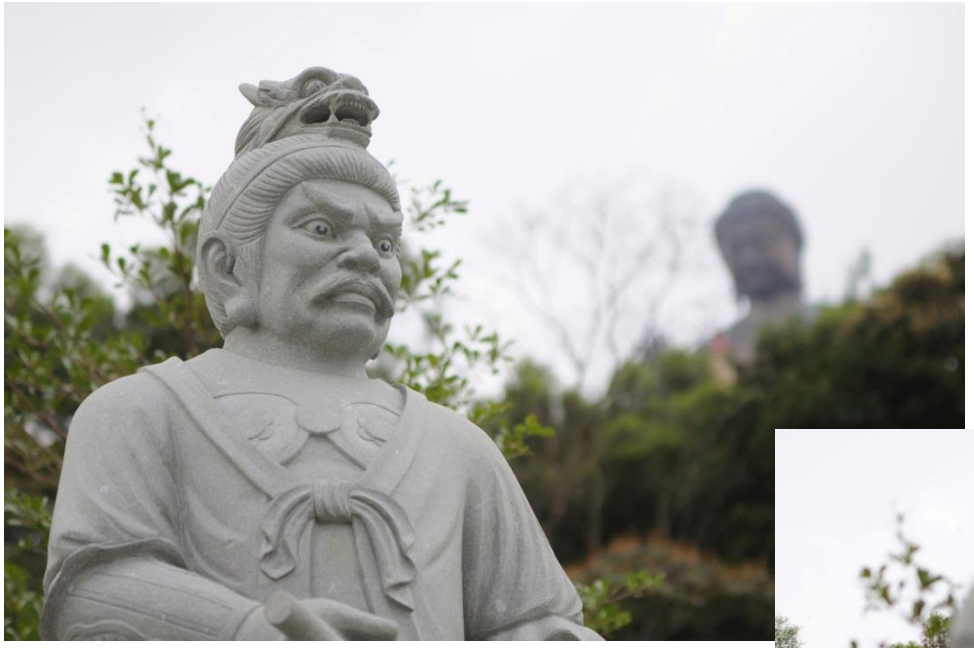
Élesség távolsága

- Milyen távolságban legyen éles a kép?
  - Különösen alacsony mélységélességnél kritikus
- Objektív és érzékelő távolságának állítása („kihúzás”)
  - Profi gépeken kézzel (is) állítható
  - Olcsóbb gépeken automatikus
    - Időigényes! (1-2 mp is lehet)
  - Fix élesség
    - Pl. Nokia EDOF kamerái: gyorsak, de közeli tárgyak nem lesznek élesek (<0.5 méter), azon túl minden éles
      - Extended Depth of Field
  - Passzív: élőkép információ alapján
    - Adott pontokban kontraszt, fáziskülönbség
  - Aktív: infra segédfény vagy lézer segítheti
    - LG G3, Asus ZenFone Zoom



PixInfo cikk

Élesség beállítása



Alacsony mélységélesség
Közeli és távoli célpont

Forrás: Szlávik Zoltán

Mélységélesség, képélesség

- Mobil eszközök
  - Kis szenzorméret, kis méretű nyílás
  - Nagy mélységélesség: a színtér nagy része éles lesz
  - Nehezebb kiemelni a lényeget
    - A laikus felhasználók körében népszerűbb, ha nincs háttérelmosás…

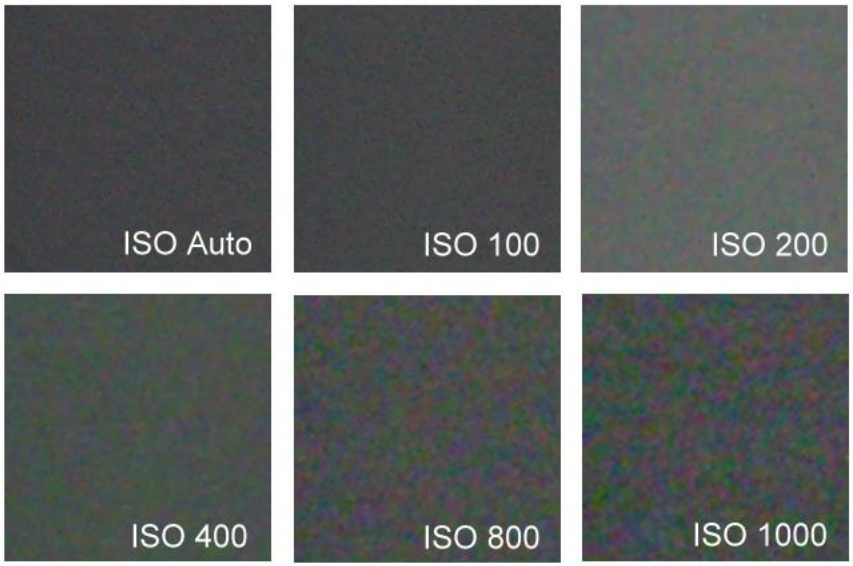
- Tükörreflexes gépek (DSLR)
  - Nagy szenzorméret
  - Nagy apertúra lehetőség
  - Alacsony mélységélesség
  - Jól kiemelhető a lényeg, viszont nehéz lehet eltalálni az élesség távolságát!
    - Jó élességállító rendszer kell!
    - Kezdő fotósok sok életlen képet készítenek a profi eszközökkel!

Expozíciós idő (Zársebesség)

- **Mennyi ideig legyen nyitva a rekesz**
  - A bejutó fény mennyiségének szabályozása
  - Rekeszértékkel együttesen
- **Alulexponált, túlexponált képek**
  - Túl kevés/sok fény
- **Expozíció korrekció (EV; exposure value)**
  - Automatikus beállítások hangolására
    - ±2 fényérték között, pl. 1, ½, vagy ¼ lépésközzel
  - Akár automatikusan több kép készítése más beállításokkal ⟶ HDR képalkotás

„Film” érzékenysége

- ISO értékek
  - A „filmes” világból
  - 50-16000 között
  - Nagyobb érték, érzékenyebb érzékelő beállítás
    - *Kevesebb fény elegendő* (rövidebb záridő és/vagy kisebb rekeszérték),
    - így *kisebb* lesz a *bemozdulás*, de
    - *zajosabb* lesz a kép.

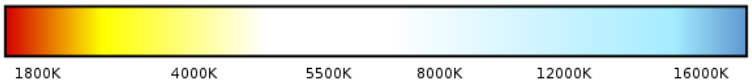


Forrás: http://www.hometheaterhifi.com/volume_13_4/samsung-nv-10-camera-10-2006-part-4.html

Színhőmérséklet

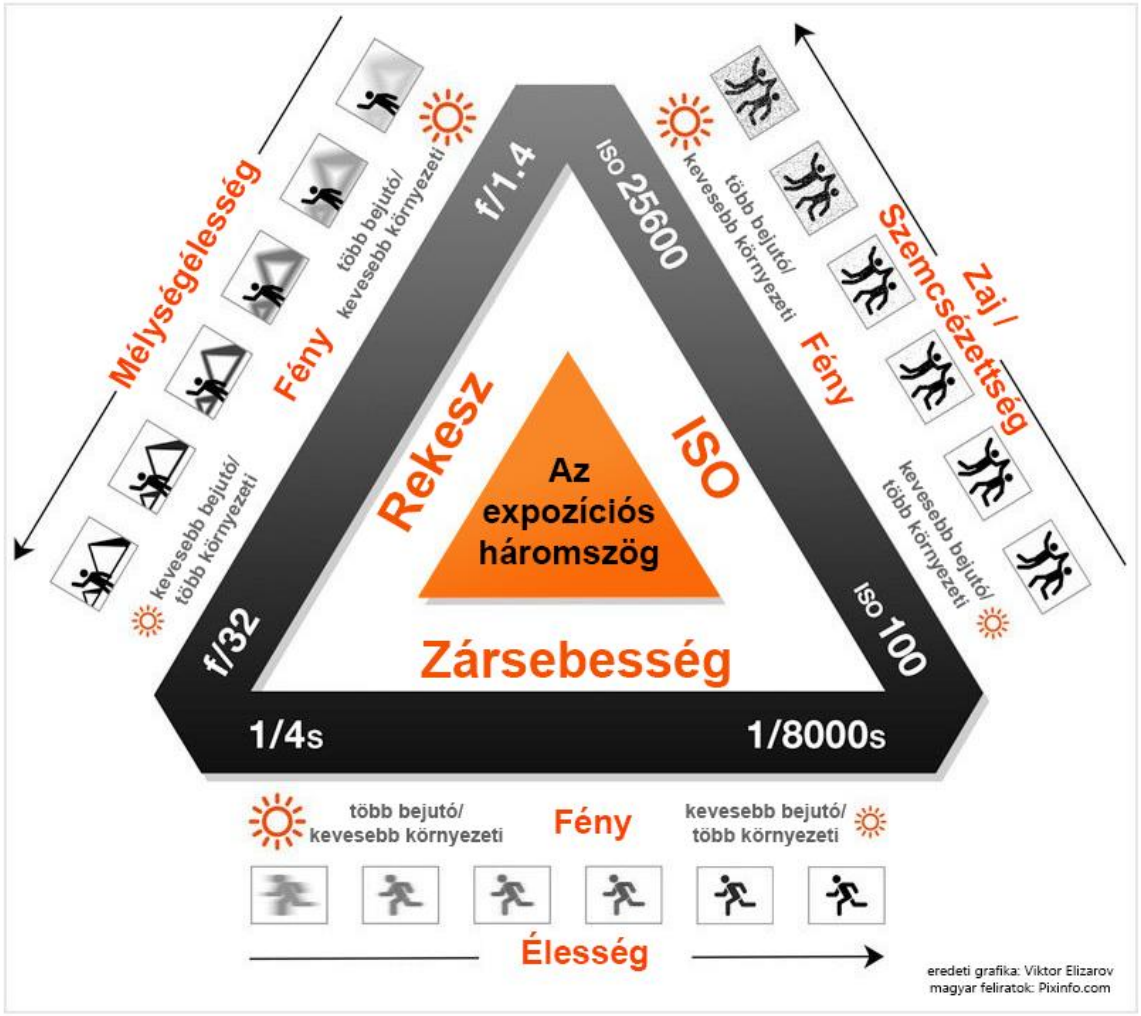
- A fényforrás színösszetételére jellemző érték
  - Kelvinben adják meg
  - Magasabb hőmérséklet: kék komponens súlya nő
  - Alacsonyabb hőmérséklet: a vörös súlya nő
  - *Az agyunk elvégzi a korrekciót!*

- Fehéregyensúly állítás
  - Kamera alapbeállítások között
  - Vagy középszürke lap segítségével



Gyertya	Kb. 1900 K
Háztartási izzólámpa	Kb. 2800 K
Fotóizzó	3200 K
Alacsony napállás (pl. reggel, késő délután)	Kb. 4800 K
Átlagos napfény, vaku	5600 K
Napos idő, árnyékban	Kb. 6000 K
Nappal, kissé felhős égbolt	Kb. 8000 K
Borult, ködös idő	Kb. 10000 K

Expozíciós háromszög



Forrás: https://pixinfo.com/cikkek/az-expozicios-haromszog/

Képek tárolása

- Formátumok
  - RAW (veszteségmentes, profi eszközökön)
    - „Előhívatlan”, nyers adat, Bayer-szűrt mérések
    - Geometriai korrekció nélkül (lencsetorzítás látható)
    - Expozíció paraméterei állíthatók (színhőmérséklet, világosság, …)
    - Mobil eszközökben jellemzően nem támogatott
      - 1-2 készülékben elérhető
      - Android platformon 5.0 felett opcionálisan megvalósíthatják a gyártók
  - JPEG (veszteséges tömörítés)
    - EXIF: képkészítés adatai
  - HEIC/HEIF (veszteséges)
    - Apple
- Mozgókép
  - 3GP, MOV, AVI, …

Kép: Yu-Lin Chan (http://oldlenses.blogspot.hu/)

Mobil kamerák jellemzői

Elegendően nagy felbontás (MP), de

- Olcsó, kis méretű érzékelők
  - Alacsony energiaigény, alacsony ár a fontos
- Gyenge objektív
  - Alacsony fényerő, hosszabb expozíciós idő
    - Bemozdulásra hajlamos képalkotás
  - Gyenge vaku, vagy annak teljes hiánya
    - Rossz fényviszonyok között nem igazán használható
  - Miniatűr méret
    - Geometriai torzulásra hajlamosság
- Fotózási paraméterek korlátozott beállítási lehetőségei
  - Sok automatikus beállítás, lassú!
- Alkalmazásaik
  - Jó fényviszonyok között olcsóbb P&S (point-and-shoot) kamerák kiváltására
  - Videók felvétele viszont meglepően jó minőségű

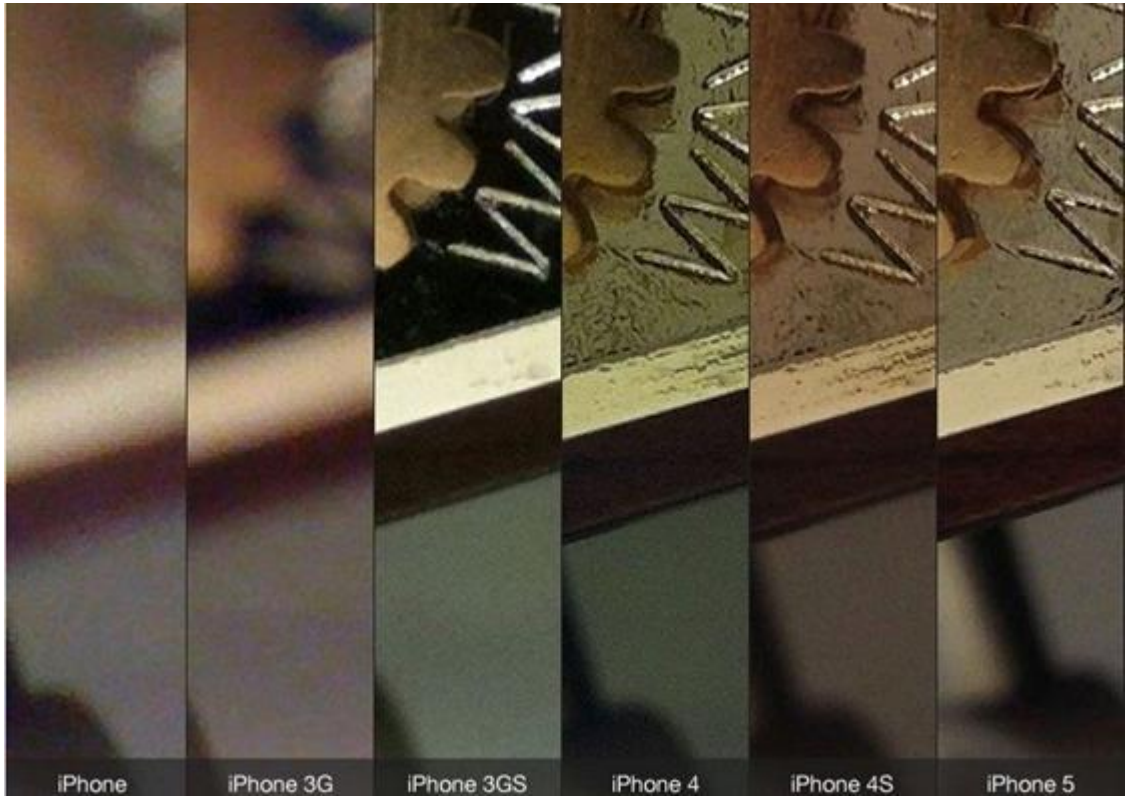
Szorgalmi feladat

- **Mobil képalkotáshoz használt szenzortípusok**
  - BSI, ISOCELL, BRITECELL, Ultrapixel, …
  - PowerPoint bemutató
  - 15-20 dia
  - Címlap + szerző neve
  - Felhasznált képek forrása (jegyzetként vagy a dián)
  - 4 pont szerezhető

Mobil kamerák fejlődése

Forrás: http://www.cameratechnica.com/2011/10/18/iphone-4s-camera-review/

Mobil kamerák fejlődése

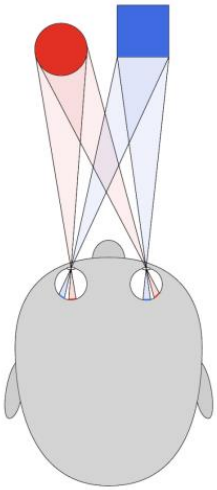


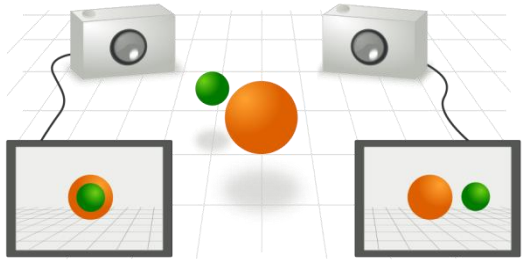
Mobil kamera képességek

- 2020. elején
  - Fotó
    - Hátlapi főkamera: átlagosan 12 megapixel feletti (MP); egyébként 8-108 MP között; magas MP szám esetén jellemzően kisebb MP kimenet (átlagolás)
    - Előlapi (videótelefonálás, „szelfi”): 2-13-20 MP között
    - 2-3-5 lencsés képalkotás
      - Mesterséges zoom-olás (eltérő fókusztávolság)
      - Fényerő növelés (színes és szűrő nélküli szenzorok)
      - Utólagos fókuszállítás (eltérő apertúra)
    - 3D sztereó képalkotás elérhető (gyakorlatilag eltűnt)
    - ToF (Time-of-Flight) kamera modul: távolságmérés, arc-alapú ellenőrzés
  - Videó
    - Hátlapi: jellemzően 1080 HD; 4K, 8K felbontás megjelent
    - Gyorsított és lassított felvételi képesség
    - 3D sztereó képalkotás (gyakorlatilag eltűnt)

Sztereo képalkotás és 3D megjelenítés

- A térbeli látás alapja
  - A két szem két különböző képet lát, ami alapján az agyban kialakul a térérzet.
  - Az objektum vetített képének helye függ a távolságtól
- Térbeli látásunk
  - évezredes evolúciós fejlődés és
  - gyermekkori tapasztalat útján alakult ki.
- Sztereo látás
  - A sztereo látás az emberi látást utánozza.
  - Két kamera által készített képből térbeli információk nyerhetők ki.
  - Ezt a módszert számos technika alkalmazza (például robot látás, 3D rekonstrukció képekből).





Kép: Arne Nordmann

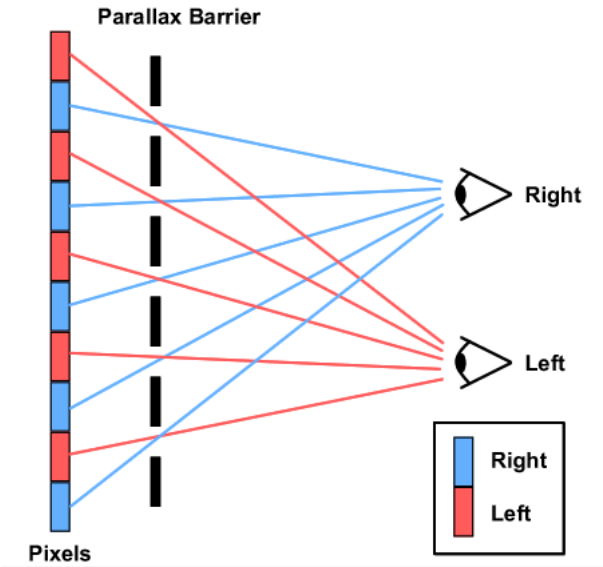
Sztereo képalkotás és 3D megjelenítés

- **Kép készítése**
  - Rögzített helyzetű, egyező paraméterű kamerapárral ugyanazon időben
  - Egy készülékkel, különböző nézőpontokból
    - Nem szerencsés

- **Kép megjelenítése**
  - Anaglif technika (színszűrős szemüveg)
  - Polarizációs technika (mozifilmek)
  - Aktív záras szemüveg (3D TV, 3D Vision Kit PC-n)
  - Autosztereoszkóp kijelző
    - Rögzített pozícióból látható, de mobil eszközöknél ez természetes

Autosztereoszkóp eszközök

- Jellemzők
  - Szemüveg nélküli 3D élmény
  - 1 rögzített pozícióból nézve működik
- Működési elve
  - Két kamerás (sztereo) képalkotás
  - „Parallax korlát” megjelenítés
- Felhasználás
  - Játékkonzolokban, mobilokban
  - 3D fotók, videók készítése
  - 3D játékok
  - Távolságbecslés



Készülékek

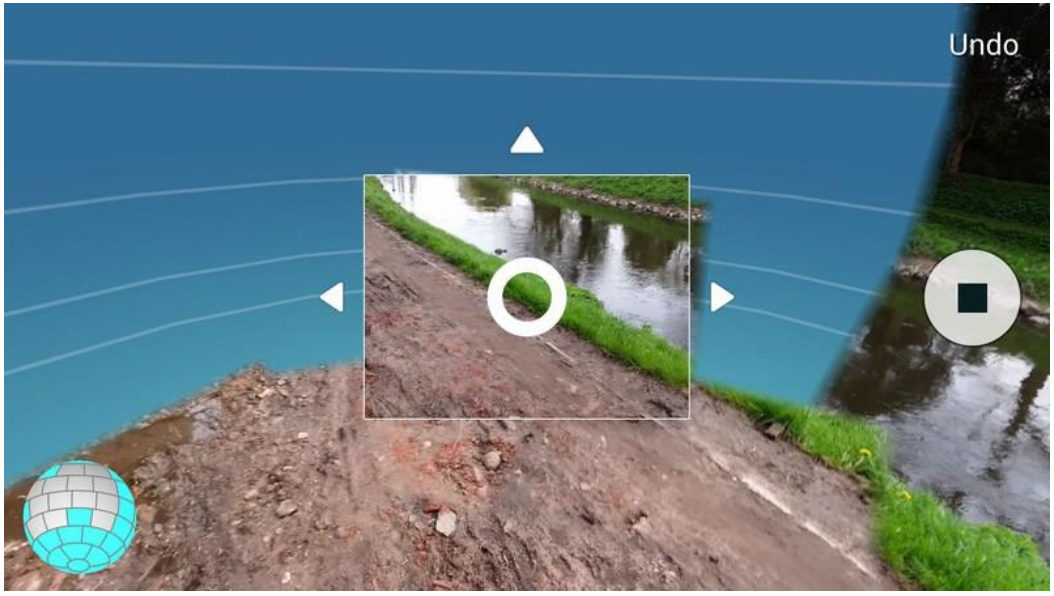
- Kamerák
  - Fuji FinePix Real 3D W1 (sztereo kamerapár)
- Mobil eszközök
  - Nintendo 3DS játékkonzol
    - Autosztereoszkóp kijelző
  - HTC EVO 3D (2011), LG Optimus 3D (2011) és LG Optimus 3D Max (2012) (sztereo kamerapár + autosztereoszkóp kijelző)
  - Nvidia fejlesztői referenciaeszközei (sztereo kamerapár + autosztereoszkóp kijelző)
- Fogadtatás
  - Nem lett sikeres irányvonal

Szoftveres képalkotás

- Panorámakép készítés
  - Csík és gömb (StreetView-szerű) képek
- HDR (High Dynamic Range) fotók
  - Egyenetlen megvilágítottság kezelése
- Mesterséges mélységélesség (”lens blur”)
  - Fontos objektumok kiemelhetősége
  - Újrafókuszálás: interaktívan kijelölt képterület
- Mesterséges intelligencia alkalmazása
  - Fotózási paraméterek automatikus becslése színtér elemzés alapján
- Gyenge fényviszonyok kezelése
  - Asztrofotózás

Gömbpanoráma

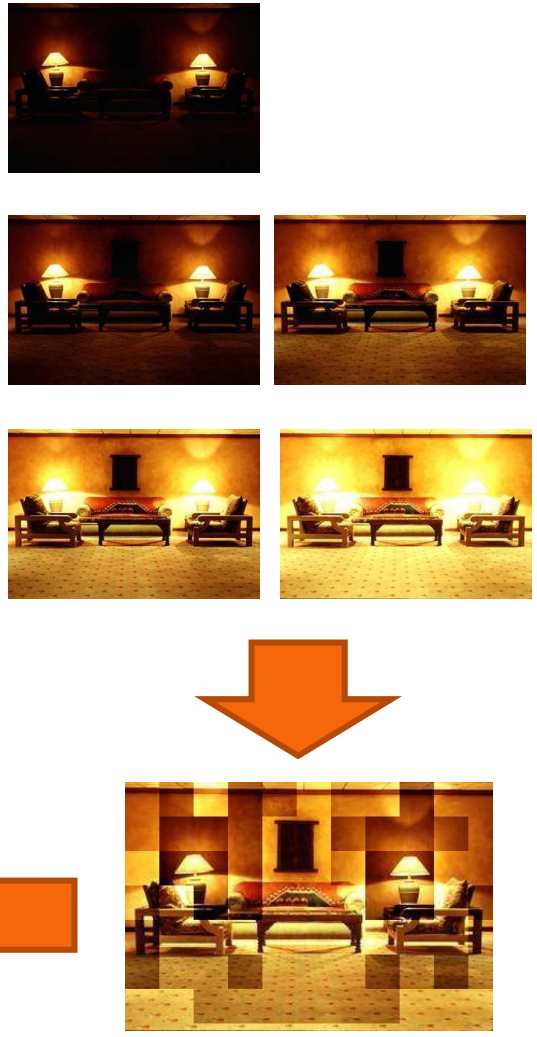
- Google Camera
  - Képkészítés szenzoros segítséggel (giroszkóp)
  - Átfedő fotók utólagos elegyítése
  - Probléma: időigényes felvételsorozat (akár 1-2 perc), így mozgó objektumok megjelenése torz lehet



Source: downloadol.com

Megvilágítás egyenetlensége

- HDR (High Dynamic Range) módszer
  - Összetett fényviszonyok között (sötét és fényes régiók vegyesen a színtérben) alul- és túlexponáltság jelenik meg a képen
  - Szoftveres megoldás
    - Képsorozat készítése többféle expozíciós értékkel (pl. ±2 EV között 5 kép)
    - Rögzített pozícióból, mozgás nélküli színtérről
    - Képek elegyítése: minden képből a legtöbb információt hordozó rész választása
      - Képek kis részekre osztása
      - Információ mérése: entrópia
      - Elegyítő függvény: a részek határai ne látszanak



Kép: Osztroluczki András és Zajác Erik

Mobil kamerák jellemzői

- Mesterséges mélységélesség-állítás („*lens blur*”)
  - Utólagosan szoftveresen
  - Kameramozgás segítségével képháttér elmosása
    - Képpontokhoz tartozó objektumpontok távolságának becslésével
    - Pl. Google Camera alkalmazás (Android)
  - Kapcsolódó cikkek
    - Előbb fotózz, majd utána döntsd el, mi legyen éles!
    - Blurring the background: Understanding focus, defocus, refocus and Ufocus
    - Battle of the Blur: Defocus camera showdown
    - Google's new camera app brings Photo Sphere and Lens Blur to Android devices
    - Google Camera: What's new, and a guide to the latest features
    - New SDK lets developers take advantage of HTC One M8's dual cameras

Mesterséges mélységélesség

- ”Lens blur”
  - Széles mélységélesség: (majdnem) minden éles a képen
    - „Dokumentálás”
  - Szűk mélységélesség : csak egy adott távolságtartományban lévő objektumok élesek, a többiek elmosódottak
    - A fotós kiemelheti, mi fontos számára a színtérben
    - **Fizikai korlátok miatt nehezebben valósítható meg mobilokon**

Photo by Sascha Haeberling

Photo without Lens Blur Photo with Lens Blur

Forrás: Google Research BlogSpot

Mesterséges mélységélesség

- **Szoftveres megoldás 1 kamerával (Google Camera app)**
  - Kamera mozgás expozíció közben (rövid videó készítése)
    - Lásd YouTube videó
  - Objektum pontok távolságbecslése (közeliekre pár használható)
  - Távolságtérkép alapú újrafókuszálás

Forrás: Google Research BlogSpot

Photo by Colby Brown

One of several input photos | Depth map (black close, white far) | Photo with Lens Blur

Lens Blur – Újrafókuszálás

Photo by Rachel Been

Lens Blur selfie, background focus

Lens Blur selfie, foreground focus

Forrás: Google Research BlogSpot

Asztrofotózás

- Cikkek
  - https://www.androidcentral.com/google-details-astrophotography-pixel-4
  - https://www.androidcentral.com/how-shoot-astrophotography-pixel-4

Hardveres fotó megoldások

- Háttér-elmosás
  - Dual pixel szenzor technológia, ToF (mélység) kamera
- Többféle kamera modul
  - Más képalkotási paraméterek
- Optikai zoom
  - Periszkóp elrendezéssel
- 360 fokos külső kamerák
  - USB-C vagy Bluetooth kapcsolattal

Hardveres háttér-elmosás

- Google reveals how to simulate shallow DOF from a single mobile camera
  - „Put simply, Google repurposes its **dual-pixel auto focus hardware** utilized increasingly in mobile cameras for fast AF. **Each pixel on the sensor is split into two photodiodes; the left- and right-looking (or up- and down-looking) photodiodes essentially establish two perspectives of the scene with a ~1mm stereo baseline.** A burst of images are aligned and averaged to reduce noise, and a stereo algorithm computes a depth map from the two perspectives. This simulates the data that would be provided by two physical cameras next to each other, enabling Google's software to determine the depth of every point within the captured scene.”

- ToF kamera (Honor View 20 review)
  - „Speaking of portraits, I can't say I found the View 20's **depth detection features to be markedly better** than its rivals, even with the much-hyped TOF sensor flanking the main camera. **The TOF sensor is supposed to improve the accuracy of the depth maps that make up portrait and aperture-effect photos, however in reality the phone produced the same mixed results I've seen from countless other devices.**”

Multi kamerák

- Két megegyező objektív
  - 3D sztereo – eltűntek a piacról
- Eltérő fókusztávolságú lencsék
  - Dual-lens smartphone cameras are coming, and this is why we want one
  - Mesterséges zoom-olási lehetőség
- Eltérő apertúra
  - Honor 6 Plus: F/2 és F/2.4 apertúrák
    - Több fotó készítése eltérő nyílásokkal
    - Újrafókuszálás és apertúra választás f/0.95 – f/16 között
- Eltérő szenzorok
  - RGB + Black&White: BW Bayer szűrés nélkül, jobb fényerő, de szürkeárnyalatos, használható a színes javítására

Multi kamerák

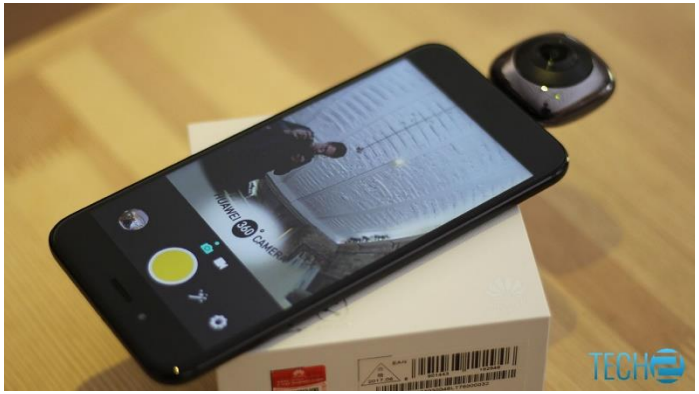
- Cikkek
  - Understanding the dual camera systems on smartphones
  - Dual camera smartphones: The history running through to the Galaxy Note 8
  - Dual Cameras for Image Fusion

Optikai zoom

- 10×-es zoomos okostelefont hoz az Oppo
  - „Noha voltak már próbálkozások optikai zoomos okostelefonok készítésére, ezek vagy túl vastagok (lásd Samsung Galaxy Zoom), vagy nem túl jó minőségűek voltak (Asus Zenfone Zoom).”
  - „A 10×-es zoom úgy jön létre, hogy egy 5×-ös optikai zoomos kamera alatt egy nagylátószögű kamera is megtalálható. A kamerák dupla stabilizátorral is rendelkeznek, és egy harmadik, tele gyújtótávolságot is kínáló modul is kerül a készülékre.”

360 fokos kamerák

- **Felépítésük**
  - Két 180 fok feletti látószögű lencse
  - Képeik gömbpanorámába elegyítve
- **Képességeik**
  - A teljes tér feltérképezése egy expozícióval történik
  - Fotó és videó rögzítése
  - Effektek
    - kisbolygó, VR, halszem, perspektív
- **Okostelefonhoz csatlakoztathatók**
  - Wifi-n
  - USB-C porton
- **Eszközök**
  - Samsung Gear 360 kamera
  - Huawei CV60



360 fokos kamerák

- Alkalmazási példák

Mobil kamera jellemzők

- Cikkek
  - GSM Arena Photo quality comparison
  - Megkerestük az Android sztárfotósát (2018. június)
  - Huawei Mate 20 Pro camera review: Triple-cam juggernaut (2019. január)
    - 40Mp main camera with 1/1.7″ sensor, f/1.8-aperture, 27mm-equivalent lens;
    - 20Mp super-wide-angle with f/2.2-aperture, 16mm-equivalent lens;
    - 8MP tele with f/2.4 aperture, OIS, 80mm-equivalent lens;
    - PDAF + laser AF, Dual-tone LED flash; 4K video at 30fps (1080p/30fps at default settings)
  - Új mobil került a csúcsra: a Xiaomi Mi 10 Pro a világ legjobb kamerás telefonja (2020. február)

Érdekességek

- Mikroszkóp Lumia módra Avagy hogyan forradalmasíthatja az orvostudományt a 1020-as
  - „A múlt hónapban egy tudományos tanulmányt tettek közzé egy orvosi napilapban, melynek a szerzője azt állítja, a Lumia 1020-as képes forradalmasítani a kórszövettant, és az orvos tudományt világszerte. Mint tudjuk, a Nokia nem kicsit pörkölt oda a konkurenciának a 41 megapixellel ellátott Lumia 1020-as bemutatásával. Ám ha a megapixelek száma nem is minden, a háromszoros veszteségmentes zoom már csakugyan elismerésre méltó. Ebben látták meg a lehetőséget Dr. Mark Li-cheng Wu és Dr. John Paul Graff, akik nem mellesleg Lumia rajongók is.”

Érdekességek

- **Próbálkozás:** Digitális kamera okostelefon operációs rendszerrel
  - Nikon Coolpix S800c
  - Samsung Galaxy Camera
  - Samsung Galaxy NX (cserélhető objektív!)
  - Panasonic Lumix CM1 (1’’ szenzor, Leica objektív)

Nem lett sikeres!

Érdekességek

- **Próbálkozás:** Kijelző nélküli objektív
  - Sony QX10 és QX100
    - Csak objektív
    - Wifi-n okostelefon kijelzőre
    - Fizikailag a mobil készülékre csatolható

Nem lett sikeres!

Képek forrása: Sony

Érdekességek

- Objektív elé csatolható kiegészítők

Inkább játék, mintsem komoly alkalmazhatóság!

Olloclip 4-az-egyben objektív
Képek forrása: Engadget

Vktech 60x mikroszkóp előtét Samsung Galaxy S4-hez
Képek forrása: Android Central

Érdekességek

- Optikai zoomos mobil
  - Asus ZenFone Zoom ZX550 (2015.)
  - 3x zoom, 10 lencsével
  - F2,7-4,8 fényerő
  - Lézeres autofókusz
  - Optikai képstabilizátor
  - Dual LED világítás
  - 13 megapixel
  - Előlapi kamera 5 megapixeles, F2,0 fényerejű fix optika
  - Full HD 30 kép/mp videó mindkét kamerával

Kép forrása: PixInfo

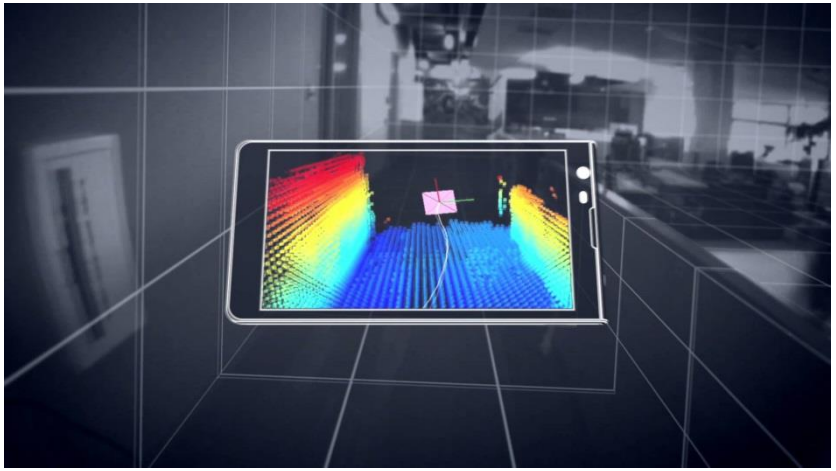
Nem lettek sikeresek!

- Panasonic CM1: nagy szenzor + Leica objektív
  - But the Lumix CM1 isn't just any smartphone: it has a **1-inch sensor**, the kind you'd find in high-end point-and-shoot cameras, plus an **f/2.8 Leica lens** that you'd be lucky to find on a high-end point-and-shoot. Along with a **20-megapixel** sensor, this phone is designed to basically reinvent the wheel when it comes to smartphone cameras. It also has a 4.7-inch, 1080p display, a Snapdragon processor, **Android** 4.4, and 2GB of RAM.

Érdekességek

- RGB-D szenzorok alkalmazása
  - Szín + távolságmérés
    - Kinect-szerű hardver elem
  - Google Project Tango: 3D tér feltérképezése
    - Kis távolság: szobák, belső terek
  - The next mobile imaging war won't be waged over megapixels
    - Though Mountain View is focused on 3D mapping, so-called **depth camera tech could dramatically improve all the pictures** you take with your smartphone.
  - Official YouTube Video

Támogatása megszűnt 2018-ban!



Gyakorlati feladatok

- Hasonlítsuk össze különféle kamerák képességeit!
  - Készülékek:
    - Mobilok: régi Windows Mobile, régi Android, újabb Android
    - Dedikált kamerák: kompakt kamera, tükör nélküli cserélhető objektíves (MILC), tükörreflexes (DSLR)
  - Vizsgálati szempontok:
    - Szenzorméret: összehasonlítva a régi filmes képkocka mérettel
    - Képkészítés különféle megvilágítási körülmények között: egyenletes természetes fény, egyenletes mesterséges fény, egyenetlen megvilágítás, fényhiányos szoba
    - Hasonlítsuk össze a képek minőségét és méretét!

Gyakorlati feladatok

- **Próbáljuk ki az újabb okostelefonok szoftveres képalkotási képességeit!**
  - Mesterséges mélységélesség (Lens Blur): Google Kamera alkalmazással, készülékek kameraszoftvere által biztosított eszközökkel (duál kamerás esetben)
  - Sztereo 3D képalkotás + autosztereoszkóp megjelenítés: HTC EVO 3D és LG Optimus 3D készülékekkel
  - HDR képalkotás: beépített kamera alkalmazás funkciójával, HDR Camera+ Android alkalmazással

Gyakorlati feladatok

- **Próbáljuk ki a gömbpanoráma képalkotást!**
  - Google Kamera alkalmazással
  - Huawei CV60 kamerával: kisbolygó, VR, halszem, perspektív megjelenítés
- **Próbáljunk ki mobil kamera elé illeszthető előtét lencséket!**
  - Nagyítás, halszem hatás
- **Próbáljuk ki a `Device Info` alkalmazást és nézzük meg, hogy API szinten milyen kamerákat érünk el!**
  - Hiába a sok kamera, saját alkalmazásban nem igen használhatjuk ki őket…

MOBIL KÉPELEMZÉS ÉS GRAFIKA

FEJLESZTÉSI LEHETŐSÉGEK MOBIL PLATFORMOKON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Mobil fejlesztési irányelvek

- Elve több ponton különbözik az asztali gépestől
  - Kevesebb memória, az alkalmazások limitált memóriát foglalhatnak (pl. 16-24 MB)
  - Kisebb kijelző
  - Más az alkalmazások életciklusa
  - Hálózati kapcsolat lassú lehet / megszakadhat
  - …
- Használhatóság
  - Az alkalmazás illeszkedjen az adott platformhoz
    - Vizuális megjelenés, működés, integráció, …
- Probléma
  - Platformok sokszínűsége → Nehézkes karbantartás!

Android

- **Multiplatform fejlesztői környezet**
  - Linux, Windows, MacOS X
  - Ingyenesen elérhető eszközök és SDK
- **Java (vagy Kotlin) programozási nyelv**
  - Könnyen tanulható
  - Több általános Java csomag is használható
  - Android specifikus kiegészítések
  - Rendelkezésre áll natív környezet is (C, C++ nyelv)
- **Tesztelés**
  - Készüléken vagy emulátorban
  - Az elkészült csomagok szabadon terjeszthetők, tetszőleges készülékre telepíthetők
  - **Fragmentáció**: sokféle készüléken (HW/SW) kell fusson!

Android

- **Android Studio**
  - IntelliJ-alapú IDE, folyamatosan bővülő lehetőségek
  - Újabb rendszer, a Google erre állt át
- **Eclipse**
  - ADT plugin; Google már nem fejleszti
  - Rengeteg információ áll rendelkezésre róla, de elavult
- **App Inventor**
  - Vizuális programozási környezet
    - Egyszerűbb alkalmazások számára
    - Akár programozási ismeret nélkül
  - A Google már nem támogatja, nyílt forráskódú projektté alakul → MIT veszi át
    - http://appinventor.mit.edu/
  - Csak állandó internetkapcsolat mellett működik

iOS (iPhone, iPad)

- Fejlesztői környezet
  - Intel processzoros MacOS X rendszerű gép kell!
  - Ingyenesen elérhető eszközök és SDK (XCode)
- Objective-C programozási nyelv
  - Nehezebben tanulható
- Swift nyelv
  - 2014-ben bejelentett új nyelv iOS és OS X programozáshoz
- Tesztelés
  - Regisztrált készüléken vagy emulátorban
  - Készüléken való teszteléshez licenszdíjat kell fizetni!
    - Évi 99 USD! (Személyes licensz.)
  - Egyetemeken rendelkezésre áll ingyenes lehetőség is
  - Zárt eszközpark: néhány készüléken kell csak tesztelni

iOS (iPhone, iPad)

- iOS fejlesztői licenszcsomagok
  - *iOS Developer Program*
    - $99 évente
    - 1 készüléken tesztelhető
    - App Store-ba feltölthető (csak így osztható meg másokkal)
  - *iOS Developer Enterprise Program*
    - $299 évente
    - "Saját" céges programok készítése (akár App Store nélkül)
  - *iOS Developer University Program*
    - Egyetemi oktatásban (nálunk is elérhető)
    - Készüléken tesztelhető
    - App Store-ba viszont nem tölthető fel

iOS anyagok

- Honlap
  - http://developer.apple.com
- Tananyag
  - Rengetek könyv és webes anyag
  - Pl. Stanford egyetem kurzusa
    - http://www.stanford.edu/class/cs193p/cgi-bin/drupal/

Windows Phone

- Fejlesztői környezet
  - Visual Studio-ra épül
  - Van ingyenes változat is
- C# programozási nyelv
  - Silverlight alapú: RIA alkalmazások, WP vezérlők
  - XNA alapú: Játékok, nagy teljesítményigényű 2D/3D grafika
- Tesztelés
  - Készüléken vagy emulátorban
  - Egyelőre kevéssé elterjedt platform, kérdés, mennyire „fut be”.
  - Többféle gyártó, de a Microsoft erősen ellenőrzi a hardver elemeket és nem engedi a rendszer testre szabását a fragmentáció elkerülése érdekében

Elavult!

Keresztplatform fejlesztés

- Előny
  - Alkalmazások könnyen átvihetők más rendszerekre
- Hátrány
  - Specifikus hardver API-k elérése nehézkes lehet
  - Esetenként nem natív kezelhetőség
- Eszközök
  - J2ME, PhoneGap, Adobe AIR, Unity, Xamarin, Crosslight, Ionic
  - Mobilra optimalizált webes alkalmazások (HTML5)
  - ...

Keresztplatform fejlesztések

- Adobe
  - Flash
    - Apple nem engedte iOS-re hatékonysági okok miatt
    - Android egy ideig támogatta (2012. augusztusáig)
      - Adobe ekkor visszavonta a Flash for Android-ot
    - Problémák az asztali géptől eltérő használat miatt
      - Interakció más
  - AIR
    - Flash köré önálló (nem böngésző alapú) környezet
    - iOS és Android is támogatja

Keresztplatform fejlesztések

- PhoneGap
  - Hibrid felépítés
    - JavaScript, HTML5, CSS
    - Platformfüggő részek natívak lehetnek (plugin)
    - Megjelenés webes nézetben
  - Platformok
    - iOS, Android, webOS, Symbian, Blackberry, Windows Phone, Windows 8
    - „Cloud compiler”
  - Adobe felvásárolta a Nitobi céget
    - PhoneGap → nyílt forráskód (Apache Callback → Apache Cordova)

Keresztplatform fejlesztések

- Xamarin
  - .NET objektumkönyvtárak használatával
  - Introduction to Xamarin.Android
- Crosslight
  - Build Cross-platform Mobile Apps Easily with Crosslight
- Unity
  - 2D, 3D grafikus motor
    - Látványos grafika + hang + játékmenet készítés
  - Exportálási lehetőség számos platformra
    - PC, konzol, web, mobil, ...

Keresztplatform fejlesztések

- Orosz Gábor Facebook bejegyzése

„A következő platformok játszanak, prioritás szerint:
1. iOS
2. Mobile web
3. Android (Google)
4. Android (non-Google)
5 WP

Néhány kérdés, amit érdemes feltenni magadnak:
- Mennyi custom vezérlő lesz az app-ban?
- View vagy logika nehéz az app?
- Mennyire fontos szempont a platform konzisztens design és viselkedés?
- Milyen komoly teljesítményre van szükség? Akár már egy nagy lista (1000+), sok adattal is necces lehet.
- Mennyire szeretnél alacsony szintű szolgáltatásokhoz nyúlni? Esetleg Hw közeli?

Tapasztalatok:
- Natív: mindent meg lehet csinálni, per platform így a legkönnyebb megoldani a dolgokat, de magas a költség
- Xamarin: Akkor jó, ha logika nehéz az app, ellenben HA nagy a view, sok a custom control, akkor hamar elveszhet az előny, akkor mennék erre, sok a szívás a wrapperekkel, főleg ha nem csinálta meg valaki, csomó natív okosságról tuti le kell mondani, ha a WP tényleg nagyon fontos, és a mobile web mehet külön, ez is költség
- PhoneGap: egyre jobb, ha van egy jó mobile web megoldásod, akkor onnan lehet közelíteni, SOK a limitáció, garantált lemondásokkal jár, de meggyőzött, hogy működik, ha sok az adat, nagy a lista, nagy a DOM, sok az animáció, akkor lesz vele szívás

Minden cross-platform cuccra igaz, hogy a platform konzisztens megoldások azért vérzenek, vagy sokkal több plusz munkába kerülnek, mint az elsőre gondolnád. Tuti le fog menni 2-3 projekt, amíg összeáll a fejedben, hogy mit lehet és mit nem.”

Mobil HTML5

- **Egyre bővebb hardver elérés weboldalakról!**
  - Pozíció (Geolocation)
  - Telefon helyzete (Device Orientation)
    - Giroszkóp, gyorsulásmérő, iránytű
  - Kamera és mikrofon (User Media)
    - **A WebRTC támogatásával, Javascript és HTML5 használatával tudjuk elérni az élő kameraképet. Valós idejű kommunikációra is képes az API.**
  - Akkumulátor állapota (Battery)
  - Rezgés (Vibration)
  - Környezeti világosság (Ambient Light)
  - Közelség (Proximity)
  - **Biztonsági kockázat!**
    - A böngésző engedélyt kér az egyes elemek használatára, de nem tudjuk követni, hogy mit tesz az adatokkal!

Mobil HTML5

- Jövő
  - Nem csak szenzorokhoz, hanem belső adatokhoz is hozzáférés
    - Vágólap, hálózati információ, UPnP, Push API, kontaktlista, naptár, …
- Böngészőtől függő működés
  - Mindenki másképp csinálja… De egyre hasonlóbban.
- További információ, dokumentáció
  - Mozilla Developer Network
  - *https://developer.mozilla.org/en-US/docs/Web/API*

Mobilra optimalizált web

- Doodle
  - http://en.blog.doodle.com/2012/08/
  - Android alkalmazás megszüntetése a fenntarthatóság problémája miatt
  - Kevés emberrel a webre koncentrálnak inkább
  - Azóta újra van natív kliens…
- Facebook
  - http://www.androidcentral.com/zuckerberg-admits-fault-facebook-s-html5-focus-pledges-native-android-app
  - Zuckerberg: hiba volt a HTML5, natív app kell

MOBIL KÉPELEMZÉS ÉS GRAFIKA

ANDROID FEJLESZTÉS – ALAPOK

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Android

- Az Android egy mobil eszközök számára kifejlesztett **szoftverrendszer**, amely magába foglal egy **operációs rendszert, középszintet** (middleware) **és célalkalmazásokat.**

- Az **Android SDK** olyan eszközöket és alkalmazásprogramozási felületeket (APIs) biztosít, melyek szükségesek az Android platformon való alkalmazásfejlesztés elkezdéséhez, mindezt **Java és Kotlin programozási nyelven**.

Főbb jellemzők

- Az **alkalmazás keretrendszer** lehetővé teszi az egyes alkotóelemek, komponensek ***újrafelhasználását*** és ***helyettesítését***.
- Mobil eszközökre optimalizált **Dalvik (újabban ART) virtuális gép**
- A nyílt forráskódú WebKit motoron alapuló **beépített böngésző**
- Egy 2D grafikai programkönyvtárra épülő **optimalizált grafikai műveletek**
  - OpenGL ES 1.1, 2.0, 3.0-ra épülő 3D grafika
  - Hardver gyorsítás opcionálisan
- **SQLite** strukturált adattároláshoz
- **Médiatámogatás** a közismert audio, video és állókép formátumokhoz
  - MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF
- **GSM telefonálás** (hardver függő)
- **Bluetooth, EDGE, 3G és WiFi** (hardver függő)
- **Kamera, GPS, iránytű és gyorsulásmérő** (hardver függő)
- **Gazdag fejlesztőkörnyezet**
  - céleszköz emulációs program,
  - hibakereső eszközök (debug),
  - bővítmény az Eclipse integrált fejlesztői környezethez, újabban Android Studio,
  - lehetővé teszi memória- és teljesítményprofilok létrehozását.

Könyvtárak

- Az Android magába foglal jó néhány **C/C++ programkönyvtárat**, amelyeket az Android rendszer különböző alkotóelemei használnak. Ezen képességek/funkcionalitások a fejlesztőkhöz az Android alkalmazás-keretrendszerén keresztül jut el. Néhány a könyvtárak közül az alábbi listán látható:
  - **C rendszerkönyvtár:** a C nyelv rendszerkönyvtárának (*libc*) egy „BSD-leszármazott” megvalósítása, beágyazott, Linux-alapú eszközökre optimalizálva.
  - **Média könyvtárak:** a PacketVideo OpenCORE multimédiás alrendszerén alapul; a könyvtárak támogatják több elterjedt audio, video és állókép formátumú fájl rögzítését és visszajátszását (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG stb.)
  - **Felület menedzser:** a kijelző alrendszerhez való hozzáférést kezeli, és zökkenőmentesen rak össze 2D és 3D grafikus rétegeket több alkalmazásból
  - **LibWebCore:** egy modern web böngésző motor, amely egyszerre „hajtja meg” az Android böngészőt és egy beágyazható web nézetet.
  - **SGL:** a mögöttes 2D grafikus motor
  - **3D könyvtárak:** egy OpenGL ES 1.0 és 2.0 API-kon alapuló megvalósítás; a könyvtárak használnak mind 3D gyorsítást (ha elérhető), mind az erősen optimalizált szoftveres 3D raszterezőt
  - **FreeType:** bittérkép és vektorgrafikus szöveg renderelés
  - **SQLite:** egy minden alkalmazás számára hozzáférhető hatékony, de egyszerű relációs adatbázis motor

Android architektúra

Alkalmazások

- Alap alkalmazások előre telepítve
  - E-mail kliens, SMS program, naptár, térkép, web böngésző, névjegyek stb.
  - Java nyelven készülnek

- Új alkalmazások
  - Fejlesztők számára elérhetők a hardver elemek, hely információ; háttér szolgáltatások futtathatók, riasztások, értesítések adhatók stb.

Alkalmazás keretrendszer

- A fejlesztőknek **teljes hozzáférésük van ugyanahhoz a keretrendszerbeli alkalmazásprogramozási felülethez**, amelyet az **alapvető alkalmazások** is használnak.
- Az alkalmazás felépítése úgy van megtervezve, hogy **egyszerűsítse a komponensek újrafelhasználását**; bármely alkalmazás nyilvánossá tudja tenni képességeit, így bármely másik alkalmazás felhasználhatja azokat (a szükséges biztonsági megszorításokról a keretrendszer gondoskodik).
- Szintén ezen mechanizmus teszi lehetővé **a komponensek egymással való helyettesítését a felhasználó számára.**

Futtató környezet

- Az Android tartalmaz **néhány alapvető könyvtárat**, amelyek biztosítják a Java programozási nyelv magkönyvtáraiban található legtöbb funkcionalitást
- **Minden Android alkalmazás a saját processzét futtatja**, a **saját Dalvik Virtuális Gép** (DVM) példányával együtt
- A Dalvik úgy lett megírva, hogy egy eszköz hatékonyan futtathassa több virtuális példányát. A DVM a Dalvik Executable *(.dex)* formátumban futtatja a fájlokat, mely a lehető legkisebb memória-felhasználásra lett optimalizálva. A DVM regiszter alapú, és a Java nyelv által lefordított osztályokat a beépített DX eszköz által *.dex* formátumúra átalakítva hajtja végre.
- A DVM mögöttes funkcionalitásért, úgy mint végrehajtási szálak (thread) használata vagy alacsonyszintű memóriakezelés, a **Linux kernelhez** fordul.
- Az újabb **ART virtuális gép** az alkalmazás telepítésekor végi el a fordítást
  - Lassabb a telepítés és nagyobb helyet foglal el az alkalmazás
  - Cserébe gyorsabb az alkalmazások indítása és futtatása

Linux Kernel

- Az Android a Linux magra támaszkodik
  - 2.6 a korábbi Android verziók esetén
  - 3.x az ICS verzió óta
  - **Alapvető rendszer szolgáltatások**, mint például biztonság, memóriakezelés, processzuskezelés, hálózati protokoll verem, illesztőprogram modell.
  - A kernel egy, a hardver és a szoftverrendszer többi része közti **absztrakciós rétegként** is működik.

Alkalmazások alapjai

- Android alkalmazások **Java programozási nyelven** íródnak
- A lefordított Java kód – bármilyen adat- és az alkalmazás által kért erőforrásfájllal egyetemben – egy ***Android package***-be van becsomagolva, amely egy **.apk** kiterjesztésű archív fájl. Egy *.apk* fájlban lévő össze kód tekinthető egy *alkalmazásnak*.

Sok tekintetben egy Android alkalmazás „a saját világában létezik”:

- **Minden alkalmazás a saját Linux processzét futtatja.** Az Android elindítja a processzt amikor valamely alkalmazás kódja végrehajtandó, és leállítja azt, amikor nem szükséges már és egyéb alkalmazásoknak szükségük van (rendszer) erőforrásra.
- **Minden processznek meg van a saját virtuális gépe (DVM),** azaz az alkalmazás kódja egyéb alkalmazásoktól elkülönítve hajtódik végre.
- **Alapértelmezésben minden alkalmazás számára ki lesz osztva egy (egyedi) Linux felhasználói azonosító.** Az engedélyek úgy vannak beállítva, hogy az alkalmazás fájljai csak az adott felhasználó és csak az alkalmazás számára legyenek láthatók – habár van mód egyéb alkalmazások számára elérhetővé tenni őket.

Alkalmazás komponensek

- Aktivitások *(Activity)*
  - Vizuális felület a felhasználó számára a feladat elvégzéséhez
  - Egy alkalmazás egy vagy több aktivitásból állhat
  - Nézet osztály leszármazottjai használhatók
- (Háttér)szolgáltatások *(Service)*
  - Nincs UI, háttérben fut
  - Pl. zenelejátszás, pozíciófigyelés, ütemezett ellenőrzések
- Broadcast-fogadók *(Broadcast receiver)*
  - Kiválasztott típusú üzenetre reagálhat
    - Pl. időzóna változott, kép elkészült, bejövő hívás, rendszer nyelvi beállítása változott
  - Nincs UI, de indíthat aktivitást, vagy adhat értesítést
- Tartalomszolgáltatók *(Content provider)*
  - Adatokat szolgáltatnak más alkalmazásoknak
    - Azok API-n keresztül, felügyelten érhessék el, ne közvetlenül az adatbázist manipulálják
  - Pl. kontakt információk, híváslista, naptár, ...

Komponensek aktiválása

- „Intent” objektum
  - Aszinkron üzenetek
    - Aktivitások, broadcast-fogadók, szolgáltatások aktiválására
    - Ezek közötti kommunikációra, paraméterátadásra
  - Fő típusai
    - **Direkt**: megmondjuk pontosan melyik csomag hajtódjon végre
    - **Indirekt**: szándékot közlünk egy feladat végrehajtására (pl. email küldése, FTP, ...), választhatunk a rendszerben regisztrált alkalmas végrehajtó komponensek közül
- Feladatok
  - Aktivitások sorozata
    - Akár más alkalmazás aktivitása is lehet
  - Aktivitás verem
  - „Vissza” gomb: előző aktivitás a veremből

AndroidManifest.xml

- Komponensek adatainak összefoglalására
  - Minden APK csomag tartalmazza
  - Rögzített a neve
- Tartalma
  - Az alkalmazás indítóikonja, címkéje
  - A komponenshez tartozó aktivitás(ok) neve(i)
  - Intent szűrők (mire reagálnak az egyes komponensek)
  - Engedélyek
    - Milyen engedélyköteles szolgáltatásokat használ: pl. telefonhívások kezdeményezése, GPS, kontakt adatokhoz hozzáférés, kamera használat, …
      - A felhasználók sokszor nem olvassák el, valamint nehézkes a jogos használatról meggyőződni.
    - Android 6.0-tól kezdődően **nem telepítéskor, egyszerre történik ezek felhasználói általi elfogadása, hanem a program futása során a kérdéses elem első használatakor**
      - A program indoklást adhat, miért van szüksége az elemre
      - Ha nem kap engedélyt, futás közben kezelheti, a program más részei működőképesek lehetnek ettől (mondjuk egy chat program internet hozzáférés megadása nélkül nem sok mindenre jó…)
  - …

Komponensek életciklusa

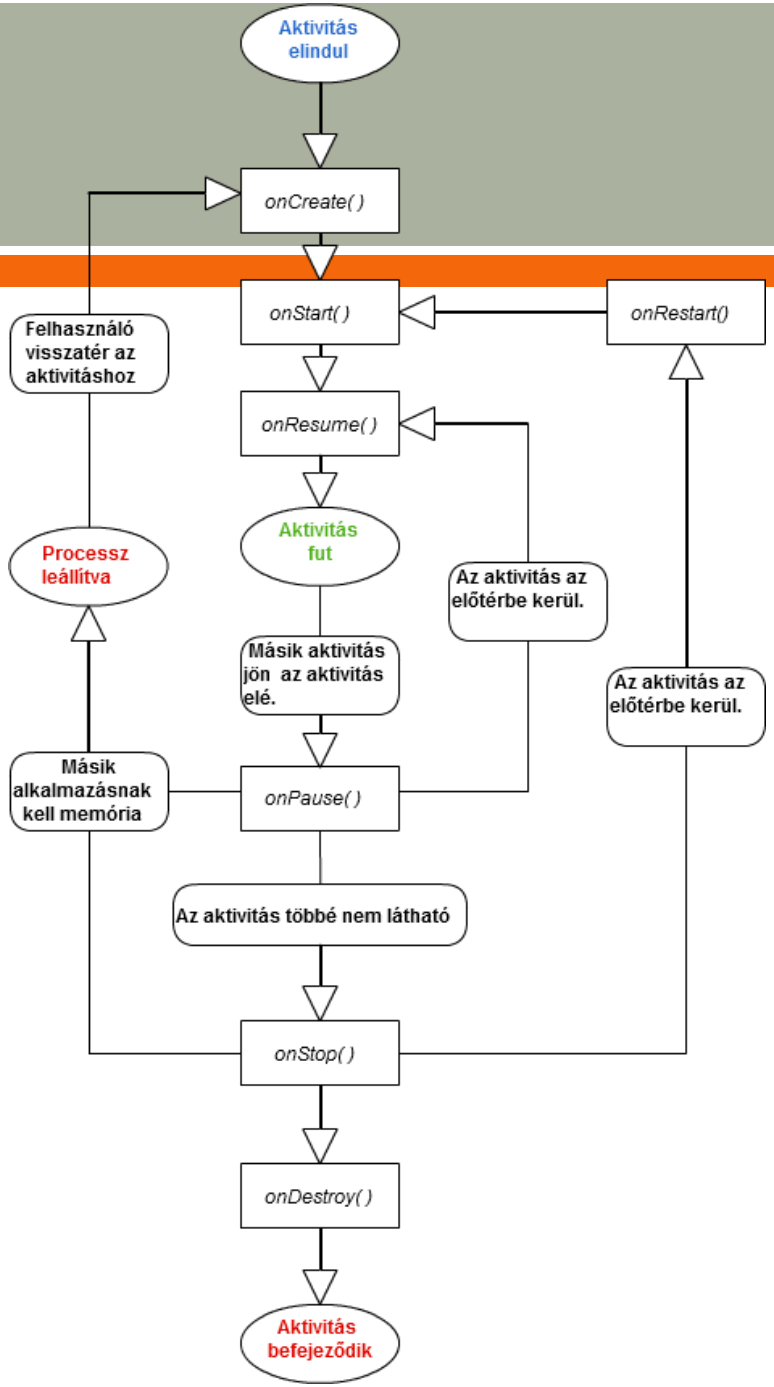
Egy aktivitásnak lényegében 3 állapota van:

- **Aktív vagy fut,** amikor **a képernyőn az előtérben van** (az aktivitás veremben legfelül helyezkedik el).
  - Ez az aktivitás kezeli a felhasználói tevékenységet.
- **Felfüggesztett**, ha nem az aktivitás reagál a felhasználói tevékenységre, de még látható a felhasználó számára.
  - Akkor van, amikor egy másik alkalmazás az addig aktív/futó felé kerül, és ezen aktivitás vagy átlátszó, vagy nem tölti ki az egész képernyőt, szóval néhány megállított alkalmazás látszódhat. Egy megállított aktivitás még teljes mértékben megmarad (fenntart minden állapot információt, és továbbra is csatlakozik az ablak kezelőhöz), de a rendszer véget vethet neki nagyon kevés rendelkezésre álló memória esetén.
- **Megállított**, ha teljesen háttérbe kerül egy másik aktivitás miatt.
  - Továbbra is fenntartja minden állapot és belső információját. Többé nem látható a felhasználó számára, ezért a hozzá tartozó ablak rejtett, és **általában a rendszer kilövi**, amikor szükség van memóriára.

Komponensek életciklusa

- Ha egy aktivitás felfüggesztett vagy megállított állapotban van, a rendszer kitörölheti a memóriából anélkül, hogy **„megkérné a befejezésre”** (*finish( )* eljárás meghívása), **vagy egyszerűen csak megszüntetheti a processzét**. Amikor újra a felhasználó elé kerül, **teljesen újraindítandó és visszaállítandó az előző állapotába**.
- Mivel egy aktivitás állapotból állapotba vált, a változásokat védett (*protected*) eljárások felüldefiniálásával kezelhetjük:

  void onCreate(Bundle *savedInstanceState*)
  void onStart()
  void onRestart()
  void onResume()
  void onPause()
  void onStop()
  void onDestroy()
- **Minden aktivitásnak implementálnia kell onCreate() eljárását**, hogy elvégezze a kezdeti beállításokat az objektum első példányosításakor.
- Sok alkalmazás az **onPause()** eljárást is implementálja, hogy kezelje az adatváltozásokat és ezen felül felkészül a felhasználóval való együttműködés befejezésére.
  - Pl. osztott erőforrások (pl. kamera) visszaadása a rendszernek, hogy más, éppen előtérbe kerülő alkalmazás használhassa
- Leállításkor az aktivitás mentheti saját adatait, hogy újraindításkor ne „tiszta lappal” induljon, hanem a legutóbbi állapotból
  - *onSaveInstanceState()* függvény + *Bundle* objektum



- *onCreate()*
  Az aktivitás első meghívásakor hívódik meg. Itt kell elvégezni az összes statikus beállítást: nézetek készítése, adatok lekötése listákhoz stb. Egy *Bundle* objektumot vár paraméterben, mely tartalmazhatja az aktivitás előző állapotát. Mindig *onStart()* követi.

- *onRestart()*
  Az aktivitás megállítása után hívódik meg, **közvetlenül azelőtt, hogy újra elindulna.** Mindig *onStart() követi.*

- *onStart()*
  Közvetlen azelőtt hívódik meg, hogy az aktivitás láthatóvá válna a felhasználó számára.

- *onResume()*
  Közvetlenül azelőtt hívódik meg, hogy az aktivitás interakcióba lépne a felhasználóval. Ilyenkor az aktivitás az aktivitás verem tetején van, a felhasználói bemenetelet ő fogadja. Mindig *onPause()* követi.

- *onPause()*
  Akkor hívódik meg, **amikor a rendszer egy másik aktivitást akar folytatni.** Általában adatváltozások mentésére, animációk és egyéb CPU-igényes feladatok leállítására használják. Bármit is csinál, azt **nagyon gyorsan kell tennie**, mivel a következő aktivitás addig nem folytatódik, amíg vissza nem tér.

- *onStop()*
  Akkor hívódik meg, **ha az aktivitás többé nem látható a felhasználó számára.** Ez történhet, ha az aktivitás megszűnik, vagy ha egy másik folytatódik, és elfedi ezt.

- *onDestroy()*
  Az aktivitás megszűnésekor hívódik. Ez történik, ha az aktivitás befejeződik, vagy ha a rendszer ideiglenesen megszűnteti ezt az aktivitáspéldányt. A két eset között az *isFinshing()* metódussal lehet különbséget tenni.

Processzusok életciklusa

Az Android rendszer igyekszik megtartani egy alkalmazás processzt ameddig csak lehetséges, de időnként szükséges eltávolítani régi processzeket, ha a memória elfogy. Meghatározandó mely processzeket tartsa meg vagy szüntesse meg, a rendszer „fontossági hierarchiába” sorolja a processzeket a bennük futó komponensek és azok állapota alapján. 5 szint van a hierarchiában, a következőkben ezek láthatóak fontosság szerint csökkenő sorrendben:

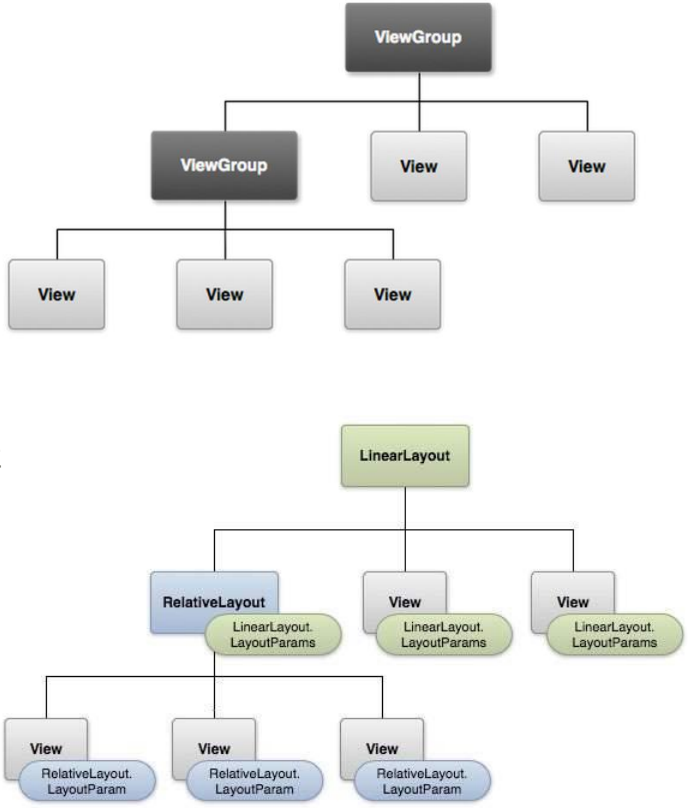
- **Előtérben lévő processz**, mely ahhoz szükséges, amit a felhasználó éppen csinál. Egy processz az előtérbe kerül bármely következő fennállása esetén.
  - Olyan aktivitást futtat, amellyel a felhasználó interakcióban van (ilyenkor az *Activity* objektum *onResume()* metódusa meghívódik).
  - Olyan szolgáltatást tartalmaz, amely szükséges a felhasználóval interakcióban lévő aktivitáshoz.
  - Tartalmaz egy *Service* objektumot, amely egy életciklus visszahívást hajt végre (*onCreate(), onStart()* vagy *onDestroy()).*
  - Tartalmaz egy *BroadcastReceiver* objektumot, amely az *onReceive()* eljárását hajtja végre.
  - Csak néhány előtérben lévő processz létezhet egy adott pillanatban. Legvégső esetben megszűnik egy, ha annyira kevés a memória, hogy nem tudja mind folytatni a futását. Általában ilyenkor a memória betelik, tehát a felhasználói interfész működőképessége érdekében meg kell szüntetni néhány processzt.
- **Látható processz** az, melynek nincs előtérbeli komponense, de hatással lehet arra, amit a felhasználó a képernyőn lát. Egy processz látható, ha az alábbiak valamelyike teljesül:
  - Olyan aktivitást nyújt, amely bár nincs a előtérben, látható a felhasználó számára (*onPause()* metódusa meghívódott). Ez történik például akkor, ha az előtérben lévő aktivitás egy párbeszéd ablak, amely mögött még látszódik az előző aktivitás.
  - Olyan szolgáltatást nyújt, amely szükséges egy látható aktivitás számára.

Processzusok életciklusa

- **Szolgáltatás processz** az, mely futtat egy szolgáltatást, mely a *startService()* metódussal lett elindítva, de nem tartozik az előző két kategóriába. Bár a szolgáltatás processzek nem köthetők közvetlenül valamihez, amit a felhasználó lát, olyan dolgokat csinálnak, amivel a felhasználó törődik (pl. mp3 lejátszása a háttérben, adat letöltése a hálózaton), így a rendszer futtatja ezeket, hacsak nincs elég memória, hogy az előtérbeli és látható processzekkel együtt fussanak.
- **Háttér processz** az, mely olyan aktivitást tartalmaz, amely éppen nem látható a felhasználó számára (az aktivitás *onStop()* eljárása meg lett hívva). Ezen processzeknek nincs közvetlen hatásuk a felhasználói élményre, és bármikor megszüntethetők annak érdekében, hogy egy fontosabb processznek legyen memóriája. Általában sok háttér processz fut, így azok egy **LRH (legritkábban használt**, angolul Last Recently Used) **listában** vannak nyilvántartva biztosítva, hogy a felhasználó által a korábbiakban legtöbbet használt aktivitás processze legyen legkésőbb megszüntetve. Ha egy aktivitás életciklus eljárásait helyesen implementálja és megőrzi az aktuális állapotát, a processz törlésének nem lehet káros hatása a felhasználói élményre.
- **Üres processz** az, mely **nem tartalmaz aktív alkalmazás összetevőt**. Az üres processz olyan folyamat, amelynek nincs aktív komponense. Ilyen folyamatokat csak cache gyanánt használunk, hogy amikor egy komponens legközelebb futtatni akarja, gyorsabban induljon. A rendszer gyakran kilövi ezeket a folyamatokat, hogy a rendszererőforrások kiegyenlítetten legyenek elosztva a folyamatcache-ek és a mögöttes kernelcache-ek között.

Felhasználói felület (UI)

- Egy Android alkalmazásban **a felhasználói interfész** *View* (nézet) és *ViewGroup* (nézetcsoport) elemekből épül fel.
  - Sokféle nézet és nézetcsoport létezik, mindegyik a *View* osztály leszármazottja.
- A nézet objektumok a felhasználói interfész alapegységei.
  - A ***View*** osztály alapját képezi a **widget**-eknek (vezérlő elemek) nevezett alosztályoknak, melyek kész grafikus elemeket jelentenek úgy, mint szövegmezők és gombok.
- A ***ViewGroup*** osztály alapját képezi a **layout**-oknak (elrendezéseknek) nevezett alosztályoknak
  - Különböző elrendezésű megjelenítést tesznek lehetővé úgy, mint sorban, táblázatban, valamihez képest relatívan elhelyezett elemek.
- **Egy *View* objektum egy olyan adatszerkezet, amelynek paraméterei tárolják az elrendezés tulajdonságait és tartalmát a képernyő egy adott téglalap alakú részéhez.**
  - A *View* objektum kezeli a saját méretét, elrendezésének (layout) rajzolását, fókuszváltozását, görgetését/lapozását és a képernyő adott részéhez tartozó felhasználói beavatkozásokat (billentyűk, gesztusok). Egy felhasználói interfészen belüli objektumként, egy *View* szintén lehet a felhasználó számára egy visszajelzési pont és a felhasználói beavatkozásokhoz tartozó események kezelője.



Nézet hierarchia

- Egy tevékenység (*Activity*) felhasználói interfésze egy hierarchiaként definiálható, melynek csomópontjai *View* és *ViewGroup* objektumok.
- Ez a hierarchia-fa olyan egyszerű vagy összetett lehet, amilyenre csak szükségünk van, és felépíthető mind az Androidban előre definiált widget és layout objektumok, mind egyéni vezérlők és elrendezések felhasználásával, melyeket mi készítünk el.
- Gyökérelem beállítása egy aktivitáshoz: ***setContentView(...)***
- Minden *ViewGroup* objektum felelős felszólítani *View* gyermekeit, hogy „rajzolják meg magukat”.
- **A gyermekei kérhetnek egy méretet és elhelyezést** a szülőn belül, és **a szülő objektum dönt**, hol és mekkora lehet minden gyerek objektum.

Elrendezés (Layout)

- Az elrendezést és nézet hierarchiát általában egy **XML elrendezés fájl** definiálja.
- **Minden elem az XML-ben vagy *View* vagy *ViewGroup* objektum** (vagy azok leszármazottja). A *View* objektumok levelek, a *ViewGroup* objektumok elágazások a fában.
- Egy XML elem neve megfelel az őt megvalósító Java osztálynak, például: **TextView**, **LinearLayout**.
- Amikor betöltünk egy elrendezést, az Android rendszer **inicializálja ezen futás idejű objektumokat**, a benne lévő elemeknek megfelelően.
- Programkódból ezekhez az elkészült objektumokhoz tudunk kapcsolódni
  - **`findViewById()`** függvénnyel
  - Az azonosítót az XML fájlban definiáljuk

Vezérlő elemek

- Egy **vezérlő elem** egy olyan *View* objektum, amely **a felhasználóval való közreműködésre szolgáló interfész.**
- Az Android biztosít **néhány teljesen implementált vezérlő elemet (gombok, választó mezők, szövegbeviteli mezők)**, ezekkel gyorsan építhető egy UI (felhasználói interfész).
- Az Android néhány vezérlő eleme **összetettebb**, mint például **egy adat begyűjtő, egy óra vagy egy zoom kezelő.**
- De nem vagyunk az Android nyújtotta vezérlő elemekre korlátozva. Létrehozható saját, specifikus elem **saját *View* objektum definiálásával**, vagy már létezők kiterjesztésével, kombinálásával.

UI események

- Egy hozzáadott View/widget elem esetén nyilván tudni akarunk a vele való felhasználói interakcióról, hogy egy bizonyos működést végrehajtva válaszolhassunk rá. Az UI események kezelésére két lehetőség van:
  - **Definiálni egy esemény figyelőt, és hozzárendelni az adott *View* objektumhoz.** Általában így figyeljük az eseményeket. A *View* osztály beépített interfészeket tartalmaz *On<valami>Listener* névvel, mindegyikhez egy *On<valami>()* visszahívható függvény tartozik.
    - Például *View.OnClickListener* („kattintások” kezelésére), *View.OnTouchListener* (érintőképernyő eseményeinek kezelése), és *View.OnKeyListener* (fizikai gombok kezelése). Tehát, hogy egy *View* objektum értesüljön arról, hogy rákattintottak (pl. egy gomb megnyomása által), implementálni kell *OnClickListener*-t definiálva az *onClick()* eljárását (ahol reagálhatunk a kattintásra) és hozzárendelve azt a *View*-hoz a *setOnClickListener()*-rel.
  - **Valamely *View*-hoz tartozó visszahívható eljárás felüldefiniálása.** Ez a teendő saját *View* típusú osztály megvalósításakor. Többek között kezelhető így a képernyő megérintése (*onTouchEvent()*), a joystick megmozdítása (*onTrackballEvent()*) vagy egy gomb megnyomása az eszközön (*onKeyDown()*). Ezáltal megadható minden eseményhez az alapértelmezett viselkedés, és meghatározható, hogy az esemény tovább legyen-e adva valamely gyerek *View*-nak. Tehát, mivel ezek a *View* osztály visszahívható eljárásai, az egyetlen lehetőség definiálni őket a saját komponens létrehozása.

Elrendezések (layout) megadása

Kétféleképpen adható meg az elrendezés:

- **UI elemek megadása XML-ben**
  - Az Android egyértelmű XML kulcsszavakkal rendelkezik a View osztályok és leszármazott osztályoknak megfelelően.
  - Ezen megoldás előnye, hogy az alkalmazás megjelenéséért felelős kódot elválasztja a viselkedésért felelőstől. Az UI leírás kívül van az alkalmazás kódján, azaz módosítható, átdolgozható a forráskód módosítása és újrafordítása nélkül.
- **Elrendezés elemek példányosítása futás közben**
  - Az alkalmazások képesek létrehozni *View* és *ViewGroup* objektumokat és befolyásolni azok tulajdonságait programból.

Az Android keretrendszer lehetővé teszi bármelyik, vagy mindkettő együttes használatát is.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
        <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
        <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

XML betöltése

- Az alkalmazás lefordításakor minden XML layout fájlból létrejön egy erőforrás.
- Ezt az erőforrás fájlt be kell tölteni az alkalmazás kódjában, az *Activity.onCreate()* visszahívható eljárásból. Ez a *setContentView()* meghívásával történhet, amelynek át kell adni az erőforrásra való hivatkozást *R.layout.<elrendezés_fájl_neve>* alakban. Például, ha az XML main_layout.xml néven van elmentve, így tölthető be:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

- Az *onCreate()* visszahívható eljárást a aktivitásban az Android keretrendszer hívja meg, amikor elindítjuk az alkalmazást.

XML attribútumok

- Minden *View* és *ViewGroup* objektum saját XML attribútumokat támogat.
- Vannak specifikus attribútumok egy View objektumhoz (például a *TextView* támogatja a *textSize* attribútumot), de ezek az attribútumokat örökli bármely View objektum, amely kiterjeszti az osztályt.
- Néhány közös minden View objektum számára, mert a gyökér View objektumból van örökölve (például az *id* attribútum).
- És vannak még az „elrendezés paraméterek”, amelyek a View objektum konkrét elhelyezkedését adják meg, ahogy a szülő ViewGroup objektumban meg van határozva.

XML attribútumok

- ID
  - **Bármely View objektum társulhat egy egész szám azonosító**, mely egyértelműen azonosítja az objektumot a fán belül. Az alkalmazás lefordításakor erre az ID-ra egész számként hivatkoznak, de tipikusan az XML fájlban van meghatározva karakterláncként, az id attribútumban. Ez egy minden View objektum számára elérhető attribútum (View osztályban által van megadva), és gyakran használatos. A szintaxis az XML-en belül a következő:

```
ANDROID:ID="@+ID/MY_BUTTON”
```

  - A kukac jel (**@**) az elején **azt jelzi, hogy az XML elemzőnek elemeznie kell és ki kell terjesztenie a sztring hátralévő részét, és egy ID erőforrásként kell beazonosítania.**
  - A plusz jel (**+**) **azt jelenti, hogy ez egy új forrás neve, amit létre kell hozni és hozzá kell adni az erőforrásainkhoz (az R.java fájlban).**
  - Számos, az Android keretrendszer által nyújtott, egyéb ID erőforrás létezik még. Amikor egy Android erőforrás ID-re hivatkozunk, nincs szükség a plusz jelre, de meg kell adni a csomag névterét, például:

```
ANDROID:ID="@ANDROID:ID/EMPTY"
```

Megadva a névteret, egy az android.R erőforrás osztályban található ID-re hivatkozunk, a helyi erőforrás osztály helyett.

Erőforrások a kódban

Nézetek készítésére és az alkalmazásból rájuk való hivatkozásra egy minta:

- Definiálni egy view/widget-et az elrendezés fájlban, és megadni hozzá egy azonosítót:

```
<Button android:id="@+id/my_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/my_button_text"/>
```

- Majd a Java változót összekötni az erőforrás elemmel (tipikusan az *onCreate()* eljárásban):

```
Button myButton = (Button) findViewByld(R.id.my_button);
```

Elrendezés paraméterek

- A *layout_<valami>* nevű XML elrendezés attribútumok a View számára elrendezés paramétereket határoznak meg a, amelyek megfelelnek a ViewGroup objektumnak, amelyben az megtalálható.
- Minden ViewGroup objektum implementál egy beágyazott osztályt, amely a *ViewGroup.LayoutParams*-ot terjeszti ki. Ez az alosztály különböző típusú tulajdonságokat tartalmaz, melyek minden gyermek objektum méretét és pozícióját megadják.
- Minden LayoutParams alosztály meg van a saját szintaxisa az értékek beállítására. Minden gyerek elemnek meg kell adnia a szülőnek megfelelő LayoutParams paramétereket, noha az szintén megadhat különböző paramétereket a saját gyerekeinek.
- **Minden nézetcsoport tartalmaz szélességet és magasságot** (*layout_width* és *layout_height*)**, és minden nézetnek definiálnia kell azokat.** Sok LayoutParams tartalmaz még margókat és szegélyeket is.
- Megadható egzakt mértékként a szélesség és a magasság, habár ez ritka. Gyakoribb a következő konstansok valamelyikének használata:
  - *wrap_content* a nézetet a tartalmának megfelelő méretűre méretezi
  - *fill_parent* (*match_parent* az API level 8-tól kezdődően) a nézetet akkorára méretezi, amekkorának hagyja a szülő nézetcsoportja.
- Általában, **egy elrendezés méreteinek abszolút egységekkel (pl. pixel) való megadása nem ajánlott.** Ehelyett relatív mértékek, mint például tömörség-független pixel egység (*dp*), *wrap_content, fill_parent*, használata egy jobb megközelítés, mert ez segít biztosítani, hogy az alkalmazás helyesen jelenjen meg különböző méretű kijelzőkön.

Tutorialok

- Példaprogramok
  - ArithmTest_01.zip, ArithmTest_02.zip, ArithmTest_03.zip
- CodeProject
  - Learn How to Develop Android Application
  - Android UI Layouts and Controls
  - Beginner's Guide to Organizing/Accessing Android Resources
  - Android Touch Gestures Capturing Interface
  - Article 5 - Android User Interactivity and Sensors
- Android Developer portál
  - Training for Android developers
  - Notepad Tutorial (Android Developer Portal)

MOBIL KÉPELEMZÉS ÉS GRAFIKA

ANDROID FEJLESZTÉS – GYAKORLAT

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Fejlesztői környezet elemei

- SDK Manager
  - SDK-k letöltése, karbantartása
  - Platform tools (adb, ...)
    - Parancssori kapcsolat csatlakoztatott eszközzel
- AVD Manager
  - Virtuális eszközök definiálása, emulátorban teszteléshez
- Java JDK
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Oracle Java telepítése javasolt

Fejlesztői környezet

- Android Studio
  - http://developer.android.com/sdk/index.html
  - Ingyenes, multiplatform
  - Ez az alapértelmezett, célszerű ezt használni!
  - **Integráltan tartalmazza a szükséges eszközöket és segédprogramokat**

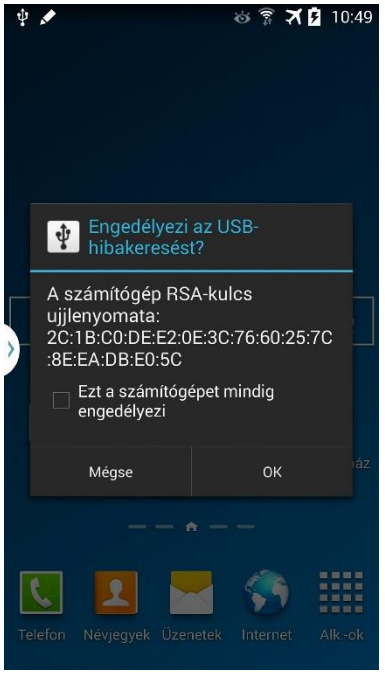
- Eclipse + ADT plugin
  - **Elavult!**
  - Ingyenes, multiplatform
  - ADT plugint külön telepíteni kell

Android Studio

- SDK kezelése, frissítése
  - Tools / Android / SDK Manager
    - `Available packages`: SDK letöltés, frissítés
- Virtuális eszközök kezelése
  - Tools / Android / AVD Manager
    - `Virtual devices`: virtuális eszközök létrehozása

Hardver előkészítés

- Android készülék előkészítése
  - **Beállítások / Alkalmazások / USB Debugging** bekapcsolása
  - **Beállítások / Fejlesztői eszközök / USB nyomkövetés**
  - Ha nincs Fejlesztői beállítások menüpont
    - **Beállítások / A telefonról / Build** számot tapogatni
    - Megfelelő számú tapintás után aktiválódik
  - PC-hez csatlakoztatás után a kapcsolat elfogadása a készüléken



- PC
  - Windows: USB driver telepítése szükséges lehet
  - Mac OS X: nincs teendő, „just works”
  - Linux: általában megy, ha nem, akkor engedélyt kell adni a hardver eléréséhez

Hardver előkészítés

- PC (folytatás)
  - Ubuntu (Linux), root-ként
    - `/etc/udev/rules.d/51-android.rules` fájl létrehozása

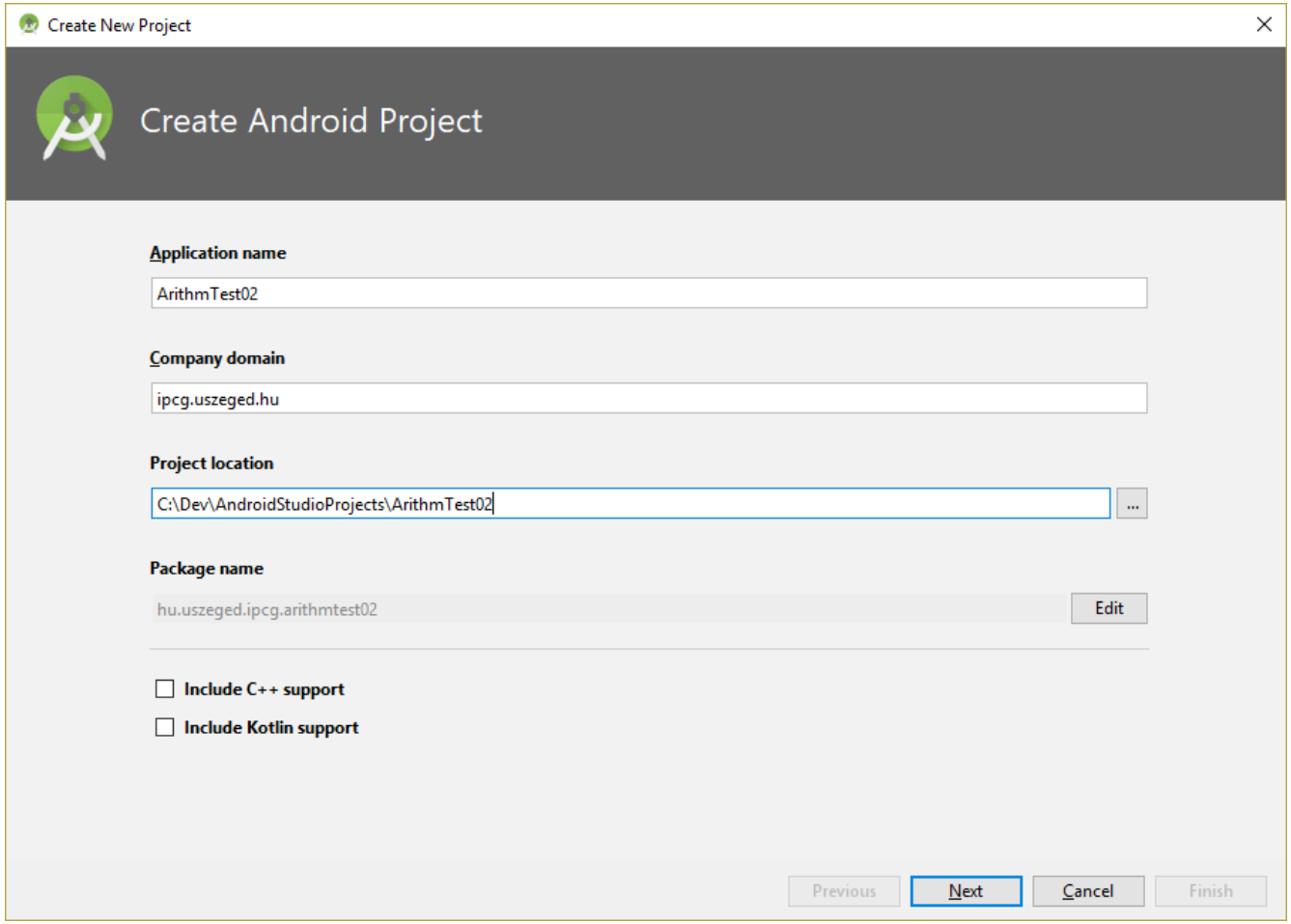
      ```
      SUBSYSTEM=="usb", SYSFS{idVendor}=="0bb4", MODE="0666"
      ```

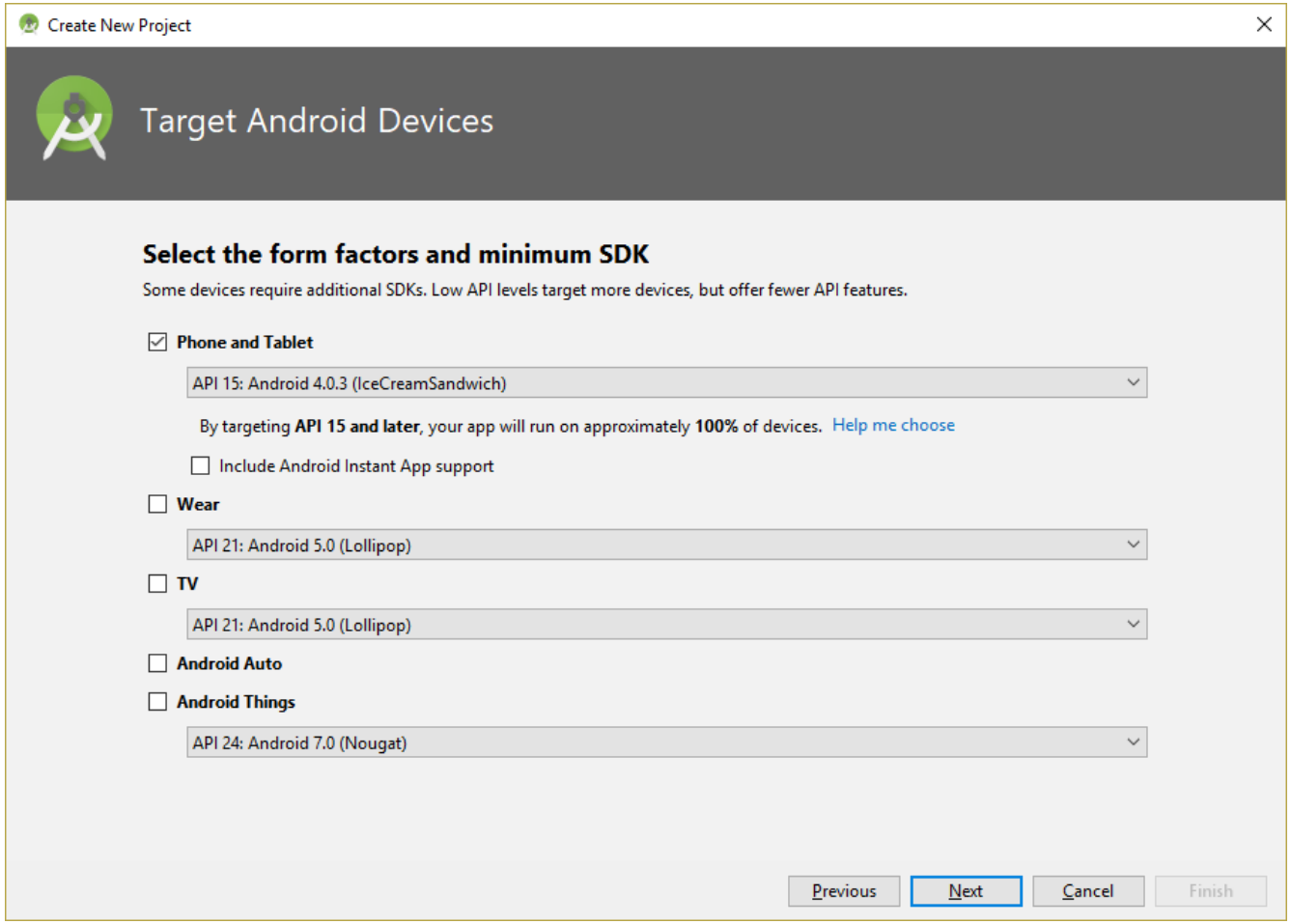
    - `chmod a+r /etc/udev/rules.d/51-android.rules`
  - Vendor kódok
    - `lsusb` listából kiolvasható, vagy Android fejlesztői oldalon
    - Elérhetőség ellenőrizhető (platform_tools): `./adb devices`

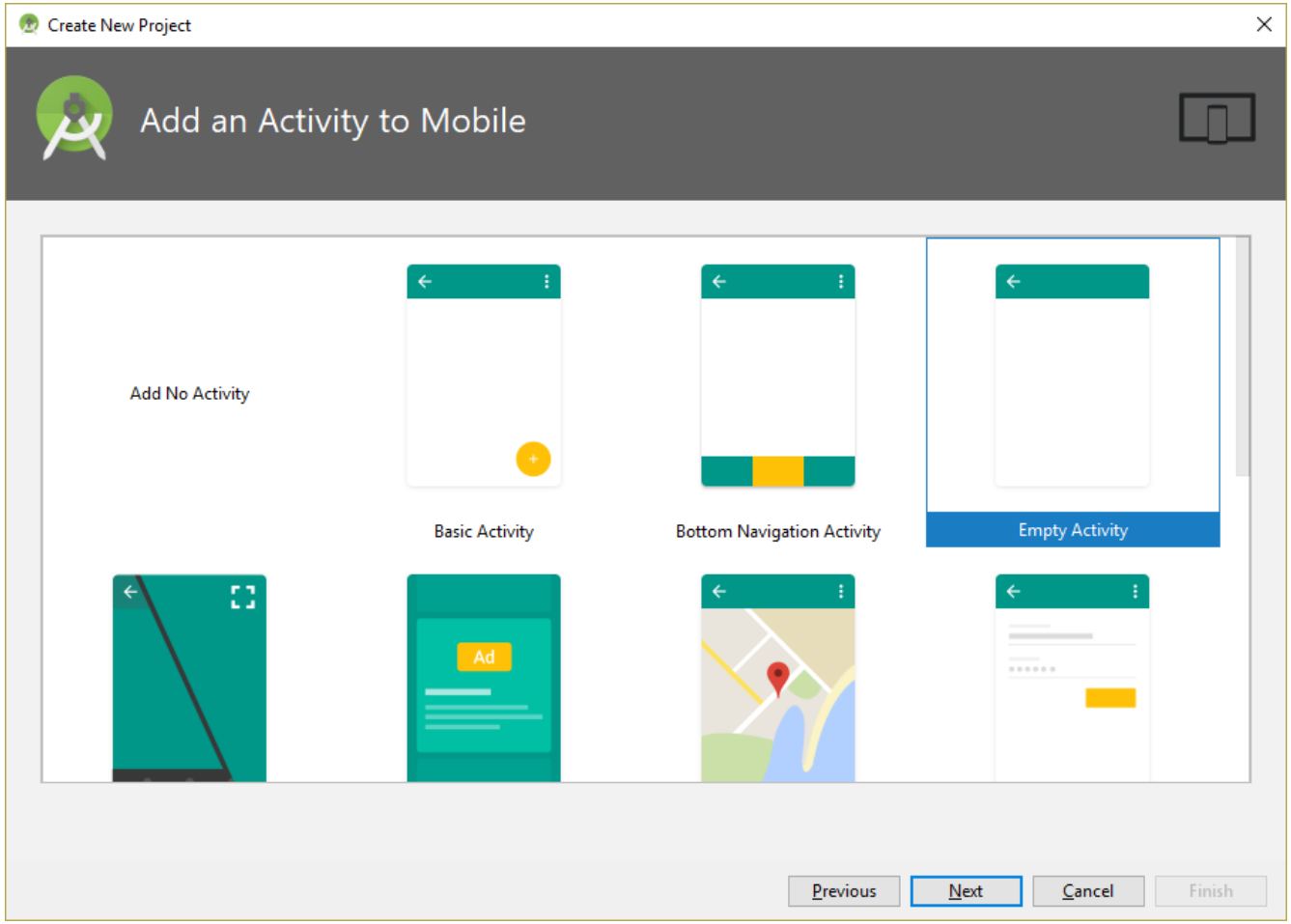
Gyártó	Kód	Gyártó	Kód	Gyártó	Kód
Acer	0502	Huawei	12d1	Pantech	10A9
Dell	413c	Kyocera	0482	Samsung	04e8
Foxconn	0489	LG	1004	Sharp	04dd
Garmin-Asus	091E	Motorola	22b8	Sony Ericsson	0fce
HTC	0bb4	Nvidia	0955	ZTE	19D2

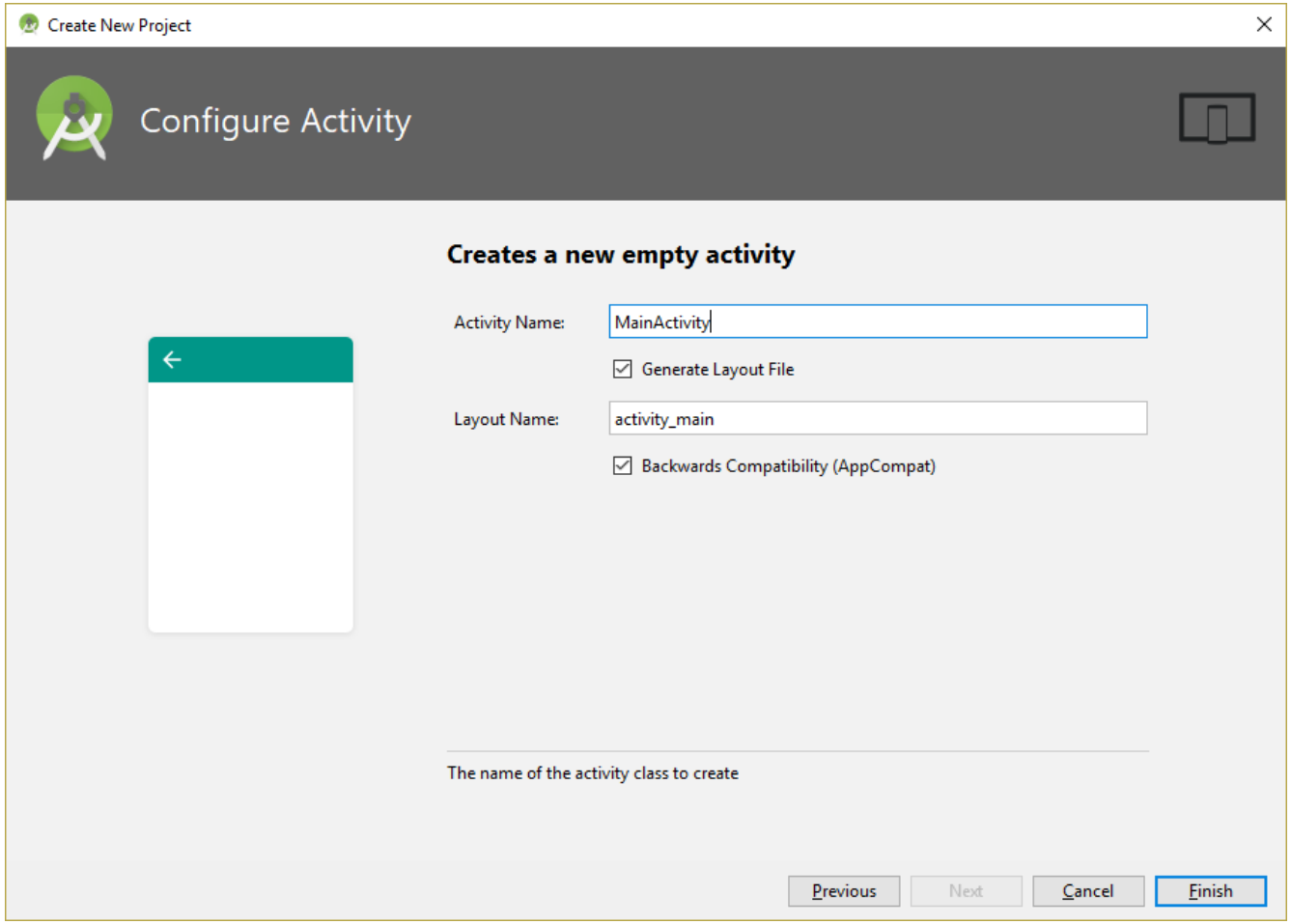
„Hello, World!” lépések

- Android Studio indítása
  - Laborban Linux alatt
    - Bejelentkezés **Android fejlesztői azonosítóval**
    - Parancssor (Terminal) nyitása
    - `cd android_studio/bin`
    - `./studio.sh`
- **Quick Start / Start a new Android Studio project**
  - **Application Name**: **Hello**
  - **Company Domain**: www.example.com; **ipcg.uszeged.hu**; ...
  - **Package Name**: **hu.uszeged.ipcg.helloworld**
- **Form factors and minimum SDK**
  - Phone and Tablet; API 15
- **Activity**
  - **Empty** (esetleg **Blank**)

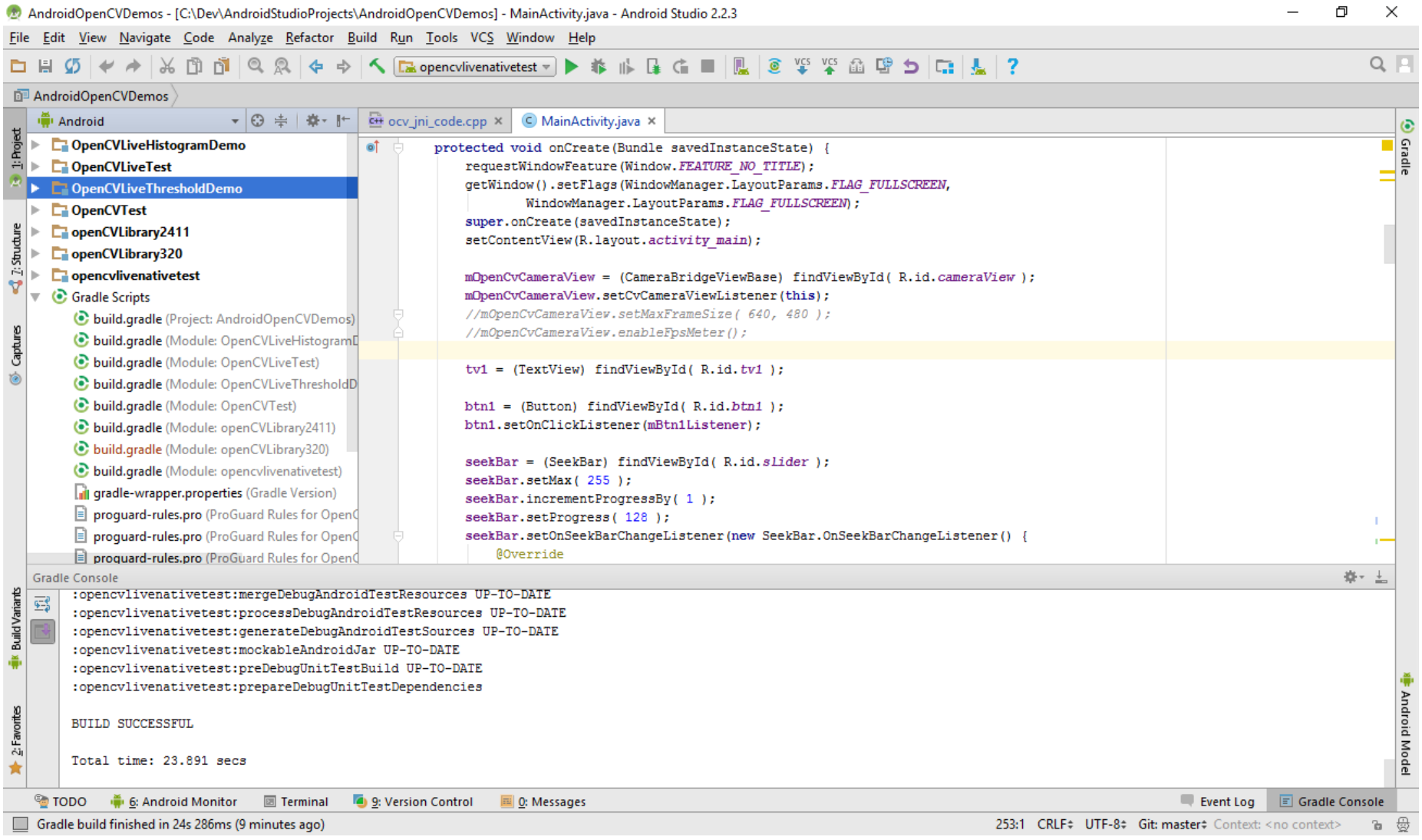

Create New Project
Create Android Project
Application name
ArithmTest02
Company domain
ipcg.uszeged.hu
Project location
C:\Dev\AndroidStudioProjects\ArithmTest02
...
Package name
hu.uszeged.ipcg.arithmtest02
Edit
Include C++ support
Include Kotlin support
Previous
Next
Cancel
Finish


Create New Project
Target Android Devices
Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.
Phone and Tablet
API 15: Android 4.0.3 (IceCreamSandwich)
By targeting API 15 and later, your app will run on approximately 100% of devices. Help me choose
Include Android Instant App support
Wear
API 21: Android 5.0 (Lollipop)
TV
API 21: Android 5.0 (Lollipop)
Android Auto
Android Things
API 24: Android 7.0 (Nougat)
Previous
Next
Cancel
Finish


Create New Project
Add an Activity to Mobile
Add No Activity
Basic Activity
Bottom Navigation Activity
Empty Activity
Ad
Previous
Next
Cancel
Finish


Create New Project
Configure Activity
Creates a new empty activity
Activity Name:
MainActivity
Generate Layout File
Layout Name:
activity_main
Backwards Compatibility (AppCompat)
The name of the activity class to create
Previous
Next
Cancel
Finish

Android Studio IDE



„Hello, World!”

- Alkalmazás futtatás
  - **`Run / Run ...`**
  - Vagy az eszközsoron listából projekt kiválasztása és a mellette lévő zöld háromszög kattintása
- Cél eszköz
  - Ha van csatlakoztatva készülék
    - Feltölti és indítja
  - Ha nincs készülék
    - Megfelelő AVD keresése és indítása
    - Program feltöltése és futtatása

Kezdő lépések

- Android fejlesztői portál
  - http://developer.android.com
  - https://developer.android.com/training/index.html
- Könyvek
  - Angol nyelven számtalan kiadvány
  - Magyarul: Android-alapú szoftverfejlesztés (BME)
    - Elég régi, kezd elavulni, bár az alapok hasonlók
- Tutorial
  - Learn How to Develop Android Application (CodeProject)
    - http://www.codeproject.com/Articles/628894/Learn-How-to-Develop-Android-Application

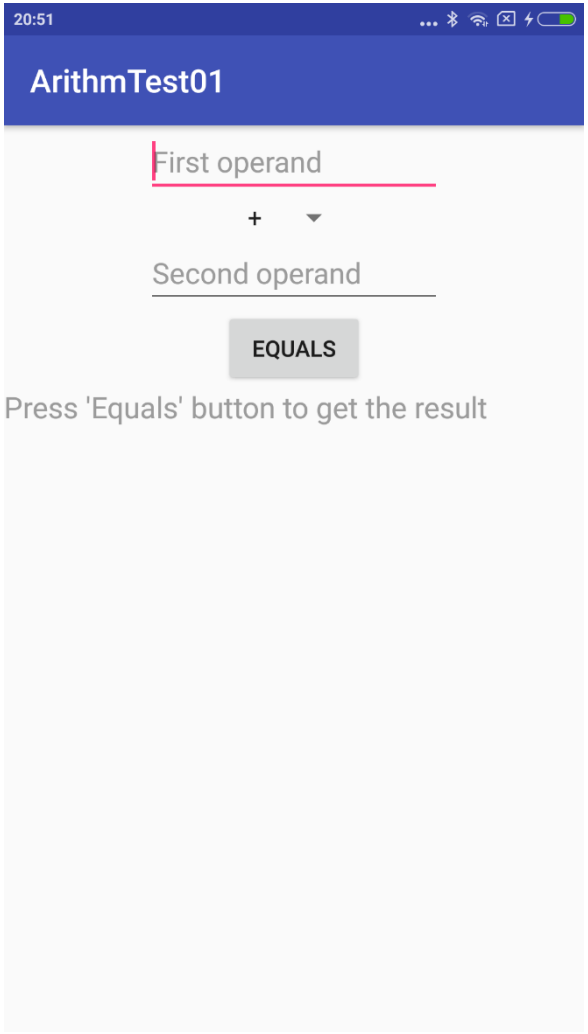
Fragmentáció

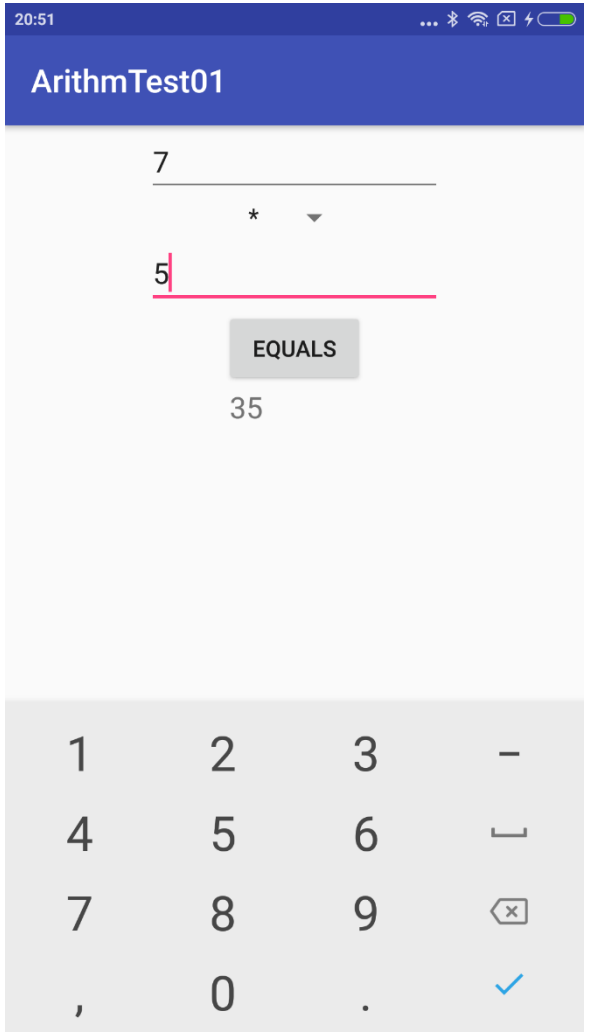
- Hardver
  - Rengeteg gyártó, rengeteg féle hardver konfiguráció
    - RAM, kijelző méret, hardver szenzorok, …
- Szoftver
  - Android verziók frissítése lassú (verziókról rövid leírás)
    - Google; gyártó; mobil kommunikációs cég
    - Flagship készülékek esetén kb. 18 hónap karbantartási idő
  - Android fork-ok (Amazon, Nokia, Cyanogen, Oxygen OS, …)
- Megoldási próbálkozások
  - Kompatibilitási könyvtár
  - Google Play Services egyre több funkciót átvesz
    - Új funkciók automatikusan megjelennek régebbi készülékeken is
    - A Google szorítani tud a túl nagy szabadságon…
  - Referencia eszközök
    - Nexus/Pixel készülékcsalád, gyártói Google Play Edition kiadások, …

ArithmTest01

- Fő funkciók
  - Egyszerű Android aktivitás alapvető aritmetikai műveletek elvégzésére
    - A művelet számítása a szerkesztőmezőkbe beírt számokon a **`spinner`**-ben kiválasztott műveletet hajtja végre és jeleníti meg szövegként.
    - A számítás nyomógombnyomásra történik.
    - Az alkalmazás üzenetei **`string`** erőforrásként definiáltak
  - Beépített vezérlőelemekkel
    - **`EditText`**, **`Spinner`**, **`TextView`**, **`Button`**
    - A vezérlőelemek XML fájlban definiáltak

ArithmTest01





ArithmTest bővítések

- **ArithmTest02**
  - Az alkalmazást kibővítjük úgy, hogy a szerkesztőmezőkben történő változások szerint azonnal frissül az eredmény.
    - A számítást kiszervezzük egy `refreshResult()` függvénybe. Ha mező tartalma érvénytelen, akkor `0` értéknek tekintjük.
    - `EditText.setOnEditorActionListener()` `onEditorAction()` függvénye: billentyűlenyomáskor frissítünk
    - `EditText.setOnFocusChangeListener()` `onFocusChange()` függvénye: fókusz elvesztésekor (másik vezérlő lesz aktív) frissítünk
    - `Spinner.onItemSelected()`: Spinner elemének kiválasztásakor eredmény frissítés
- **ArithmTest03**
  - Készítünk egy saját `View` leszármaztatást vízszintes vonal rajzolására: **MySeparatorView.java**

MOBIL KÉPELEMZÉS ÉS GRAFIKA

KAMERA HASZNÁLATI LEHETŐSÉGEK ANDROID PLATFORMON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

Android kamera használat

- Lehetőségek
  - Képek és videók készítése
  - Kamera élőkép feldolgozása
  - Akár több kamera is lehet (pl. előlapi és hátlapi)
- Eldöntendő
  - Mennyire fontos a kamera használat az alkalmazásban?
    - Ez a lényeg, e nélkül nincs értelme (pl. fotó alkalmazás).
    - Kiegészítő funkció, a lényeg nélküle is működik
  - Mennyire fontos a képkészítés paraméterezhetősége?
    - Elég egy külső fotó komponenst hívni?
    - Saját felületet kell készíteni?
  - Képek tárolása
    - Csak a saját alkalmazásunk érheti el a készült fotókat?
    - Vagy más is? (Pl. képgaléria.)

Kamera alkalmazás típusok

- Fotók készítése külső fotós komponens hívásával
  - `Intent` objektumon keresztül
    - Explicit: pontosan melyik komponens hívódjon
    - Implicit: a felhasználó választhat a rendszerben regisztrált, adott célt megvalósító komponensek közül
  - Előnyök
    - Egyszerű, nem kell kamera használati engedély, a felhasználó a megszokott felületet használhatja (pl. beépített applikáció)
  - Hátrány
    - Paraméterezhetőség nincs a programozóként a kezünkben (kép mérete, kép készítésének körülményei, ...)

Kamera alkalmazás típusok

- Saját fotós komponens készítése
  - `Camera`, `SurfaceView`, `SurfaceTexture`, `MediaRecorder`, `Intent` objektumok
  - **`Camera2`** API (újabb Android verziók)
    - **`CameraX`** API: egyszerűbb használat
  - Kép készítés; élőkép megjelenítés; videó felvétel; kép mentése
  - Előnyök
    - Teljes kontroll, bővebb paraméterkészlet, mint pl. a beépített alkalmazással
    - Kamera élőkép is feldolgozható
  - Hátrányok
    - Hardver használati engedélykérésre odafigyelni!
    - Komplexebb feladat, hardverfüggő részek tesztelése nehézkes
    - A gyártók általában nem adnak hozzáférést a manapság divatos multi-kamera megvalósítás minden eleméhez, csak a saját kamera szoftverük éri el teljes egészében

Külső komponens hívása

- **`Intent`** objektum összeállítása
  - **`MediaStore.ACTION_IMAGE_CAPTURE`** akció: képkészítés
    - **`MediaStore.EXTRA_OUTPUT`** mező: mentés helye (URI)
  - **`MediaStore.ACTION_VIDEO_CAPTURE`** akció: videó
    - **`MediaStore.EXTRA_OUTPUT`** mező: mentés helye (URI)
    - **`MediaStore.EXTRA_VIDEO_QUALITY`** mező: minőség (0 vagy 1)
    - **`MediaStore.EXTRA_DURATION_LIMIT`** : felvétel maximális hossza
    - **`MediaStore.EXTRA_SIZE_LIMIT`** : felvétel maximális mérete (bájtban)
- **`Intent`** aktiválása
  - **`startActivityForResult()`**
  - Alkalmazásunk háttérbe kerül, megjelenik a kért komponens
- **`Intent`** eredményének fogadása
  - **`onActivityResult()`** tagfüggvényünkben
  - Elkészült a kép, vagy megszakította a felhasználó a folyamatot
    - Vizsgálni kell, melyik következett be!

Intent objektum összeállítása (fotó)

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private Uri fileUri;

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

    // create Intent to take a picture and return control
    // to the calling application
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    // create a file to save the image
    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
    // set the image file name
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);

    // start the image capture Intent
    startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

Intent objektum összeállítása (videó)

```
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
private Uri fileUri;

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

    //create new Intent
    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

    // create a file to save the video
    fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO);
    // set the image file name
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
    // set the video image quality to high
    intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);

    // start the Video Capture Intent
    startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
}
```

Intent fogadása

- Az aktivitás 1 darab **`onActivityResult()`** függvénnyel rendelkezhet
  - Itt kell lekezelni, melyik **`Intent`** hívás tér vissza
    - **`requestCode`** egyedi legyen
  - Figyelni kell, hogy a felhasználó megszakította-e a folyamatot?
    - **`resultCode`**
  - A visszaadott adat az Intent objektum **`data`** mezejében lesz
    - Átadott URI esetén az eredmény helye
    - Nélküle egy kis méretű bélyegképet kapunk Bitmap objektumként
      - Gyártói implementációtól függ, mit tartalmaz, emiatt gyakorlatilag használhatatlan!
  - A kép/videó fájlból betölthető és felhasználható

Intent eredmény fogadása

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Image captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Image saved to:\n" +
                    data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the image capture
        } else {
            // Image capture failed, advise user
        }
    }

    if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Video captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Video saved to:\n" +
                    data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the video capture
        } else {
            // Video capture failed, advise user
        }
    }
}
```

Saját kamera alkalmazás

- Engedélyek kérése
  - `Manifest.xml` fájlban **KELL** megadni! (API 23 szint alatt)
  - Engedélykérés hiányában a program futása megszakad (force close)! És a hiba okát sem mondja meg a rendszer egyértelműen!
- Kamera detektálás és elérés
  - A kamera megosztott erőforrás! Csak akkor kérjük az elérését, ha az alkalmazásunk fut előtérben! Egy időben 1 alkalmazásé lehet!
- Előnézeti kép és képkészítés paramétereinek definiálása
  - Csak olyat válasszunk, amit a rendszer megvalósít! (Le kell kérni a listát.)
- Kamera előnézeti képhez leszármaztatás
  - `SurfaceView` osztályból; `SurfaceHolder.Callback` interfész
- Képkészítéshez kezelő függvény megadása
- Eredmény elmentése
  - Alkalmazás területre vagy (közös) galériába
- Kamera foglalás felszabadítása
  - Hibás vagy hiányzó felszabadítás esetén más alkalmazások nem érik el!
  - onPause() függvényben mindenképpen célszerű!

Engedélyek kérése

- Kamera hardver használati engedély

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Kamera képességek jelzése (megkövetelt vagy opcionális)

```
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="false" />
```

  - Megköveteltek hiánya esetén az alkalmazás nem jelenik meg a Google Play áruházban kereséskor/listázáskor (mert nincs értelme telepíteni)!

- Adattár használati engedély

```
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

  - Ha közös területre mentjük a képet

- Hangfelvétel készítésekor (pl. videó)

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

- GPS információ elhelyezése a képekhez

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Engedélyek kérése

- Android 6.0 felett (Marshmallow, API 23)
  - Változott az engedélykérés mechanizmusa!
  - A program használat közben kell ellenőrizze, van-e engedély a funkcióra
  - Ha nincs, akkor „menet közben” kell kérnie
  - Alapbeállításként tiltva van az elérés
- Megoldási lehetőségek
  - Építsük be a kódba az engedélyek futás közben ellenőrzését
    - Requesting Permissions at Run Time (Android Developer Portal)
  - 23-as API szint alatti verzióval fordítsuk
    - Hosszabb távon nem megoldás!
  - Tájékoztassuk a felhasználót, hogy a **`Beállítások / Alkalmazások / Alkalmazás neve`** kiválasztásával adjon engedélyt a funkciónak
    - Elég kényelmetlen a felhasználónak, hagyjuk inkább!

Engedélykérés (API 23)

```
private void requestCameraPermission() {
    if (FragmentCompat.shouldShowRequestPermissionRationale(this,
       Manifest.permission.CAMERA)) {
        new ConfirmationDialog().show(getChildFragmentManager(), FRAGMENT_DIALOG);
    } else {
        FragmentCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA},
                REQUEST_CAMERA_PERMISSION);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_CAMERA_PERMISSION) {
        if (grantResults.length != 1 || grantResults[0] !=
       PackageManager.PERMISSION_GRANTED) {
            ErrorDialog.newInstance(getString(R.string.request_permission))
                    .show(getChildFragmentManager(), FRAGMENT_DIALOG);
        }
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

Camera és Camera2 interfészek

- Camera (régi, elavult)
  - Egyszerűbb használat
  - Android 6.0 felett elavultnak lett nyilvánítva, hosszabb távon érdemes lesz a Camera2 objektumot használni!
  - Az új eszközök is támogatják egyelőre

Először ezzel foglakozunk!

- Camera2 (új API)
  - Több funkció, többféle megvalósítási szint
  - Gyártó specifikus, opcionális API részek!
  - Android 6.0 alatti eszközökön nem érhető el!

Megvalósítási szint	Jellemzők
LEGACY	Kompatibilitási mód, kevés támogatott funkcióval
LIMITED	Kb. a régi Camera interfésszel egyező funkcionalitás, de hatékonyabb és letisztultabb
EXTERNAL	Külső kamera modul, kevés támogatott funkcióval (~LIMITED)
FULL	Teljes Camera2 API támogatás
LEVEL3	Opcionális extra funkciók is elérhetők

Kamera hardver elérhetősége

- Ellenőrzés
  - Ha nem követeltük meg a kamera hardver meglétét
  - Ha nincs, ne próbáljuk használni!

```
/** Check if this device has a camera */
private boolean checkCameraHardware(Context context) {
    if (
context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)){
        // this device has a camera
        return true;
    } else {
        // no camera on this device
        return false;
    }
}
```

Kamera elérése

- Kamerák számának lekérése
  - Android 2.3 (API 9) felett
  - **`Camera.getNumberOfCameras()`**
- Kamera biztonságos elérése
  - Kivételkezelés kihagyhatatlan!
    - Nem fordul le, ha nincs try ... catch blokkban
  - **`Camera.open()`** alapértelmezett kamera elérése (~hátlapi)
  - **`Camera.open(int)`** több kamera esetén sorszám megadása

```
/** A safe way to get an instance of the Camera object. */
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open(); // attempt to get a Camera instance
    }
    catch (Exception e){
        // Camera is not available (in use or does not exist)
    }
    return c; // returns null if camera is unavailable
}
```

Kamera paraméterek

- **`Camera.getCameraInfo()`**
  - API 9 felett
  - Előlapi vagy hátlapi a kamera
  - Álló vagy fekvő helyzetű a kép

- **`Camera.getParameters()`**
  - **`Camera.Parameters`** objektumot ad vissza
  - Képformátum, vaku, fénymérés, fehéregyensúly, ...
  - Időugrásos felvétel (timelapse; API 11)
  - Kép mentése videófelvétel közben (API 14)
  - Fénymérési és fókusz területek (API 14)
  - Arcdetektálás (API 14)

Kamera paraméterek

- Kamera jellemzők lekérése
  - **`getSupported…()`**; **`isSupported…()`**; **`getMax…()`**; függvények

```
// get Camera parameters
Camera.Parameters params = mCamera.getParameters();

List<String> focusModes = params.getSupportedFocusModes();
if (focusModes.contains(Camera.Parameters.FOCUS_MODE_AUTO)) {
  // Autofocus mode is supported
}
```

Kamera paraméterek

- Kamera jellemzők beállítása
  - Paraméterek lekérése
  - Módosítása/beállítása
  - **`Camera.setParameters()`**

```
// get Camera parameters
Camera.Parameters params = mCamera.getParameters();
// set the focus mode
params.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);
// set Camera parameters
mCamera.setParameters(params);
```

Kamera élőnézeti előkép

- Lépések
  - Leszármaztatás a **`SurfaceView`** osztályból
  - **`SurfaceHolder.Callback`** interfész megvalósítása
  - Konstruktor
    - Callback beállítása; puffer típus megadása
  - **`public void surfaceCreated(SurfaceHolder holder)`**
    - Felület kész, kamerához hozzárendelés, élőkép elindítása
  - **`public void surfaceChanged(SurfaceHolder holder, int format, int w, int h)`**
    - Felület változik: létrejön, elfordul, mérete más lesz
    - Előnézet méretének optimális beállítása (ld. gyakorlaton)
    - Élőkép leállítása változtatás előtt!
  - **`public void surfaceDestroyed(SurfaceHolder holder)`**
    - Felület megszűnik

```
** A basic Camera preview class */
public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;

    public CameraPreview(Context context, Camera camera) {
        super(context);
        mCamera = camera;

        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        // deprecated setting, but required on Android versions prior to 3.0
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        // The Surface has been created, now tell the camera where to draw the preview.
        try {
            mCamera.setPreviewDisplay(holder);
            mCamera.startPreview();
        } catch (IOException e) {
            Log.d(TAG, "Error setting camera preview: " + e.getMessage());
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        // empty. Take care of releasing the Camera preview in your activity.
    }
    // ...
```

```
public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        // If your preview can change or rotate, take care of those events here.
        // Make sure to stop the preview before resizing or reformatting it.

        if (mHolder.getSurface() == null){
          // preview surface does not exist
          return;
        }
        // stop preview before making changes
        try {
            mCamera.stopPreview();
        } catch (Exception e){
          // ignore: tried to stop a non-existent preview
        }
        // set preview size and make any resize, rotate or
        // reformatting changes here

        // start preview with new settings
        try {
            mCamera.setPreviewDisplay(mHolder);
            mCamera.startPreview();

        } catch (Exception e){
            Log.d(TAG, "Error starting camera preview: " + e.getMessage());
        }
    }
```

Kamera élőkép elhelyezése UI-n

- Helyfoglalás FrameLayout elrendezésben

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
  <FrameLayout
    android:id="@+id/camera_preview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    />

  <Button
    android:id="@+id/button_capture"
    android:text="Capture"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    />
</LinearLayout>
```

Kamera élőkép elhelyezése UI-n

- Összekötés programból
  - Élőképet indítani célszerűbb **`onResume()`**-ban!

```
public class CameraActivity extends Activity {

    private Camera mCamera;
    private CameraPreview mPreview;

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

        // Create an instance of Camera; null pointer check missing!
        mCamera = getCameraInstance();

        // Create our Preview view and set it as the content of our activity.
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);
    }
}
```

Elrendezés iránya

- Fekvő helyzetű aktivitás megkövetelése
  - A készülék forgatása ne befolyásolja az élőképet
  - **`Manifest.xml`** fájlban

```
<activity android:name=".CameraActivity"
          android:label="@string/app_name"

          android:screenOrientation="landscape">
          <!-- configure this activity to use landscape orientation -->

          <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Előnézet iránya, teljes kijelző

- Android 2.2 (API 8) felett állítható
  - **`setDisplayOrientation()`**
- Forgatásra automatikusan
  - **`surfaceChanged()`** függvényben
    - Az előnézeti képet leállítva, a módosítás után újraindítva
- Teljes kijelzős üzemmód kérése onCreate() függvényben

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
this.getWindow().setFlags(WindowManager.LayoutPar
ams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

Kép készítése

- Kezelő objektum definiálása
  - **`PictureCallback`** osztály példányosítása
    - **`public void onPictureTaken(byte[] data, Camera camera)`** függvényének megvalósítása
- Képkészítés jellemzően UI esemény hatására
  - Pl. exponáló gomb lenyomása
  - Lehet idővezérelt is (pl. saját timelapse felvétel)
  - **`Camera.takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback postview, Camera.PictureCallback jpeg)`** hívása
  - A kamera élőkép automatikusan leáll
    - Újra kell **`startPreview()`**, ha folytatni akarjuk
  - RAW, postview nem minden eszközön támogatott!

PictureCallback objektum

```
private PictureCallback mPicture = new PictureCallback() {

    @Override
    public void onPictureTaken(byte[] data, Camera camera) {

        File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);
        if (pictureFile == null){
            Log.d(TAG, "Error creating media file, check storage permissions: " +
                e.getMessage());
            return;
        }

        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.close();
        } catch (FileNotFoundException e) {
            Log.d(TAG, "File not found: " + e.getMessage());
        } catch (IOException e) {
            Log.d(TAG, "Error accessing file: " + e.getMessage());
        }
    }
};
```

Indítás gombnyomásra

```
// Add a listener to the Capture button
Button captureButton = (Button) findViewById(id.button_capture);
captureButton.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // get an image from the camera (JPG)
            mCamera.takePicture(null, null, null, mPicture);
        }
    }
);
```

Videófelvétel készítése

- Lásd az Android fejlesztői portálon
  - http://developer.android.com/guide/topics/media/camera.html

Kamera foglalás felszabadítása

- Mikor?
  - Mihelyst nincs rá szükségünk
  - **`onPause()`** függvényben mindenképpen!
    - Hacsak nincs rá különösen nyomós okunk!

```
public class CameraActivity extends Activity {
    private Camera mCamera;
    private SurfaceView mPreview;
    private MediaRecorder mMediaRecorder;

    ...

    @Override
    protected void onPause() {
        super.onPause();
        releaseCamera();  // release the camera immediately on pause event
    }

    private void releaseCamera(){
        if( mCamera != null ) {
            mCamera.stopPreview(); // release the camera for other applications
            mCamera.setPreviewCallback( null );
            mPreview.removeCallback( mPreview );
            mCamera.release();
            mCamera = null;
        }
    }
}
```

Média fájlok mentése

- Javaslat
  - Célszerű SD kártyára menteni (Manifest engedély kell!), saját alkönyvtárba
  - Android 2.2 (API 8) felett
- Lehetőségek
  - Publikus kép könyvtárba
    - Fájlok az alkalmazás törlése után is megmaradnak

```
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
```

  - Alkalmazáshoz tartozó könyvtárba
    - Alkalmazás törlésekor törlődnek, viszont SD kártya tartalom olvasható és módosítható más alkalmazásoknak is!

```
Context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)
```

```
public static final int MEDIA_TYPE_IMAGE = 1;
public static final int MEDIA_TYPE_VIDEO = 2;

/** Create a file Uri for saving an image or video */
private static Uri getOutputMediaFileUri(int type){
      return Uri.fromFile(getOutputMediaFile(type));
}

/** Create a File for saving an image or video */
private static File getOutputMediaFile(int type){
    // To be safe, you should check that the SDCard is mounted
    // using Environment.getExternalStorageState() before doing this.
    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
              Environment.DIRECTORY_PICTURES), "MyCameraApp");
    // This location works best if you want the created images to be shared
    // between applications and persist after your app has been uninstalled.

    // Create the storage directory if it does not exist
    if (! mediaStorageDir.exists()){
        if (! mediaStorageDir.mkdirs()){
            Log.d("MyCameraApp", "failed to create directory");
            return null;
        }
    }
```

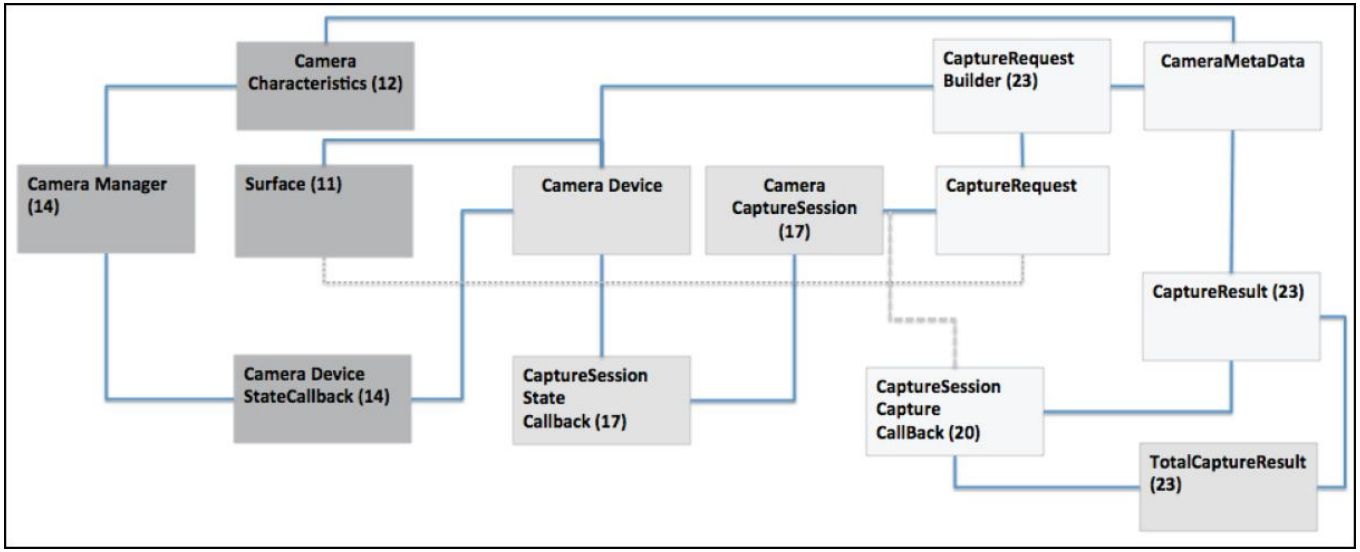
```
// Create a media file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    File mediaFile;
    if (type == MEDIA_TYPE_IMAGE){
        mediaFile = new File(mediaStorageDir.getPath() + File.separator +
        "IMG_"+ timeStamp + ".jpg");
    } else if(type == MEDIA_TYPE_VIDEO) {
        mediaFile = new File(mediaStorageDir.getPath() + File.separator +
        "VID_"+ timeStamp + ".mp4");
    } else {
        return null;
    }

    return mediaFile;
}
```

További információk

- Alapos leírás az Android Developer oldalon
  - http://developer.android.com/guide/topics/media/camera.html
- Open Camera
  - Ingyenes alkalmazás a Play áruházban
  - Forráskód elérhető (Git repozitórium)
    - http://opencamera.sourceforge.net/
    - http://sourceforge.net/projects/opencamera/
    - http://sourceforge.net/p/opencamera/code/ci/master/tree/

Camera2 API

- API 21 felett a Camera interfész elavultnak lett nyilvánítva!
  - Azért még elég sokáig támogatott lesz (nagy valószínűséggel).
- Új API: Camera2
  - Android Developer leírás
  - Google példaprogram (GitHub)
    - Kamera engedélykérésre is mutat példát



Camera2 API

- Android SDK példaprogramok
  - Android Studio: Import an Android Code Sample
  - Szűrjünk a Camera2 kifejezésre
    - Camera2Basic
    - Camera2Raw

CameraX API

- CameraX API
  - Camera2 interfész egyszerűbb használatára
  - https://developer.android.com/training/camerax
  - https://developer.android.com/jetpack
  - https://github.com/android/camera-samples/tree/master/CameraXBasic

Gyakorlat

- Példaprogramok
  - `04_01_CameraIntent.zip`
    - Fotókészítés implicit külső komponens hívással
    - Bélyegkép átvétele visszatéréskor
    - Aktivitás állapot mentése és visszaállítása
  - `04_02_CameraParameters.zip`
    - Kamera paraméterek lekérése és megjelenítése szöveges listában nyomógomb megnyomásával
    - Kamera használati engedély kérése
  - `04_03_CameraPreviewRatio.zip`
    - Kamera élőkép beállítása
  - Android SDK példaprogramok
    - Android Studio: `Import an Android Code Sample`
    - Szűrjünk a `Camera2` kifejezésre
      - `Camera2Basic`
      - `Camera2Raw`

Gyakorlat

- Feladatok
  - Bővítsük ki a `04_01_CameraIntent` alkalmazást úgy, hogy a külső kamera komponenstől a teljes képet vegyük át fájlon keresztül és ImageView-ban jelenítsük is meg!
  - Valósítsuk meg a `04_03_CameraPreviewRatio` példaprogram hiányzó nyomógombjainak funkcióit!
    - Kamera élőkép méretválasztások: képpontszám (kész), legjobb arány, széthúzás, natív méret
    - Képkészítés és mentés a publikus galériába

MOBIL KÉPELEMZÉS ÉS GRAFIKA

KAMERA ÉLŐKÉP FELDOLGOZÁS ANDROID PLATFORMON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Android kamera használat

- A régi Camera API felhasználásával
- Kamera lehetőségek
  - Képek és videók készítése/rögzítse
  - **Kamera élőkép feldolgozása**
  - Akár több kamera is lehet (pl. előlapi és hátlapi)
- Előnézeti élőkép jellemzői
  - A felhasználói felületen jelenik meg
  - Biztosítja a fotó megkomponálását
    - Kamera nézőpont, expozíciós beállítások hatása, …
    - 15, 30, … FPS
  - **Nem egyezik meg az elkészülő fotóval!**
    - Jóval kisebb képméret (választható többféle)
    - Beállítások hatása pontatlan
  - Bájtfolyamként megkaphatjuk a képkocka adatait

Camera.Parameters

- Előkép mérete
  - `Camera.Size getPreviewSize();`
  - `List<Camera.Sizes> getSupportedPreviewSizes();`
    - Csak ezek közül választhatunk!
- Frissítési frekvencia
  - `getPreviewFpsRange(int[]);`
    - Min és max értékek ezredmásodpercben
  - `List<int[]> getSupportedPreviewFpsRange();`
- Adatformátum
  - `int getPreviewFormat();`
  - `List<Integer> getSupportedPreviewFormats();`
- Előzőek `set` párjai
- Egyéb
  - `Camera.Size getPreferredPreviewSizeForVideo();`

Kamera előnézeti kép

- Előkészítés
  - `Camera.PreviewCallback` osztály példányosítása
  - `void onPreviewFrame (byte[] data, Camera camera)` függvény implementálása
- Csatlakozás a kamerához
  - `setPreviewCallback(Camera.PreviewCallback);`
    - Folyamatos élőkép feldolgozáshoz
  - `setOneShotPreviewCallback(Camera.PreviewCallback);`
    - Csak a következő képkocka feldolgozásához
  - `setPreviewCallbackWithBuffer(Camera.PreviewCallback);`
    - Hatékonyabb megvalósítás újra felhasználható pufferrel
    - `addCallbackBuffer(byte[])` kell előzetesen, lefoglalt memóriaterülettel
  - `SurfaceView` leszármazottjának `surfaceCreated` függvényében

Adatfolyam formátuma

- Alapértelmezésként YUV420SP (NV21 kódolás)
  - `Y`: szürkeárnyalat
  - `U`, `V`: színes komponensek
  - Adat struktúrája *w*×*h* méretű előnézeti kép esetén:
    - Első *w*·*h* darab bájt: `Y` komponens bájtos értéke
    - További *w*·*h*/2 darab bájt: felváltva `U` és `V` komponens bájtos értéke
      - 2×2 méretű blokkok alulmintavételezve
      - Színes komponensek kevésbé fontosak

Adatfolyam formátuma

- További formátum lehetőségek
  - `Camera.Parameters` segítségével
  - JPEG, NV16, NV21, RGB_565, UNKNOWN, YUV_420_888, YUV2, YV12
  - **NV21 kivételével nem mindegyik támogatott!**
    - Ellenőrizzük használat előtt, illetve inkább ne használjuk!

Konverzió szürkeárnyalatra

```
static public void decodeYUV420SPGr(int[] rgb, byte[] yuv420sp, int width, int height)
    {
          final int frameSize = width * height;

          for (int pix = 0; pix < frameSize; pix++)
          {
                    int pixVal = (0xff & ((int) yuv420sp[pix])) - 16;
                    if (pixVal < 0) pixVal = 0;
                    if (pixVal > 255) pixVal = 255;
                    rgb[pix] = 0xff000000 | (pixVal << 16) | (pixVal << 8) | pixVal;
          } // pix
    }
```

Figyeljük meg: A képpontok elérése egydimenziós tömbön keresztül történik!

Konverzió RGB-re

```
static public void decodeYUV420SP(int[] rgb, byte[] yuv420sp, int width, int height) {
        final int frameSize = width * height;

        for (int j = 0, yp = 0; j < height; j++) {
                int uvp = frameSize + (j >> 1) * width, u = 0, v = 0;
                for (int i = 0; i < width; i++, yp++) {
                        int y = (0xff & ((int) yuv420sp[yp])) - 16;
                        if (y < 0) y = 0;
                        if ((i & 1) == 0) {
                                v = (0xff & yuv420sp[uvp++]) - 128;
                                u = (0xff & yuv420sp[uvp++]) - 128;
                        }

                        int y1192 = 1192 * y;
                        int r = (y1192 + 1634 * v);
                        int g = (y1192 - 833 * v - 400 * u);
                        int b = (y1192 + 2066 * u);

                        if (r < 0) r = 0; else if (r > 262143) r = 262143;
                        if (g < 0) g = 0; else if (g > 262143) g = 262143;
                        if (b < 0) b = 0; else if (b > 262143) b = 262143;

                        rgb[yp] = 0xff000000 | ((r << 6) & 0xff0000) | ((g >> 2) &
0xff00) | ((b >> 10) & 0xff);
                }
        }
    }
```

Képkocka felhasználása

- Gyors műveletekre használjuk csak!
  - Ne akadozzon a kijelzőn az eredmény
  - Lassabb műveletek esetén tegyük új szálra a feldolgozást!
- Lehetőségek
  - Adatok kinyerése és kirajzolása
    - Hisztogram, statisztika
  - Detekció, felismerés
    - Területek, információk kiírása az előképre
- Alkalmazások
  - Kiterjesztett valóság (iOnRoad, WordLens, ...)
  - Képelemzés (Doctor Mole, ...)
  - Vizuális kódok (Barcode Scanner, ...)

Képi eredmények megjelenítése

- Jellemzően
  - Kamera élőkép felett elhelyezett UI elemeken (szöveges)
  - Kamera élőkép felett elhelyezett Bitmap képeken (képek)

- Bitmap objektummal összekötés
  - RGB adat formátuma
    - 32 bites (int típus), ARGB sorrendben
  - RGB adat beállítható Bitmap pixel adatként
    - `setPixels(int[] pixels, int offset, int stride, int x, int y, int width, int height);`

Vászon grafika

- Vászon grafika
  - 2D rajzterület (**`Canvas`** objektum)
    - Pl. **`View`** objektumok esetén **`onDraw()`** metódusban
    - Újrarajzolás „manuális” kezdeményezése: **`View.invalidate();`**
  - Elemi grafikus primitívek metódusként
    - Vonal, kör, ellipszis, szöveg, kép, alakzatok, …
  - Megjelenítési módok (**`Paint`** objektum)
    - Rajzolás, kitöltés, aliasing, szövegtípus, …

- Leírások
  - http://developer.android.com/guide/topics/graphics/2d-graphics.html
  - Hallgatói anyag
  - Android Wireless Application Development könyv

Vászon grafika

- **`View.onDraw()`** metódus
  - Jól illeszkedik a GUI fastruktúrájába
  - Frissítését a rendszer kezdeményezi
  - **Gyors változások (pl. játékprogramok) megjelenítésére nem igazán jó!**

- **`SurfaceView`** leszármaztatás
  - Minél gyorsabb újrarajzolás igénye esetén (pl. játékprogramok)
  - A játék vezérlése külön szálon (**`Thread`**) futhat
  - **`lockCanvas()`** és **`unlockCanvasAndPost()`** metódusok
  - Példa: Lunar Lander az SDK samples anyagában
  - Simple Java Android Game Loop

Munka Bitmap objektumokkal

- Probléma
  - A kicsomagolt képek nagy méretűek lehetnek
  - Az Android rendszer korlátozza a Java keretrendszer programjai számára a memóriahasználatot!
    - 5 MP kép 19MB!

Screen Size	Screen Density	Application Memory
small / normal / large	ldpi / mdpi	16MB
small / normal / large	tvdpi / hdpi	32MB
small / normal / large	xhdpi	64MB
xlarge	mdpi	32MB
xlarge	tvdpi / hdpi	64MB
xlarge	xhdpi	128MB

- Lehetséges megoldások
  - Használjunk különböző méretűeket
  - Glide függvénykönyvtár használata
    - https://github.com/bumptech/glide
  - Eleve kisebb méretben töltsük be
    - Displaying Bitmaps Efficiently
  - Használjunk natív C/C++ kódot a képfeldolgozáshoz
    - Fotós alkalmazások esetén szinte elkerülhetetlen
    - **`AndroidJniBitmapOperations-master.zip`**

Vászon grafika

- Példaprogramok
  - `05_01_ViewWithRedDot.zip`
    - Vörös, kitöltött kör rajzolása az aktivitás középső részére
    - Feladat: Próbáljunk ki többféle grafikus primitívet és rajzolási módot!
  - `05_02_CameraOverlay.zip`
    - Átlátszó hátterű vászon grafika kamera élőkép felett
    - Küszöbölés és Sobel operátor eredményének megjelenítése

CameraOverlay példaprogram

- Fő funkciói
  - Általunk megadott képkocka mérethez legközelebbi előnézet méret kiválasztása
  - Kamera élőkép átvétele a rendszertől és átadása a feldolgozó objektumnak
  - A feldolgozó objektum
    - Globális küszöbölést vagy Sobel gradiens számítást végez a képkockán
    - Az eredményt vászon grafikával megjeleníti

- Osztályok
  - MainActivity.java
  - CameraPreview.java
  - OverlayView.java

MainActivity.java

```
public class MainActivity extends Activity {
          private Camera mCamera;
          private CameraPreview mPreview;
          private OverlayView mOverlay;
          private TextView textPreviewRequestedSize;
          private TextView textPreviewSize;
          private TextView textPreviewDisplaySize;
          private FrameLayout preview;
    private Button btnThreshold;
    private Button btnSobel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }
```

Fő aktivitás változói és onCreate() függvénye. Teljes kijelzős megjelenítést kérünk.

MainActivity.java

```
@Override
   public void onResume() {
          super.onResume();

          // Resume after our camera has not been stopped.
          // E.g., after power off and on.
          if( mCamera != null )
                    return;

          // First run, or after camera has been stopped
          if( !checkCameraHardware( this ) ) {
                    setContentView(R.layout.activity_nocamera);
                    return;
          }

          mCamera = getCameraInstance();
          if( mCamera == null ) {
                    setContentView(R.layout.activity_noavailablecamera);
                    return;
          }

          setContentView(R.layout.activity_main);

          DisplayMetrics displaymetrics = new DisplayMetrics();
          getWindowManager().getDefaultDisplay().getMetrics(displaymetrics);
          int screenWidth = displaymetrics.widthPixels;
          int screenHeight = displaymetrics.heightPixels;

          preview = (FrameLayout) findViewById(R.id.camera_preview);
          LayoutParams layoutParameters = preview.getLayoutParams();
```

Kamera hardver elérésének ellenőrzése

és

GUI beállítás

Képernyőméret lekérdezése képpontban

MainActivity.java

```
  Camera.Parameters cameraParameters = mCamera.getParameters();
  List<Camera.Size> supportedPreviewSizes = cameraParameters.getSupportedPreviewSizes();
  int preview_width = 320;
  int preview_height = 240;
  int previewSize = preview_width * preview_height;
  Camera.Size bestPreviewSize = getOptimalPixelCountPreviewSize( supportedPreviewSizes,
previewSize );
  //Camera.Size bestPreviewSize = getOptimalSizeRatioPreviewSize( supportedPreviewSizes,
screenWidth, screenHeight );

  cameraParameters.setPreviewSize( bestPreviewSize.width, bestPreviewSize.height );
  mCamera.setParameters( cameraParameters );
```

A támogatott kamera élőkép méretek közüli választás előkészítése, a számító függvény meghívása, és kamera paraméter beállítása.

MainActivity.java

```
float previewRatio = ((float)bestPreviewSize.width) / ((float)bestPreviewSize.height);
float screenRatio = ((float)screenWidth) / ((float)screenHeight);
float ratio;
if( previewRatio > screenRatio ) {
          ratio = ((float)screenWidth) / ((float)bestPreviewSize.width);
} else {
          ratio = ((float)screenHeight) / ((float)bestPreviewSize.height);
}
layoutParameters.width = (int)(((float)bestPreviewSize.width) * ratio);
layoutParameters.height = (int)(((float)bestPreviewSize.height) * ratio);
preview.setLayoutParams(layoutParameters);

textPreviewRequestedSize = (TextView) findViewById( R.id.text_previewrequestedsize );
textPreviewRequestedSize.setText( "Required preview size: " + preview_width + "x" +
preview_height );

textPreviewSize = (TextView) findViewById( R.id.text_previewsize );
textPreviewSize.setText("Best fitting preview size: " + bestPreviewSize.width + "x" +
bestPreviewSize.height);

textPreviewDisplaySize = (TextView) findViewById( R.id.text_previewdisplaysize );
textPreviewDisplaySize.setText( "Preview display size: " + layoutParameters.width + "x" +
layoutParameters.height );
```

Kamera élőkép méretének nagyítási mértéke, hogy a teljes kijelzőt torzulásmentesen kitöltse. A méretek szövegdobozban sztringként való megjelenítése.

MainActivity.java

```
mOverlay = new OverlayView( this );
mOverlay.setViewRatio(ratio);

mPreview = new CameraPreview( this, mCamera, mOverlay );
preview.addView( mPreview );
preview.addView( mOverlay );

btnThreshold = (Button) findViewById( R.id.button_threshold );
btnThreshold.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mOverlay.selectThreshold();
    }
});

btnSobel = (Button) findViewById( R.id.button_sobel );
btnSobel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mOverlay.selectSobel();
    }
});
}
```

Overlay és Preview objektumok létrehozása a számított paraméterekkel.

Figyeljük meg, hogy a Preview megkapja az Overlay példányt, hogy a képkocka adatokat át tudja adni!

Küszöbölés és Sobel operátor nyomógombok „bekötése”.

MainActivity.java

```
@Override
   public void onPause() {
       super.onPause();

       if( mCamera != null ) {
               mPreview.setFinished(true);
               mCamera.stopPreview();
               mCamera.setPreviewCallback( null );
               mPreview.getHolder().removeCallback( mPreview );
               mCamera.release();
               mCamera = null;
       }
   }
```

Kamera élőkép leállítása és kamera objektum visszaadása a rendszernek.

Biztos ami biztos alapon. ☺

MainActivity.java

```
private boolean checkCameraHardware(Context context) {
        if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)){
            // this device has a camera
            return true;
        } else {
            // no camera on this device
            return false;
        }
    }

    public static Camera getCameraInstance(){
        Camera c = null;
        try {
            c = Camera.open(); // attempt to get a Camera instance
        }
        catch (Exception e){
            // Camera is not available (in use or does not exist)
        }
        return c; // returns null if camera is unavailable
    }
```

Kamera kezelő segédfüggények.

MainActivity.java

```
 private Camera.Size getOptimalPixelCountPreviewSize( List<Camera.Size> supportedPreviewSizes,
int previewSize ) {
        // Select preview size with closest pixel count
        Camera.Size bestPreviewSize = mCamera.new Size(0, 0);
        if( supportedPreviewSizes.size() > 0 )
            bestPreviewSize = supportedPreviewSizes.get( 0 );
        for( Camera.Size size : supportedPreviewSizes )
        {
            if( Math.abs( size.width * size.height - previewSize ) <
                Math.abs( bestPreviewSize.width * bestPreviewSize.height - previewSize) )
            {
                bestPreviewSize = size;
            }
        }

        return bestPreviewSize;
    }
```

Optimális kamera élőkép méret számítása: a lehetséges méretek közül azt választjuk, amely képpontszáma legközelebb van az általunk megadotthoz.

MainActivity.java

```
private Size getOptimalSizeRatioPreviewSize(List<Size> sizes, int w, int h) {

        final double ASPECT_TOLERANCE = 0.2;
        double targetRatio = (double) w / h;
        if (sizes == null)
            return null;

        Size optimalSize = null;
        double minDiff = Double.MAX_VALUE;
        int targetHeight = h;
        // Try to find an size match aspect ratio and size
        for (Size size : sizes) {
        double ratio = (double) size.width / size.height;
            if (Math.abs(ratio - targetRatio) > ASPECT_TOLERANCE)
                continue;
            if (Math.abs(size.height - targetHeight) < minDiff) {
                optimalSize = size;
                minDiff = Math.abs(size.height - targetHeight);
            }
        }

        // Cannot find the one match the aspect ratio, ignore the requirement
        if (optimalSize == null) {
            minDiff = Double.MAX_VALUE;
            for (Size size : sizes) {
                if (Math.abs(size.height - targetHeight) < minDiff)
                {
                    optimalSize = size;
                    minDiff = Math.abs(size.height - targetHeight);
                }
            }
        }
        return optimalSize;
    }
```

Másik lehetőség az optimális élőkép méret számítására: az általunk megadott sor és oszlopszám arányához legjobban közelítők közül azt választjuk, ahol a magasság a legközelebbi a megadotthoz.

CameraPreview.java

```
public class CameraPreview extends SurfaceView implements Callback {
    private static final String TAG = "CameraPreview";
    private SurfaceHolder mHolder;
    private Camera mCamera;
    private Activity parent;
    private OverlayView mDrawOnTopView;

    public boolean isFinished() {
        return mFinished;
    }

    public void setFinished(boolean mFinished) {
        this.mFinished = mFinished;
    }

    private boolean mFinished;
```

Kamera előnézet osztály változói és segédfüggvényei.

CameraPreview.java

```
public CameraPreview(Context context, Camera camera, OverlayView drawOnTop) {
        super(context);
        mCamera = camera;
        mDrawOnTopView = drawOnTop;
        mFinished = true;

        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = getHolder();
        mHolder.addCallback(this);
        // deprecated setting, but required on Android versions prior to 3.0
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

        parent = (Activity)context;
    }
```

Kamera előnézet osztály konstruktora.
Figyeljük meg az OverlayView objektum átvételét!

CameraPreview.java

```
@Override
   public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
       if (mHolder.getSurface() == null) {
           return;
       }

       // stop preview before making changes
       try {
               mCamera.stopPreview();
           } catch (Exception e)  // ignore: tried to stop a non-existent preview
       }
```

Kamera előnézetet felületváltozás kezelése.

CameraPreview.java

```
try {
    mCamera.setPreviewDisplay(mHolder);
    mCamera.setOneShotPreviewCallback(new Camera.PreviewCallback() {
        @Override
        public void onPreviewFrame(byte[] data, Camera camera) {
            if((mDrawOnTopView == null) || mFinished)
                return;

            Camera.Parameters params = camera.getParameters();
            // Initialize if necessary
            mDrawOnTopView.initializeBitmap( params.getPreviewSize().width,
                    params.getPreviewSize().height, data.length );
```

Egy képkocka fogadásának beállítása.
A beérkező adatból állapítjuk meg az élőkép folyam képkocka méretét, ezzel inicializáljuk a tömböt, amiben a képkockát tároljuk majd.

CameraPreview.java

```
    camera.setPreviewCallback(new Camera.PreviewCallback() {
        @Override
        public void onPreviewFrame(byte[] data, Camera camera) {
            if(mFinished) return;

            mDrawOnTopView.copyImageData( data ); // Pass YUV data to Overlay
            mDrawOnTopView.invalidate(); // Refresh view
        }
    }); } });
    mCamera.startPreview();
} catch (Exception e){
    Log.d(TAG, "Error starting camera preview: " + e.getMessage()); }}
```

Folyamatos élőkép elérés kérése.
A beérkező képi adatot átadjuk az OverlayView objektumnak, csináljon vele, amit tud. ☺

CameraPreview.java

```
 @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            mCamera.setPreviewDisplay(holder);
            mFinished = false;
        } catch (IOException e) {
            Log.d(TAG, "Error setting camera preview: " + e.getMessage());
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        // Preview will be stopped by the activity!
        mFinished = true;
    }
}
```

OverlayView.java

```
public class OverlayView extends View {
    private static final String TAG = "OverlayView";
    private Bitmap mBitmap;
    private int mImageWidth, mImageHeight;
    private byte[] mYUVData;
    private int[] mRGBData;
    private int mMethod;

    private Paint mPaint;
    private Paint mPaintResult;
    private Matrix bitmapmtx;
    private float mViewRatio;
    private float mViewRatioMultiplier;

    private final int METHOD_THRESHOLD = 0;
    private final int METHOD_SOBEL = 1;

    public float getViewRatio() {
        return mViewRatio;
    }

    public void setViewRatio(float mViewRatio) {
        this.mViewRatio = mViewRatio;
    }
```

Élőkép feldolgozó és eredmény megjelenítő osztály változói és segédfüggvényei.

OverlayView.java

```
public OverlayView( Context context ) {
        super( context );

        mViewRatio = 1.0f;
        mViewRatioMultiplier = 1.0f;
        bitmapmtx = new Matrix();

        mPaint = new Paint( Paint.ANTI_ALIAS_FLAG );
        mPaint.setColor(0x4400FF00); // Semi transparent green: AARRGGBB

        mPaintResult = new Paint();
        mPaintResult.setColor(Color.WHITE);

        mBitmap = null;
        mYUVData = null;
        mRGBData = null;
    }
```

Konstruktor: Paint objektum vászon grafikához, átlátszó háttérrel.

OverlayView.java

```
public void initializeBitmap( int width, int height, int dataLength ) {
        if( mBitmap == null ) { // Create a Bitmap
            mImageWidth = width;
            mImageHeight = height;
            mBitmap = Bitmap.createBitmap( mImageWidth, mImageHeight, Bitmap.Config.ARGB_8888 );
            mYUVData = new byte[ dataLength ]; // Incoming data from CameraPreview
            mRGBData = new int[ mImageWidth * mImageHeight ]; // RGB representation
        }
    }

    public void copyImageData( byte[] data ) {
        // CameraPreview calls this when a new frame is available
        System.arraycopy( data, 0, mYUVData, 0, data.length );
    }

    public void selectThreshold() {
        mViewRatioMultiplier = 1.0f;
        mMethod = METHOD_THRESHOLD;
    }

    public void selectSobel() {
        mViewRatioMultiplier = 0.6f;
        mMethod = METHOD_SOBEL;
    }
```

Bitmap inicializálás beérkező képmérethez és képkocka adat átmásolása.

Képművelet paraméterek átvétele.

OverlayView.java

```
@Override
   protected void onDraw( Canvas canvas ) {
       super.onDraw( canvas );

       if( mBitmap != null ) {
           // Convert from YUV to RGB Grayscale
           float ratio = mViewRatio * mViewRatioMultiplier;
           decodeYUV420SPGrayscale(mRGBData, mYUVData, mImageWidth, mImageHeight);
           // Run filter
           if( mMethod == METHOD_THRESHOLD ) {
               threshold( mRGBData, 128, mImageWidth, mImageHeight );
           }
           if( mMethod == METHOD_SOBEL ) {
               sobel( mRGBData, mImageWidth, mImageHeight );
               //sobel2(mRGBData, mImageWidth, mImageHeight);
           }
           // Copy result to Bitmap to display
           mBitmap.setPixels(mRGBData, 0, mImageWidth, 0, 0, mImageWidth, mImageHeight);
           // Display result in given size
           bitmapmtx.reset();
           bitmapmtx.setScale( ratio, ratio, 0.0f, 0.0f); // Full view
           // Draw scaled Bitmap to canvas
           canvas.drawBitmap(mBitmap, bitmapmtx, mPaintResult );
       }

       // Draw a transparent green circle in the center on top of everything else
       int cx = canvas.getWidth() / 2;
       int cy = canvas.getHeight() / 2;
       float radius = (float)cy / 10.0f;
       canvas.drawCircle( cx, cy, radius, mPaint );
   }
```

Képkocka feldolgozás hívása és eredmény vászonra rajzolása megadott nagyítással.

OverlayView.java

```
static public void threshold( int[] rgb, int threshold, int width, int height ) {
        int pos = 0;
        for ( int j = 0; j < height; j++ ) {
            for ( int i = 0; i < width; i++, pos++ ) {
                if ( ( rgb[pos] & 0xFF ) > threshold ) {
                    //rgb[pos] = 0xFFFFFFFF; // Opaque white
                    rgb[pos] = 0x40FF0000; // Transparent red
                } else {
                    //rgb[pos] = 0xFF000000; // Opaque black
                    rgb[pos] = Color.TRANSPARENT; // Fully transparent
                }
            }
        }
    }
```

Küszöbölés megvalósítása. A küszöbértéknél nagyobb szürkeségi értéknél áttetsző vörös, egyébként teljesen átlátszó színt adunk meg.

Feladatok

- Vászon grafika
  - Készítsünk táblát/játékteret az alábbi játékokhoz:
    - Amőba
    - Malom
    - Sakk
  - Egyszerű árkád játék megvalósítása *game loop* technikával

Feladatok

- Egyszerű képfeldolgozó műveletek megvalósítása kamera élőképen
  - További gradiens operátorok
  - Adaptív küszöbölés, Niblack
  - Komponensdetektálás
  - Szűrések, élesítés
  - Hisztogram műveletek (kiegyenlítés)
  - …
- Eredmények képi megjelenítése
  - Kamera keret rajzolása az élőképre
  - Küszöbölt komponensek átlátszó módon élőképre vetítve
  - Küszöbölt komponensek kontúrjának élőképre vetítése

MOBIL KÉPELEMZÉS ÉS GRAFIKA

OPENCV ANDROID PLATFORMON (JAVA+JNI)

JNI+BITMAP KEZELÉS

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

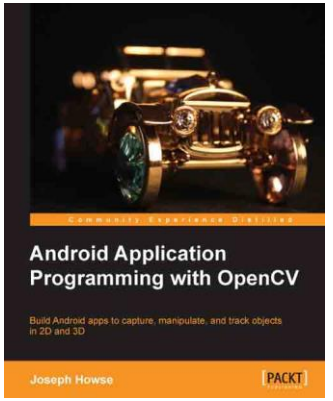
BEFEKTETÉS A JÖVŐBE

OpenCV

- Nyílt forráskódú szoftver (BSD licensz)
  - Számítógépes látás
  - Gépi tanulás
- >2500 optimalizált algoritmus
  - Szűrések, élkeresés, hisztogram, Hough-transzformáció, …
  - Arcdetekciós és –felismerés, mozgásosztályozás, objektumok követése, panorámakép készítés, képtartalom-alapú keresés, …
- Többféle támogatott platform és programozási nyelv
  - >15 év fejlesztés, számos közreműködő
  - C++ sablon (template) interfész
  - Python, Matlab, Java kötések
  - Windows, Linux, Mac OS, Android, iOS verziók

Anyagok

- Hivatalos honlap és letöltés
  - https://opencv.org/ (Általános)
  - https://opencv.org/android/ (Android)
- Dokumentáció
  - http://docs.opencv.org/
  - Hivatalos tutorialok
- Könyv
  - Android Application Programming with OpenCV
    - Jó, csak kissé elavult
    - http://nummist.com/opencv/default.aspx
  - C++ vagy Java OpenCV anyag is megfelelő



Android OpenCV

- Android Java API reference
  - http://docs.opencv.org/ oldalon a Java részt keressük
  - Android specifikus dolgok
    - `CameraBridgeViewBase`: kamera élőképpel összekötés
    - OpenCV loader: OpenCV lib-ek csatolása külső programon keresztül (OpenCV Manager) – 2018. őszén megszűnt!
  - Java mellett natív (C, C++) kódból is használható az OpenCV
- Példaprogramok üzembe állítása (Eclipse)

Elavult!

  - http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/O4A_SDK.html
- Android fejlesztési leírás (Eclipse)
  - http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html#dev-with-ocv-on-android

OpenCV Android Studio-val

- **Töltsük le az Android OpenCV csomagot és csomagoljuk ki**
  - https://opencv.org/releases.html
  - Android Pack kell
  - Csomagoljuk ki (majdnem) tetszőleges helyre
    - **Az elérési útvonal neve nem tartalmazhat szóközt és speciális karaktert!**
  - Célszerű átnevezni a mappát, hogy tükrözze az OpenCV verziószámot
    - OpenCV-android-sdk -> OpenCV-4.2.0-android-sdk

- **Megjegyzés**
  - A példaprogramjainkat tartalmazó OpenCV420Initfull csomag tartalmazza az OpenCV 4.2.0 fájlokat is, így külön nem muszáj letölteni

OpenCV Android Studio-val

- Nyissunk meg egy projektet (vagy kezdjünk újat)
  - Az OpenCV 4.2.0 verzió esetén legalább 21-es API szint kell
- Importáljuk az OpenCV-t modulként
  - `File / New / Import Module`
  - Forrás mappa megadásként kétféle lehetőségünk van:
    - **`sdk`** mappát adjuk meg az OpenCV mappán belül: előnye, hogy nem kell lib elérhetőségekkel bajlódnunk, viszont kb. 1 GB (!!!) OpenCV anyagot bemásol a projekt mappánkba! (Fordítás után több GB méretre számíthatunk…)
    - **`sdk/java`** mappát adjuk meg: elegendő a **`native/libs`** tartalmat később „kézzel” átmásolni
  - `Module name` átírása, pl. `opencv420sdk` vagy `opencv420java`
    - A régi OpenCV verziók értelmes modul nevet adtak alapból, az újak nem (sdk vagy java, a mappa neve alapján)

OpenCV Android Studio-val

- Adjunk hozzá modul függőséget
  - Válasszuk ki a modulunkat, amelyben OpenCV-t használunk (alapértelmezetten `app` a neve)
  - `F4` vagy jobb egérgomb / `Open Module Settings`
  - `Dependencies` fül
    - `+` jel a `Declared Dependencies` panel bal felső részén
    - `Module dependency`
    - A listából válasszuk ki az OpenCV modult (pl. `opencv420sdk`)

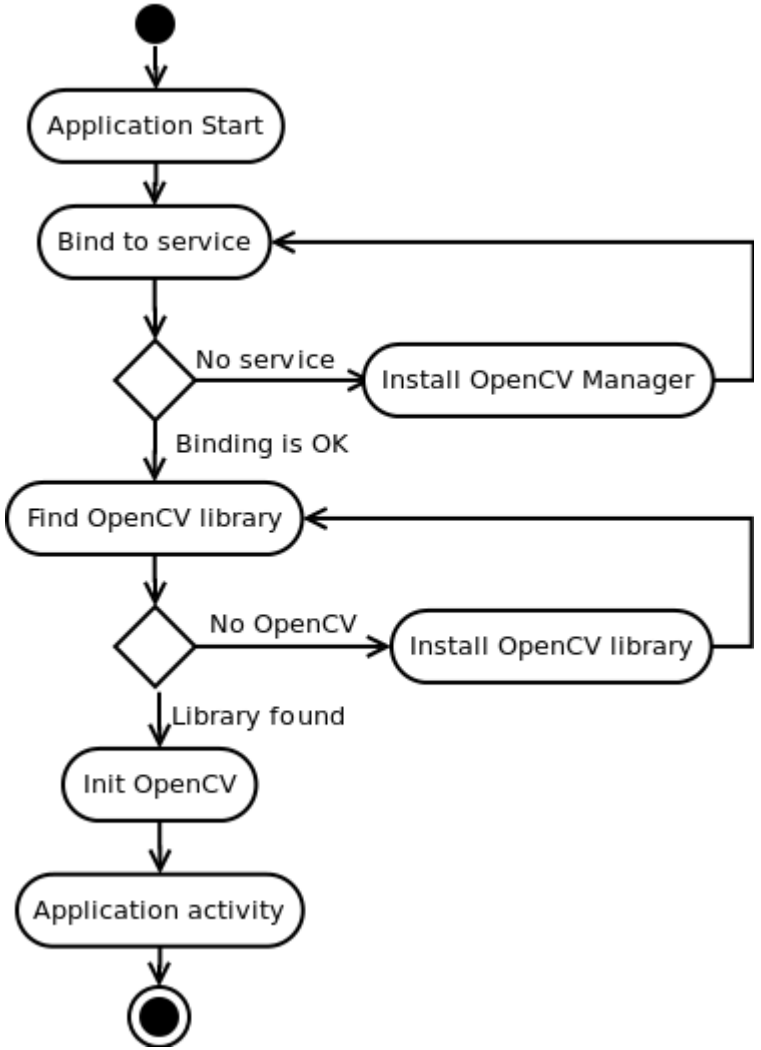
- Ha nem jelenik meg az Opencv modul a függőségeknél
  - `build.gradle` (`Module: opencv420java`) szkriptben átírni:
    - `apply plugin: 'com.android.library'`
    - `// applicationId "org.opencv"`

OpenCV Manager

- Probléma
  - Nagy méretű Android alkalmazások, ha minden alkalmazás saját OpenCV lib-et kap
- (Korábbi) megoldás
  - Különálló alkalmazás (Google Play áruházból letölthető)
  - Megfelelő OpenCV verzió biztosítása
    - A platformnak legjobban megfelelő bináris kód
    - Pl. Tegra platform speciális utasításkészletének kihasználása
  - OpenCV lib-ek csak 1 példányban, ezen „osztoznak” az alkalmazások
  - 2018. őszén visszavonták, nem érhető el a Play Áruházban!
- Jelenlegi lehetőség
  - Minden modulhoz bemásolódnak az OpenCV binárisok
  - Jóval nagyobb alkalmazás méret! (40-50 MB minimum)

OpenCV Manager

- Induláskori feladatok
  - OpenCV Manager elérhetőségének ellenőrzése
    - Ha nincs, telepítés felajánlása
  - Megfelelő verziójú OpenCV lib keresése
    - Ha nincs, telepítés felajánlása
  - Inicializálás



OpenCV Manager

```
public class MyActivity extends Activity implements HelperCallbackInterface
{
            private BaseLoaderCallback mOpenCVCallBack = new BaseLoaderCallback(this) {
               @Override
               public void onManagerConnected(int status) {
                 switch (status) {
                   case LoaderCallbackInterface.SUCCESS:
                   {
                      Log.i(TAG, "OpenCV loaded successfully");
                      // Create and set View
                      mView = new puzzle15View(mAppContext);
                      setContentView(mView);
                   } break;
                   default:
                   {
                      super.onManagerConnected(status);
                   } break;
                 }
               }
            };

            /** Call on every application resume **/
            @Override
            protected void onResume()
            {
                Log.i(TAG, "Called onResume");
                super.onResume();

                Log.i(TAG, "Trying to load OpenCV library");
                if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_11, this, mOpenCVCallBack))
                {
                    Log.e(TAG, "Cannot connect to OpenCV Manager");
                }
}
```

OpenCV binárisok modulhoz adása

- `sdk` importálás esetén
  - Nincs „kézzel” másolandó anyag

- `sdk/java` importálás esetén
  - Adjunk a modulhoz új JNI mappát
    - Jobb egérkattintás a modul nevén (`app`)
    - `New / Folder / JNI Folder`
    - `Change Folder Location` bekattintása
    - `src/main/jni/` helyett `src/main/jniLibs/`
  - Másoljuk be ebbe a mappába az OpenCV bináris lib-eket
    - `<OpenCV-android-sdk>/sdk/native/libs` almappáit
    - A teljes mappákat kell, a mappa nevek az egyes cél architektúrákat írják le

Tesztelés #1

- Adjuk hozzá az alábbi sort az `onCreate()` függvény végéhez
  - `Log.d("verify", String.valueOf(OpenCVLoader.initDebug()));`
- Kellenek még az alábbi import-ok
  - `import android.util.Log;`
  - `import org.opencv.android.OpenCVLoader;`
- Kapcsoljuk be a LogCat ablakot
  - `Alt + 6`
- Futtassuk a programot
  - A részletes library betöltési információk a LogCat ablakban láthatók

```
Trying to get library list
OpenCV error: Cannot load info library for OpenCV
Library list: ""
First attempt to load libs
Trying to init OpenCV libs
Trying to load library opencv_java4
Library opencv_java4 loaded
First attempt to load libs is OK
General configuration for OpenCV 4.0.1 =====================================
  Version control:               4.0.1
  Platform:
    Timestamp:                   2018-12-22T07:41:15Z
    Host:                        Linux 4.15.0-42-generic x86_64
    Target:                      Android 1 aarch64
    CMake:                       3.6.0-rc2
    CMake generator:             Ninja
    CMake build tool:            /opt/android/android-sdk.gradle/cmake/3.6.4111459/bin/ninja
    Configuration:               Release
  CPU/HW features:
    Baseline:                    NEON FP16
  C/C++:
    Built as dynamic libs?:      NO
…
  OpenCV modules:
    To be built:                 calib3d core dnn features2d flann highgui imgcodecs imgproc java java_bindings_generator ml objdetect photo stitching video videoio
    Disabled:                    world
    Disabled by dependency:      -
    Unavailable:                 gapi js python2 python3 ts
    Applications:                -
    Documentation:               NO
    Non-free algorithms:         NO
  Android NDK:                   /opt/android/android-ndk-r18b (ver 18.1.5063045)
    Android ABI:                 arm64-v8a
    NDK toolchain:               aarch64-linux-android-clang
    STL type:                    c++_static
    Native API level:            21
  Android SDK:                   /opt/android/android-sdk.gradle (tools: 26.1.1 build tools: 28.0.3)
  GUI:
  Media I/O:
    ZLib:                        z (ver 1.2.7)
    JPEG:                        build-libjpeg-turbo (ver 1.5.3-62)
    WEBP:                        build (ver encoder: 0x020e)
    PNG:                         build (ver 1.6.35)
    TIFF:                        build (ver 42 - 4.0.9)
    JPEG 2000:                   build (ver 1.900.1)
    OpenEXR:                     build (ver 1.7.1)
    HDR:                         YES
    SUNRASTER:                   YES
    PXM:                         YES
    PFM:                         YES
  Video I/O:
  Parallel framework:            TBB (ver 2018.0 interface 10001)
  Trace:                         YES (built-in)
  Other third-party libraries:
    Custom HAL:                  YES (carotene (ver 0.0.1))
    Protobuf:                    build (3.5.1)
  Python (for build):            /usr/bin/python2.7
  Java:                          export all functions
    ant:                         NO
    Java wrappers:               YES
    Java tests:                  NO
  Install to:                    /build/master_pack-android/build/o4a/install
-----------------------------------------------------------------
2019-03-12 08:06:30.634 29486-29486/hu.uszeged.ipcg.opencv4test D/verify: true
```

Tesztelés #2

- OpenCV420InitFull / app
  - Az inicializálás eredményét egy `TextView` objektumban mutatja meg
  - Layout XML fájlba:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:id = "@+id/tv"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Aktivitás

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

    TextView tv = (TextView)findViewById(R.id.tv);

    if(OpenCVLoader.initDebug()) {
        tv.setText("OpenCV loaded successfully!");
        Log.d("verify", "OpenCV loaded successfully!");
    } else {
        tv.setText("Cannot load OpenCV!");
        Log.d("verify", "Cannot load OpenCV!");
    }
}
```

Tesztelés

- Ha „`cannot find symbol variable styleable`” hibát kapunk fordításkor (pl. OpenCV 4.0.1)
  - `Gradle.build (Module: opencv401java)` fájlban a `sourceSets`-ben módosítsuk a `res` bejegyzést: `res.srcDirs = ['res']`
  - OpenCV 4.1.1 verzióban javítva lett

OpenCV Java API csomagok

- `Android`
  - Betöltő kód, kamera élőkép interfészek, Java és natív kamera elérés, FPS számláló
- `Core`
  - Alap osztályok (`Mat`, `Point`, ...)
  - Típusok (`CvTypes`)
  - Mátrix műveltek, rajzolások
- `Imgproc`
  - Képfeldolgozó műveletek
- `Utils`
  - Típusok közötti konverterek
- `HighGUI`
  - Kép beolvasó és kiíró függvények
  - Videó rögzítés vagy fájlból beolvasás
  - Ablakozást és interakciót az Android elemekkel kell végezni!
- `Features2d`
  - Kulcspont detektorok és leírók
- `ML`
  - Gépi tanulás (Machine Learning)
- `Contrib`
  - Külső fejlesztők algoritmusai + szabadalommal védettek
  - Nem része alapból a csomagnak!

OpenCV Java API használat

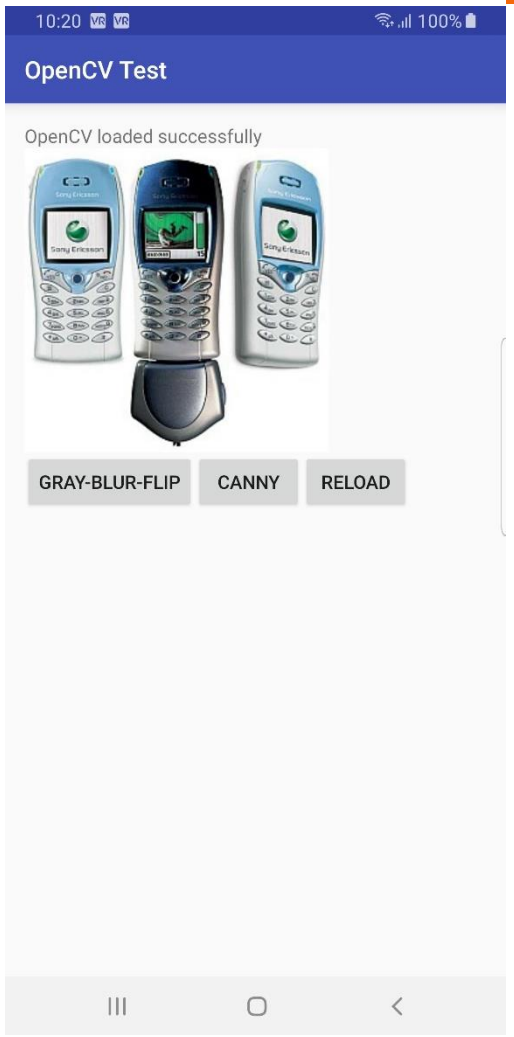
- Különbségek C/C++-hoz képest
  - Névtér helyett csomagok
  - Nincsenek operátor felüldefiniálások!
    - Több hatékony megvalósítás nem érhető el
  - Más absztrakt adattípusokba kerül az eredmény
    - Listák, vektorok, …

Példaprogramok

- `OpenCV420InitFull` csomagban
  - `App`
    - OpenCV inicializálás
  - `OpenCVTester`
    - OpenCV képfeldolgozás erőforrásból betöltött képen
    - Eredmény `ImageView` objektumban megjelenítve
    - Funkciók nyomógombbal érhetők el

OpenCVTester UI elrendezés

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/imageView1"
        android:text="Gray-Blur-Flip" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button1"
        android:layout_toRightOf="@+id/button1"
        android:text="Canny" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button2"
        android:layout_toRightOf="@+id/button2"
        android:text="Reload" />

</RelativeLayout>
```



```
public class OpenCVTestMainActivity extends AppCompatActivity {
    private static final String TAG = OpenCVTestMainActivity.class.getSimpleName();
    ImageView iv;
    TextView tv;
    Bitmap bm;
    Button btn1, btn2, btn3;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_opencv_main);
        bm = BitmapFactory.decodeResource(getResources(), R.drawable.t68ikamera);
        iv = (ImageView)findViewById(R.id.imageView1);
        iv.setImageBitmap( bm );
        tv = (TextView)findViewById(R.id.textView1);
        btn1 = (Button)findViewById(R.id.button1);
        btn1.setOnClickListener(mBlurFlipBtnListener);
        btn2 = (Button)findViewById(R.id.button2);
        btn2.setOnClickListener(mEdgeBtnListener);
        btn3 = (Button)findViewById(R.id.button3);
        btn3.setOnClickListener(mReloadBtnListener);
    }

    @Override
    protected void onResume() {
        super.onResume();
        if(OpenCVLoader.initDebug()) {
            Log.d(TAG, "OpenCV loaded successfully");
            tv.setText( "OpenCV loaded successfully" );
        } else {
            this.finish();
        }
    }
```

Függvényhívások nyomógombokra kötése

OpenCV modul betöltése

Nyomógomb funkciók

```
private OnClickListener mBlurFlipBtnListener = new OnClickListener() {
    public void onClick(View v) {
        tv.setText("BlurFlip button is pressed...");
        Mat im = new Mat();
        Utils.bitmapToMat( bm, im );
        Imgproc.cvtColor( im, im, Imgproc.COLOR_RGB2GRAY );
        Imgproc.blur( im, im, new Size(5, 5) );
        Core.flip( im, im, 1 );
        Utils.matToBitmap( im, bm );
        iv.setImageBitmap( bm );
    }
};

private OnClickListener mEdgeBtnListener = new OnClickListener() {
    public void onClick(View v) {
        tv.setText("Edge button is pressed...");
        Mat im = new Mat();
        Utils.bitmapToMat(bm, im);
        Mat edges = new Mat();
        Imgproc.Canny( im, edges, 50, 200, 3, true );
        Utils.matToBitmap( edges, bm );
        iv.setImageBitmap( bm );
    }
};

private OnClickListener mReloadBtnListener = new OnClickListener() {
    public void onClick(View v) {
        tv.setText("Reload button is pressed...");
        bm = BitmapFactory.decodeResource(getResources(), R.drawable.t68ikamera);
        iv.setImageBitmap( bm );
    }
};
```

Konverzió Android `Bitmap` és OpenCV `Mat` között, OpenCV képfeldolgozással.

OpenCV kamera élőkép elérése

- Az aktivitásunkat az OpenCV `CameraActivity` osztályából származtassuk
  - Ekkor nem kell a kamera engedélykéréssel sem foglalkoznunk
- Implementáljuk a `CameraBridgeViewBase.CvCameraViewListener2` interfész függvényeit
  - `void onCameraViewStarted(int width, int height)`
  - `void onCameraViewStopped()`
  - `Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame)`
- Bár nem kötelező, de ha nem definiáljuk felül, akkor nem lesz kamera élőkép
  - `getCameraViewList()`
- Helyezzünk el egy `JavaCameraView` objektumot az UI-n
  - Csatlakozzunk a létrejövő objektumhoz *listener*-ként az `onCreate()`-ben
  - Ha sikeres az OpenCV inicializálás, akkor indítsuk el az élőképet
- Ne felejtsük el a `Manifest.xml` módosítását
  - Kamera engedélyek, landscape megjelenítés, fejléc nélküli téma

Példaprogramok

- `OpenCV420InitFull / OpenCVLiveView`
  - Otsu küszöbölés kamera élőképen
- `OpenCV420InitFull / OpenCVLiveThreshold`
  - Összetettebb példaprogram
  - Csúszkával állítható küszöbérték
  - Nyomógombbal választható módszer
- `OpenCV420InitFull / color-blob-sample`
  - Az OpenCV példaprogramja átemelve saját projektünkbe
  - Képernyőérintést is kezel
  - A „megérintett” pont környékéről színinformációt gyűjt, és az alapján végez színes küszöbölést, aminek az határát kirajzolja az élőképre

OpenCVLiveView Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hu.uszeged.ipcg.opencvliveview">

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera" android:required="true"/>
    <uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
    <uses-feature android:name="android.hardware.camera.front" android:required="false"/>
    <uses-feature android:name="android.hardware.camera.front.autofocus" android:required="false"/>

    <supports-screens android:resizeable="true"
        android:smallScreens="true"
        android:normalScreens="true"
        android:largeScreens="true"
        android:anyDensity="true" />

    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen">
        <activity android:name=".OpenCVLiveViewMainActivity"
            android:screenOrientation="landscape"
            android:configChanges="keyboardHidden|orientation">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

OpenCVLiveView layout.xml

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

    <org.opencv.android.JavaCameraView
        android:id="@+id/cameraView"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="visible" />

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello"
        android:layout_alignParentTop="true"
        android:background="#CCFF4081" />

</RelativeLayout>
```

MainActivity import-ok

```
package hu.uszeged.ipcg.opencvliveview;

import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceView;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;

import org.opencv.android.CameraActivity;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;

import java.util.Collections;
import java.util.List;
```

Osztály fejléc + adattagok

```
public class OpenCVLiveViewMainActivity extends CameraActivity implements
    CameraBridgeViewBase.CvCameraViewListener2 {
    private CameraActivity mActivity = null;
    private TextView tv1;
    private CameraBridgeViewBase mOpenCvCameraView;
    private static final String TAG = "OpenCVLiveView";
    private int threshValue = 128;
    private Mat grayMat;
    // private Mat rgbMat;
```

onCreate()

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_opencv_live_view_main);
    tv1 = (TextView) findViewById( R.id.tv1 );
    mActivity = this;

    mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.cameraView);
    mOpenCvCameraView.setMaxFrameSize( 640, 480 );
    mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
    mOpenCvCameraView.setCvCameraViewListener(this);
}
```

A kamera élőkép méretének a kívánt maximális méretét adhatjuk meg. Más, közvetlen beállítási lehetőségünk erre vonatkozóan nincs. Ezen túl a kameranézet objektumhoz csatlakozunk megfigyelőként.

```java
@Override
public void onResume()
{
    super.onResume();
    if(OpenCVLoader.initDebug()) {
        mOpenCvCameraView.enableView();
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        Log.d("enabled view", "enabled");
        tv1.setText("OpenCV loaded - initializing");
    } else {
        // Cannot load OpenCV!
        mActivity.finish();
    }
}

@Override
public void onPause() {
    super.onPause();
    if(mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}

public void onDestroy() {
    super.onDestroy();
    if (mOpenCvCameraView != null)
        mOpenCvCameraView.disableView();
}
```

Sikeres OpenCV betöltés esetén elindítjuk a kamera élőképet.

Az alkalmazásunk háttérbe kerülésekor leállítjuk az élőkép feldolgozást.

Interfész függvények

Ha ez nincs, akkor nem indul el a kamera élőkép...

```
@Override
protected List<? extends CameraBridgeViewBase> getCameraViewList() {
    return Collections.singletonList(mOpenCvCameraView);
}
```

OpenCV képmátrix a kamera élőkép fogadására.

```
@Override
public void onCameraViewStarted(int width, int height) {
    grayMat = new Mat( height, width, CvType.CV_8UC1 );
    // rgbMat = new Mat( height, width, CvType.CV_8UC4 );
}
```

Felszabadítjuk a képmátrixot, ha leáll a kamerakép.

```
@Override
public void onCameraViewStopped() {
    grayMat.release();
    // rgbMat.release();
}
```

Előkép feldolgozása

```
@Override
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    grayMat = inputFrame.gray();
    // rgbMat = inputFrame.rgba();

    double th = Imgproc.threshold(grayMat, grayMat, 0, 255,
                        Imgproc.THRESH_BINARY + Imgproc.THRESH_OTSU );
    threshValue = (int) th;
    final int ith = (int) th;

    runOnUiThread(new Runnable() {
        public void run() {
            tv1.setText(String.format("Computed Otsu Threshold: %1$d",
                        threshValue));
        }
    });

    return grayMat;
    // return rgbMat;
}
```

Kamera élőkép képkocka átvétele szürke (// vagy színes) formában.

Otsu küszöbölés OpenCV függvénnyel, Java oldalról.

Információ TextView-ban. A feldolgozó függvény külön szálon fut, így a módosítást a GUI szálon kell indítani.

Eredmény képmátrix visszaadása. Ez fog élőképként megjelenni.

AndroidOpenCVGradlePlugin

- Egy újabb lehetőség az OpenCV projektünkhöz adására
  - https://github.com/ahasbini/AndroidOpenCVGradlePlugin
  - Az oldal alján leírásra kerül a használata
  - Tartozik hozzá példa projekt (`sample`)
  - Viszont nagyon nehézkesen működik egyelőre
- Egy másik részletes leírás a lehetséges lépésekről
  - https://stackoverflow.com/questions/43766092/adding-opencv-to-native-c-code-through-cmake-on-android-studio

Feladatok

- Paraméterezhető képfeldolgozó operátorok
  - Kamera élőképen vagy erőforrásból betöltött képeken
  - Példák
    - Simítás környezete interaktívan, csúszkán állítható legyen
- Komponensdetektálás és számlálás
  - A műveletek sebessége miatt erőforrásból betöltött képen célszerű dolgozni
    - Lehetőleg külön szálon futtatva
  - Homogén komponenseket tartalmazó képeken (pl. feliratok)
    - Alkalmazzunk a képen adaptív küszöbölést
    - Határozzuk meg az összefüggő komponenseket
    - Adott mérettartományba eső komponensek határát vörös színnel rajzolják rá a szürkeárnyalatos képre
- Képszerkesztő intent-re reagáló alkalmazás
  - Egyszerű művelet eredményét adjuk vissza a hívónak

Natív C/C++ Android programozás

Mobil képelemzés és grafika

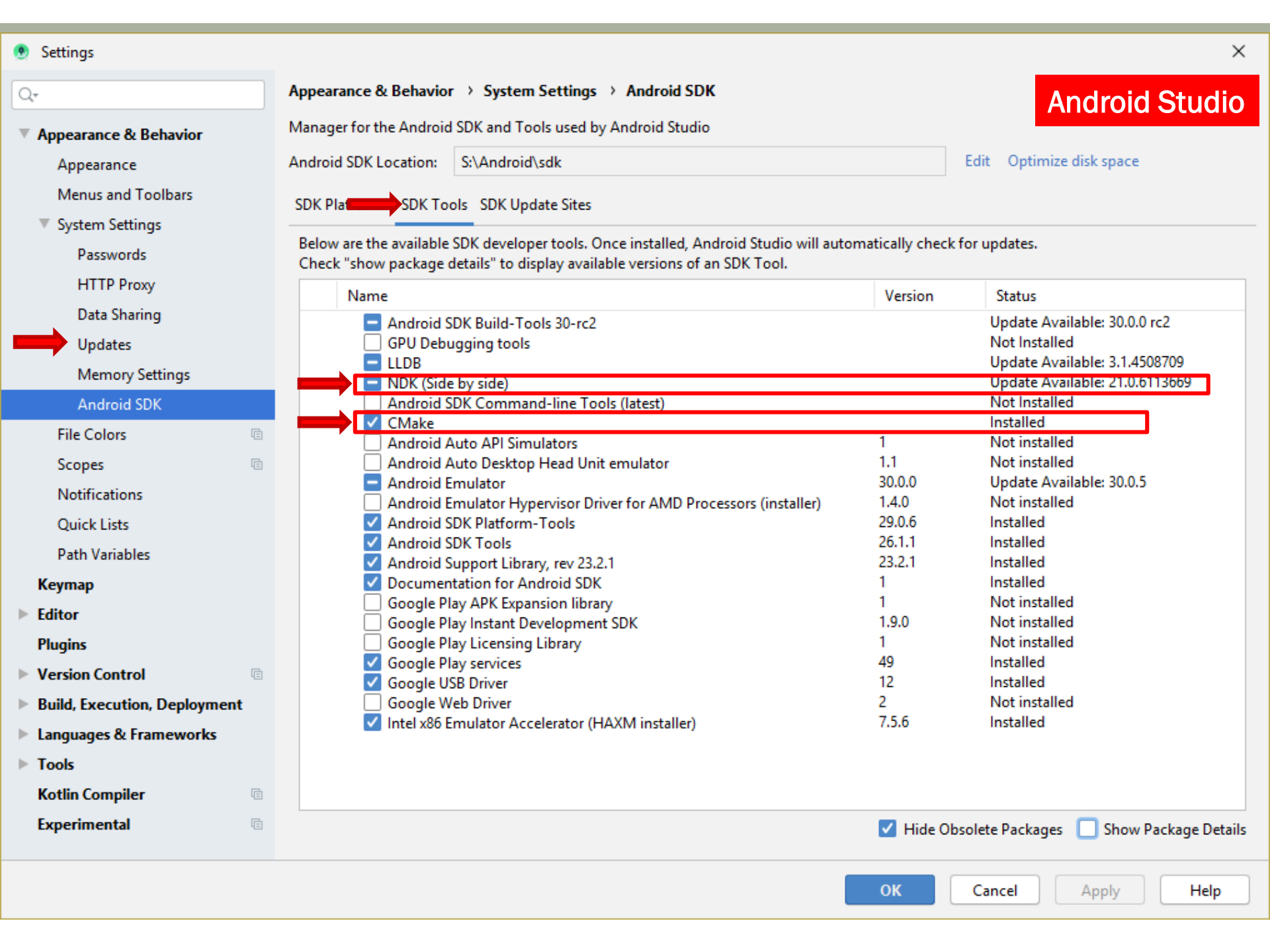
Natív C/C++ Android programozás

- Az alkalmazás egyes részeit C/C++ nyelven írhatjuk
  - Nem a Java API alternatívája!
  - A GUI, vezérlő kód jellemzően Java nyelvű marad
- Az Android API elérése kényelmetlenebb C/C++ felől
  - Android App Glue: akár az életciklus eseményekről is kaphatunk értesítést (onCreate(), ...)
- Előnye
  - Korábbi/meglévő C/C++ kód egyszerűbb átemelésére Android környezetbe
    - OpenGL, OpenCV, különféle lib-ek, ...
  - Számítás-intenzív részek megvalósítására
  - Memóriakorlát kikerülésére
    - Pl. képek, fotó esetén
- Hátránya
  - Sok adatcsere Java<->C/C++ között nem hatékony, komplexebb

Odafigyelést igénylő, rázós terep!

Működése

- A C/C++ kód külön fordul
  - make + C/C++ fordító környezet kell, újabban CMake
  - .so (shared object) modulként jön létre
  - Akár többféle architektúra is kezelhető vele (ARM, MIPS, x86, …)
- Az Android fordítási környezet az APK-ba helyezi
  - .so fájlok architektúránként különböző mappákban elhelyezve
  - A futtatási környezet választja ki, melyik töltődjön be
- Paraméterátadás a JNI specifikáció szerint
  - A Java programunk és a C/C++ kód között
- Az újabb Android Studio verziók már tartalmaznak NDK támogatást
  - A fordítás együtt történik
  - Az IDE környezet felismeri és kezeli a C/C++ szintaktikát


Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Updates
Memory Settings
Android SDK
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Kotlin Compiler
Experimental
Appearance & Behavior › System Settings › Android SDK
Android Studio
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: S:\Android\sdk
Edit
Optimize disk space
SDK Plat
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools 30-rc2 Update Available: 30.0.0 rc2
GPU Debugging tools Not Installed
LLDB Update Available: 3.1.4508709
NDK (Side by side) Update Available: 21.0.6113669
Android SDK Command-line Tools (latest) Not Installed
CMake Installed
Android Auto API Simulators 1 Not installed
Android Auto Desktop Head Unit emulator 1.1 Not installed
Android Emulator 30.0.0 Update Available: 30.0.5
Android Emulator Hypervisor Driver for AMD Processors (installer) 1.4.0 Not installed
Android SDK Platform-Tools 29.0.6 Installed
Android SDK Tools 26.1.1 Installed
Android Support Library, rev 23.2.1 23.2.1 Installed
Documentation for Android SDK 1 Installed
Google Play APK Expansion library 1 Not installed
Google Play Instant Development SDK 1.9.0 Not installed
Google Play Licensing Library 1 Not installed
Google Play services 49 Installed
Google USB Driver 12 Installed
Google Web Driver 2 Not installed
Intel x86 Emulator Accelerator (HAXM installer) 7.5.6 Installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply
Help

Natív SDK használata (régi)

- **Manuális C/C++ fordítás**
  - A natív kódot a `<project>/jni` könyvtárba tegyük
  - Készítsünk a `jni` könyvtárba egy `Android.mk` fájlt
    - Megadja, hogyan készüljön `.so` fájl a forrásból
  - Opcionálisan `Application.mk` fájlt is ugyanitt
    - Milyen chipset-re készüljön el a bináris (`armeabi`, `mips`, `x86`, ...)?
  - Futtassuk az `ndk-build` szkriptet (`NDK` könyvtárban van)
    - `<project>` könyvtárból indítsuk, parancssorból
    - Ne legyen szóköz az elérési útvonal nevében!
  - Utána fordítsuk le a Java Android alkalmazást a szokásos módon

Android Studio: <project/modul>/src/main/jni

Windows: ndk-build.cmd

<project/modul>/src/main/

- **Figyeljünk!**
  - A Java fordítás akkor is sikeres, ha nincs lefordítva a natív rész!
  - Ekkor futási idejű hibát kapunk!
  - A natív rész módosítása után ne felejtsük el annak fordítását!

Natív SDK használata (régi)

- Manuális fordítás Android Studio-val
  - Hozzuk létre a `jni` és a `jniLibs` mappákat a `<Project/modul>/src` -ben
  - A `<modul>/build.gradle` fájlt módosítsuk, hogy ne fordítsa a C/C++ részt! Ha van `ndk` bejegyzés, vegyük ki!

```
android {
        …
        sourceSets {
                main {
                        jni.srcDirs = []
                }
        }
}
```

Az NDK alapértelmezésként a libs mappába teszi a csomagokat, az Android Studio viszont a jniLibs-ben keresi őket…

  - Fordítsunk parancssorból az `ndk-build` használatával
    - `<Project/modul>/src/main` mappából indítsuk
    - `ndk-build NDK_LIBS_OUT=./jniLibs NDK_OUT=./obj`
  - Jelezni fog a Java kódban a jni függvény nevénél, hogy nincs ilyen, de futni rendben fog.

Hello-JNI make (régi, új)

- Application.mk

```
APP_ABI := all
```

- Android.mk

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE    := hello-jni
LOCAL_SRC_FILES := hello-jni.c
include $(BUILD_SHARED_LIBRARY)
```

Hello-JNI CMakeLists.txt (új)

```
cmake_minimum_required(VERSION 3.4.1)

add_library(hello-jni SHARED
            hello-jni.c)

# Include libraries needed for hello-jni lib
target_link_libraries(hello-jni
                      android
                      log)
```

Hello-JNI C (régi-új)

Függvénynév kötött! (Java csomagnév alapján.)
(Vagy lehet használni a RegisterNatives függvényt a JNI_OnLoad()-ban.)

```
#include <string.h>
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env, jobject thiz ) {
#if defined(__arm__)
  #if defined(__ARM_ARCH_7A__)
    #if defined(__ARM_NEON__)
      #define ABI "armeabi-v7a/NEON"
    #else
      #define ABI "armeabi-v7a"
    #endif
  #else
   #define ABI "armeabi"
  #endif
#elif defined(__i386__)
   #define ABI "x86"
#elif defined(__mips__)
   #define ABI "mips"
#else
   #define ABI "unknown"
#endif

    return (*env)->NewStringUTF(env, "Hello from JNI !  Compiled with ABI " ABI ".");
}
```

Ezt a két paraméter mindig átadásra kerül!

Működése:
C++ feltételes fordítási kritériumokat használva visszaadja a natív architektúra megnevezését sztringként. Más-más sztring kerül be a különféle architektúrákhoz tartozó .so könyvtárakba.

Java interface függvények elérése.

hello-jni.c (2020)

```
#include <string.h>
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_hellojni_HelloJni_stringFromJNI( JNIEnv* env,
                                                  jobject thiz )
{
#if defined(__arm__)
    #if defined(__ARM_ARCH_7A__)
    #if defined(__ARM_NEON__)
      #if defined(__ARM_PCS_VFP)
        #define ABI "armeabi-v7a/NEON (hard-float)"
      #else
        #define ABI "armeabi-v7a/NEON"
      #endif
    #else
      #if defined(__ARM_PCS_VFP)
        #define ABI "armeabi-v7a (hard-float)"
      #else
        #define ABI "armeabi-v7a"
      #endif
    #endif
  #else
   #define ABI "armeabi"
  #endif
#elif defined(__i386__)
#define ABI "x86"
#elif defined(__x86_64__)
#define ABI "x86_64"
#elif defined(__mips64)  /* mips64el-* toolchain defines __mips__ too */
#define ABI "mips64"
#elif defined(__mips__)
#define ABI "mips"
#elif defined(__aarch64__)
#define ABI "arm64-v8a"
#else
#define ABI "unknown"
#endif

    return (*env)->NewStringUTF(env, "Hello from JNI !  Compiled with ABI " ABI ".");
}
```

Hello-JNI Java (régi-új)

```
package com.example.hellojni;

import android.app.Activity;
import android.widget.TextView;
import android.os.Bundle;

public class HelloJni extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView  tv = new TextView(this);
        tv.setText( stringFromJNI() );
        setContentView(tv);
    }

    public native String stringFromJNI();

    static {
        System.loadLibrary("hello-jni");
    }
}
```

Natív C++ függvény hívása

Natív C++ függvény fejléce

Natív .so betöltése az aktivitás példányosításakor.

Android Studio 2.2 (régi)

- Megjelenik az NDK integráció a keretrendszerben
  - CMake telepítése szükséges
    - `Settings / System Settings / Android SDK / SDK Tools`
  - Új projekt generálásakor bekapcsolhatjuk az NDK integrációt
    - `Include C++ support`
    - CMake-et fog használni
  - Már létező projekthez hozzáadhatjuk a C++ támogatást
    - Jobb egérgomb a projekt nevén / `Link C++ Project with Gradle`
    - Választhatunk a CMake és a régi `.mk` fájlok között

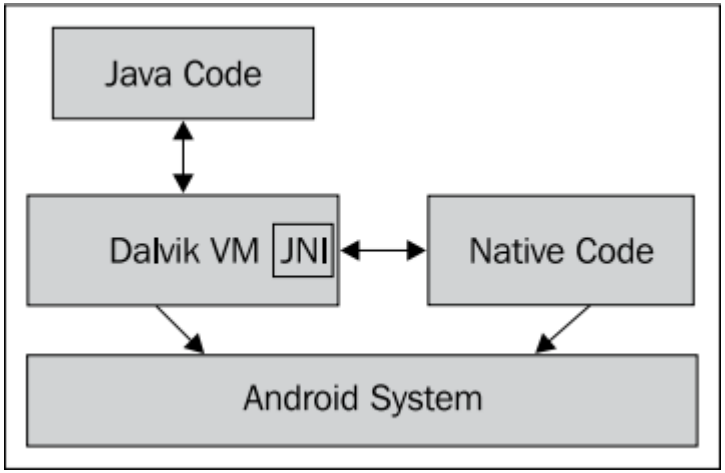
A `CMakeLists.txt` fájlt magunknak kell megírni.

Android Studio 3 (aktuális)

- Jellemzői
  - Új projekt létrehozása, vagy már meglévőhöz hozzáadás
  - Használata integrált, zökkenőmentes, nincs szükség kézi fordítási lépések megtételére
- Új projekt létrehozása varázslóval
  - `Native C++` sablon választásával
  - `C++ Standard`: `Toolchain Default`, `C++ 11`, `C++ 14`, `C++ 17`
  - CMake-et fog használni
  - C++ vázat generál
- Natív C/C++ rész hozzáadása
  - Mint az előző dián az AS 2.2 résznél

JNI működés

- Paraméterek átvétele Java kódból
  - JNI: Java Native Interface
    - Általános Java-C++ interfész (nem Android specifikus)
    - https://developer.android.com/guide/practices/jni.html
  - `JNIEnv` mutató
    - JNI függvénytábla címe
    - C/C++ belépési pont kapja meg
  - `jobject thiz`
    - Aktuális Java objektumpéldány



- C++ kód esetén
  - A belépési kód C nyelv szerinti
  - Kell egy `extern "C" { … }` a megfelelő kezeléshez
  - Ebből hívhatunk C++ kódot

JNI típusok

- Elemi típusok

Java Type	JNI Type	Number of bytes	Sign
`boolean`	`jboolean`	1	unsigned
`byte`	`jbyte`	1	signed
`char`	`jchar`	2	unsigned
`short`	`jshort`	2	signed
`int`	`jint`	4	signed
`long`	`jlong`	8	signed
`float`	`jfloat`	4	-
`double`	`jdouble`	8	-

- Java objektumok (pl. **`Bitmap`**, **`string`**)
  - **`jobject`**
  - **`jstring`**

Natív C/C++ SDK elemei

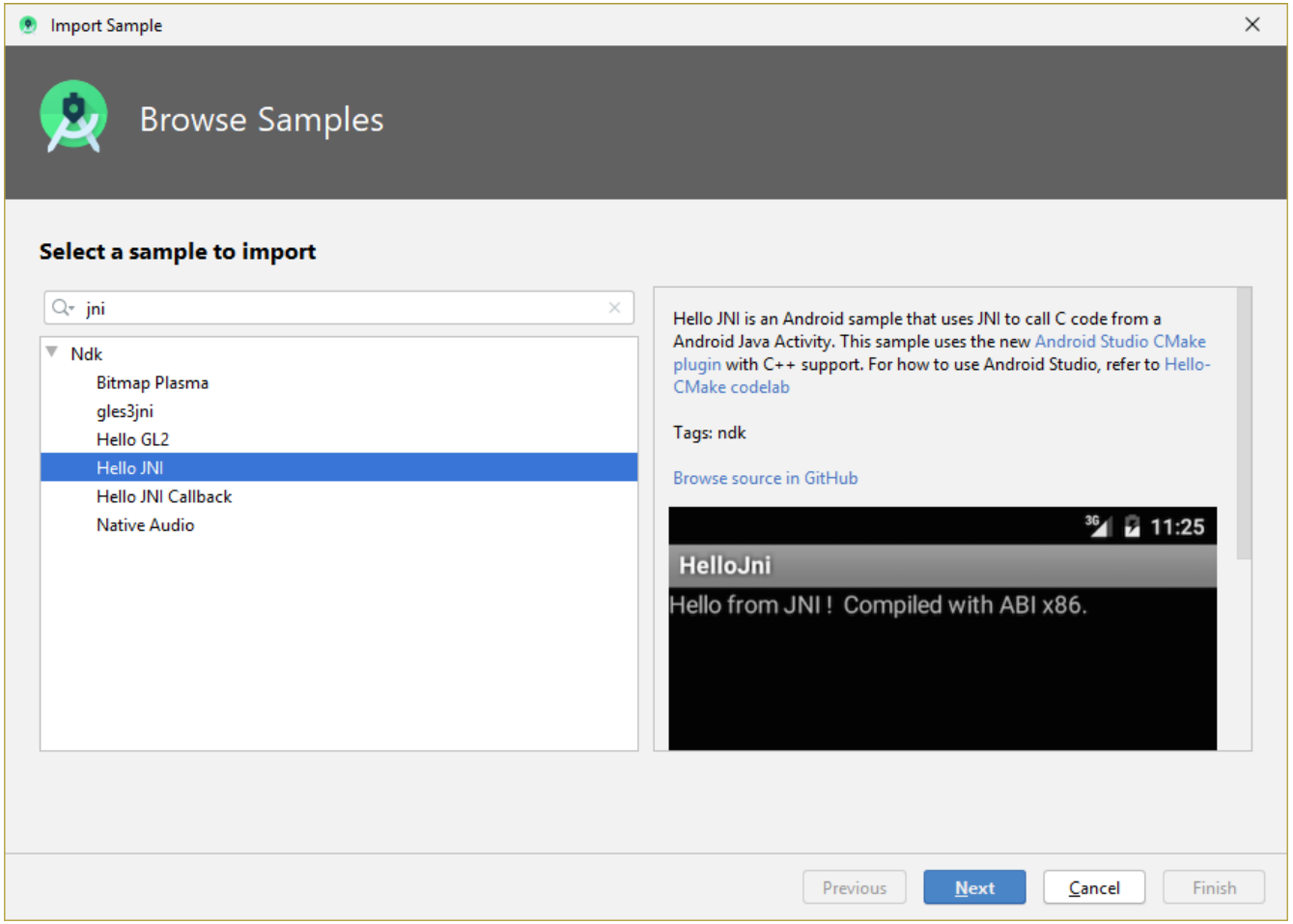
- Natív logolás
  - Forráskódban

    ```
    #include <android/log.h>
    ```
  - `Android.mk` fájlban fel kell sorolni a lib-eket

    ```
    LOCAL_LDLIBS := -llog
    ```
  - `Log` függvény fejléc

    ```
    __android_log_print(ANDROID_LOG_INFO, LOG_TAG, __VA_ARGS__);
    ```
- Példák
  - **`hello-jni`**: sztringet ad vissza a natív kód, amit a Java oldal megjelenít
- JNI példák kipróbálása (Android Studio)
  - Főképernyő / `Import an Android code sample`
    - Szűrjünk az `ndk` vagy `c++` kifejezésre
    - Gradle rendben lefordítja, nem kell kézzel

Import Code Sample



Munka Bitmap objektumokkal

- Probléma
  - A kicsomagolt képek nagy méretűek lehetnek
  - Az Android rendszer korlátozza a Java keretrendszer programjai számára a memóriahasználatot!
    - 5 MP kép 19MB!

Screen Size	Screen Density	Application Memory
small / normal / large	ldpi / mdpi	16MB
small / normal / large	tvdpi / hdpi	32MB
small / normal / large	xhdpi	64MB
xlarge	mdpi	32MB
xlarge	tvdpi / hdpi	64MB
xlarge	xhdpi	128MB

- Lehetséges megoldások
  - Használjunk különböző méretűeket
  - Eleve kisebb méretben töltsük be
    - Displaying Bitmaps Efficiently
  - Használjunk natív C/C++ kódot a képfeldolgozáshoz
    - Nincs méretkorlát (csak a rendelkezésre álló memória)
    - Fotós alkalmazások esetén szinte elkerülhetetlen
    - AndroidJniBitmapOperations-master.zip

Egyelőre OpenCV nélkül dolgozunk!

Android natív C/C++ Bitmap API

- Fejléc fájl

```
#include <android/bitmap.h>
```

- Kezelt Bitmap formátumok

```
enum AndroidBitmapFormat {
    ANDROID_BITMAP_FORMAT_NONE      = 0,
    ANDROID_BITMAP_FORMAT_RGBA_8888 = 1, // 4 bájtos RGBA kép
    ANDROID_BITMAP_FORMAT_RGB_565   = 4, // 2 bájtos RGB kép
    ANDROID_BITMAP_FORMAT_RGBA_4444 = 7,
    ANDROID_BITMAP_FORMAT_A_8       = 8,
};
```

- Bitmap információs struktúra

```
typedef struct {
    uint32_t    width;
    uint32_t    height;
    uint32_t    stride;     // Egy képsor mérete bájtban
    int32_t     format;
    uint32_t    flags;      // 0 for now
} AndroidBitmapInfo;
```

C/C++ Bitmap kezelőfüggvények

- `getInfo()`
  - `int AndroidBitmap_getInfo(JNIEnv* env, jobject jbitmap, AndroidBitmapInfo* info);`
  - `AndroidBitmapInfo` struktúra elkészítése Java `Bitmap` objektumról
  - Formátum, méret, memóriabeli struktúra információk
- `lockPixels()`
  - `int AndroidBitmap_lockPixels(JNIEnv* env, jobject jbitmap, void** addrPtr);`
  - `Bitmap` képi adata a visszaadott pointeren keresztül elérhető
  - Zárolás *(lock)*: a Java nem mozgatja át másik memóriaterületre
- `unlockPixels()`
  - `int AndroidBitmap_unlockPixels(JNIEnv* env, jobject jbitmap);`
  - `Bitmap` zárolás feloldása

C/C++ kódtöredék példa

```
extern "C" JNIEXPORT jstring JNICALL
Java_hu_uszeged_ipcg_jnibitmaptester_MainActivity_JNIProcess
Bitmap(JNIEnv* env, jobject thiz, jobject jBitmap)
{
  AndroidBitmapInfo abmi;
  void *basePtr = NULL;
  char msg[512];

  AndroidBitmap_getInfo( env, jBitmap, &abmi );
  AndroidBitmap_lockPixels(env, jBitmap, &basePtr);
  // Feldolgozás kódja...
  AndroidBitmap_unlockPixels(env, jBitmap);

  sprintf( msg, "Bitmap format: %d; Size: %d x %d; Stride:
  %d", abmi.format, abmi.width, abmi.height, abmi.stride );

  return env->NewStringUTF( msg );
}
```

C++ Bitmap

- Képpontok elérése
  - `lockPixels()` által visszaadott pointer értékkel
  - Az első képpont memóriacímét adja
  - A struktúra `stride` értéket használjuk a sorok váltásához!

- Mit csinál az alábbi kódrészlet?

```
if(abmi.format == 1)
    {
        memset((unsigned char *)basePtr +
                abmi.stride * ( abmi.height >> 1 ), 255,
                abmi.width * 4);
    }
```

Fordítás

CMakeLists.txt

```
target_link_libraries(
        native-lib jnigraphics
        ${log-lib} )
```

Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE    := native-lib
LOCAL_SRC_FILES := native-lib.cpp
LOCAL_LDLIBS := -llog
LOCAL_LDFLAGS += -ljnigraphics

include $(BUILD_SHARED_LIBRARY)
APP_OPTIM := debug
LOCAL_CFLAGS := -g
```

`jnigraphics` könyvtárat fel kell venni a szerkesztési listára!

Feladat

- Példaprogram tesztelése
  - `07_JNIBitmapTester`
    - Betölt erőforrásból egy Bitmap képet,
    - átadja egy natív C++ függvénynek,
    - amely sztringben visszaadja a kép adatait (méret, típus, stride, ...),
    - valamint fehér vonalat húz vízszintesen a középső sorban, ha megfelelő formátumú a Bitmap kép (32 bit/képpont).
    - A sztringet a Java oldalon kiírja `TextView` objektumban, a képet pedig megjeleníti egy `ImageView` objektumban.

- Egészítsük ki a példaprogramot!
  - Írjunk függvényeket a képmátrix vízszintes és függőleges tükrözésére!
  - Implementáljunk további képmátrix manipulációkat!

További példaprogramok

- `Bitmap Plasma` (Android Sample)
  - Android Studio nyitóoldal, `Import an Android code sample`
- `AndroidJniBitmapOperations-master`

Android OpenCV használat C/C++-ban

Mobil képelemzés és grafika

Android OpenCV C++

- Jellemzők
  - Az OpenCV eredetileg C++ függvénykönyvtár
  - Legoptimálisabb ezen a nyelven használni
  - Az Android C/C++ környezete fordítja az OpenCV forráskódot
  - JNI-n keresztül tudunk C++ kódot hívni Java-ból

- Multiplatform (okostelefon, PC) alkalmazások esetén előnyös lehet
  - A feldolgozási rész mehet C++-ban, ami könnyen átvihető a platformok között
  - A GUI, fájlkezelés Java-ban, vagy platform specifikus módon
  - Igaz, Java-n keresztül is dolgozhatunk PC-n is…

Android OpenCV C++

- Fordítási megközelítések
  - Használhatjuk az új, CMake megközelítést
    - Manuális gradle módosításokat igényel
  - Egyelőre egyszerűbb az élet a régi make `.mk` fájlokkal
    - Ezt választjuk

Natív OpenCV fordítás (make)

- Lehetséges lépések (Android Studio-ban)
  - Nyissuk meg az `OpenCV420InitFull` projektet
  - Adjunk hozzá új modult, pl. `OpenCVLiveNativeTest` néven
  - Adjuk hozzá az OpenCV modul függőséget (`F4`, `Dependencies`)
  - Másoljuk át az `OpenCVLiveView` forrásanyagát
    - Java kód
    - Layout erőforrás
  - Módosítsuk a `Manifest.xml` fájlt
    - Kamera engedélyek
    - Fekvő orientáció
  - Adjuk hozzá a natív függvény fejléce(ke)t és gondoskodjunk a natív lib statikus betöltéséről
  - Megfelelő helyen hívjuk meg a natív függvényeket

Egyszerűbb, ha az OpenCVLiveView modult egészítjük ki...

Natív OpenCV fordítás (make)

- Lépések (natív, cpp forrásmappában)
  - Adjunk hozzá C++ kódot az `src/cpp` mappába
    - Például `ocv_jni_code.cpp` néven, benne az implementációval
  - Adjuk meg ugyanitt az `Android.mk` fájlt

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

include include ../opencv420sdk/native/jni/OpenCV.mk

LOCAL_MODULE    := ocv_jni
LOCAL_SRC_FILES := ocv_jni_code.cpp
LOCAL_LDLIBS +=  -llog -ldl

include $(BUILD_SHARED_LIBRARY)
```

Akkor működik, ha a teljes OpenCV-t beimportáltuk!

Natív OpenCV fordítás (make)

- Az `Application.mk` fájl tartalma az **`src/cpp`** mappában

```
APP_STL := c++_static
APP_CPPFLAGS := -frtti -fexceptions
APP_ABI := armeabi-v7a arm64-v8a
APP_PLATFORM := android-28
```

- Fordítsuk le és futtassuk a projektet!
  - `Build / Refresh Linked C++ Projects`, ha a natív kód változott
  - De a `Run` is észre szokta venni

Natív OpenCV használat

- Paraméterátadás C++ kódnak
  - Alaptípusok és Java objektumok
    - Mint korábban a JNI résznél
  - OpenCV `Mat` objektum átadása
    - Referencia szerint történik
      - Ha a C++ oldal változtat a `Mat`-on, akkor a Java oldali is változik
    - Java oldal (átadás)
      - `Mat.getNativeObjAddr()`
    - C++ oldal (átvétel)
      - `long` értékként vehető át
      - Cast-olni kell `Mat` referenciára, pl.:

        ```
        Mat &mIn = *(Mat *)addrIn;
        ```

      - `mIn` innentől kezdve C++ `Mat` objektumként használható

Példaprogram

- `OpenCV420InitFull / OpenCVNativeMk`
  - Kamera élőképen működik
  - Niblack adaptív küszöbölés (C++-ban)
  - Egyszerű küszöbölés 128 értéknél (C++-ban)
    - Ki van kommentározva alapból

Android Studio 3 (CMake)

- Lépések
  - New Project
    - Include C++ support
    - Empty Activity
    - Exceptions Support
    - Runtime Type Information
  - Add Module
    - OpenCV 4.1.1
  - Settings / Dependencies
    - Add Module Dependency: OpenCV

Az Android Studio 3.6 verzióban már nem állíthatók be, csak a projekt létrehozása után tehetjük meg!

Egyelőre hagyjuk a CMake vonalat az OpenCV fordítása kapcsán!

Android Studio 3 (CMake)

- Lépések
  - CMakeList.txt

```
include_directories(C:/Dev/OpenCV-2.4.11-android-sdk/sdk/native/jni/include)
add_library( lib_opencv SHARED IMPORTED )
set_target_properties(lib_opencv PROPERTIES IMPORTED_LOCATION ${CMAKE_CURRENT_SOURCE_DIR}../../../../OpenCV-2.4.11-android-sdk/sdk/native/libs/${ANDROID_ABI}/libopencv_java.so)

target_link_libraries( # Specifies the target library.
                       native-lib
                       lib_opencv

                       # Links the target library to the log library
                       # included in the NDK.
                       ${log-lib} )
```

Android Studio 3 (CMake)

- Lépések (statikus szerkesztés esetén)
  - jniLibs mappa az app/src/main-ben
    - Ide összegyűjteni a különféle architektúrák .so OpenCV fájljait
      - Armeabi-v7a, ...
  - Natív kód
    - Main/cpp mappában
  - Natív lib betöltése
    - Java kódban (Activity osztályban)

```
// Used to load the 'native-lib' library on
// application startup.
    static {
        System.loadLibrary("opencv_java4");
        System.loadLibrary("native-lib");
    }
```

Képi jellemzők OpenCV-ben
Kiegészítő anyag

Mobil képelemzés és grafika

Képi jellemzők OpenCV-vel

- Intenzitás értékek, színek
- Pontok
  - Sarokpontok (Moravecz, Harris-Stephens)
  - Kulcspont detektorok és leírók (SIFT, SURF, STAR, FAST, …)
- Élek, egyenesek, körök, geometriák
  - Gradiens operátorok, Laplace, Canny, …
  - Hough transzformáció (egyenesek, körök)
- Komponens (régió) detekció
  - Küszöbölés (optimális, adaptív, …)
  - Morfológiai műveletek (dilatáció, erózió, nyitás/zárás, …)
  - Szegmentálás (régiónövelés, aktív kontúr, mean shift, …)
  - ROI kinyerés

Felismerés

- Komponens jellemzők
  - Határ-alapú leírók
  - Régió-alapú leírók
  - Transzformációs kódolások
  - Különböző leírók (súlyozott) együttese, vektora
- Szimbólum felismerés
  - Tanító halmazon osztályok, jellemzővektorok összeállítása
  - Felismerendő objektumra jellemzővektor számítása
  - Osztályozás
    - Vektorok „hasonlósága” alapján melyik osztályba tartozik
      - Legközelebbi elem, SVM, …
    - Neuronhálók betanítása
    - …

Inercia mátrix

- Inercia mátrix

$$C(x,y)=\begin{bmatrix}\overline{I_x(x,y)}\cdot\overline{I_x(x,y)} & \overline{I_x(x,y)}\cdot\overline{I_y(x,y)}\\ \overline{I_y(x,y)}\cdot\overline{I_x(x,y)} & \overline{I_y(x,y)}\cdot\overline{I_y(x,y)}\end{bmatrix}$$

$I_x(x,y),\quad I_y(x,y)$ Kép gradiense x- és y-irányban

$\overline{I_x(x,y)},\quad \overline{I_y(x,y)}$ Gradiens átlaga (x,y) pont lokális (körlap alakú) környezetében

- Egyszerű mértékek
  - $\det(\mathbf{C})$, $\det(\mathbf{C})/\mathrm{tr}(\mathbf{C})$
  - $\mathbf{C}$ sajátértékeinek (λ_1, λ_2) vizsgálata
    - $\det(\mathbf{C}-\lambda\mathbf{I})=0$
    - **Ahol mindkettő nagy, ott lehet sarokpont.**

2D-s sarokpont-mértékek

- Harris és Stephens

$$R = \frac{k}{(k+1)^2} \cdot [\det(C) - tr(C)]$$

$$k = \frac{\lambda_1}{\lambda_2}$$

Gyorsabb számítási mód:

$$R = \det(C) - \kappa \cdot tr^2(C) \qquad 0.04 \le \kappa \le 0.15$$

- Beaudet

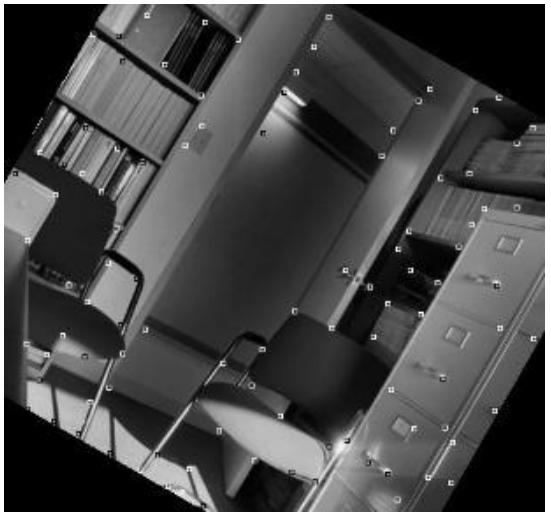
$$I_{xx} I_{yy} - I_{xy}^2$$

- Kitchen és Rosenfeld

$$\frac{I_{xx} I_y^2 - 2 I_{xy} I_x I_y + I_{yy} I_x^2}{I_x^2 + I_y^2}$$

Sarokpontok detektálása

- Stabil sarkok választása
  - Különböző méretű, kör alakú lokális környezetek vizsgálatával
  - Stabil sarokpont: több környezetméretnél is jelen van
- Pontok számának csökkentése
  - Lokális környezetben csak 1 sarokpont
- Keresés felgyorsítása
  - Élkép felhasználásával



Forrás: A.A. Goshtasby

Kulcspont detektorok és leírók

- **Detektor**
  - Pontok, régiók automatikus detektálása a képen
    - Pl. pontok egy adott alakú környezetében, hasonló intenzitású területek, azokra illesztett ellipszis alakú régiók, …
- **Leíró (deszkriptor)**
  - Az egyes objektumokhoz rendelt jellemző adatok
- **Párosító**
  - A leírók hasonlóságát és/vagy geometriai kényszereket alkalmazva

Tinne Tuytelaars and Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey
Foundations and Trends in Computer Graphics and Vision Vol. 3, No. 3 (2007) 177–280

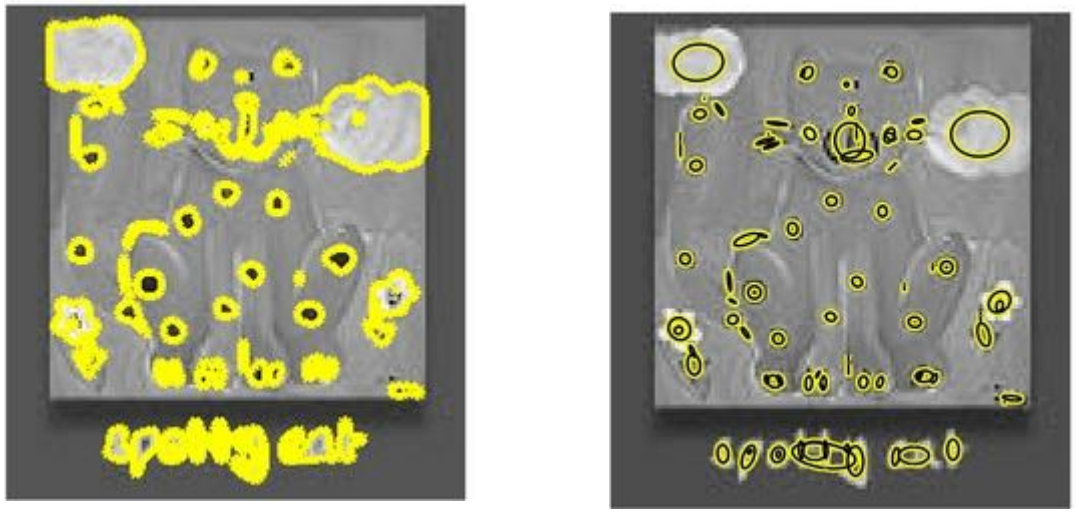
Kulcspont detektorok és leírók

- Detektor
  - SIFT, SURF, MSER, IBR, SUSAN, Harris-Laplace, Harris-Affine, él alapú, ...
  - Hatékonyság: SURF, FAST, DoG
- Leíró (deszkriptor)
  - SIFT, SURF
- Párosító
  - Brute-force, FLANN, RANSAC
- Szempontok
  - Invariancia: milyen geometriai torzulásokat képes kompenzálni?
    - Eltolás, orientáció, skála, affin, perspektív torzulás
  - Számítási sebesség
  - Pontosság, robusztusság, megismételhetőség

MSER detektor

- „Maximally Stable Extremal Regions”
  - (Matas, 2002)
- Képstruktúrák detektálása
  - Affin invariáns

Forrás: http://www.vlfeat.org/overview/mser.html



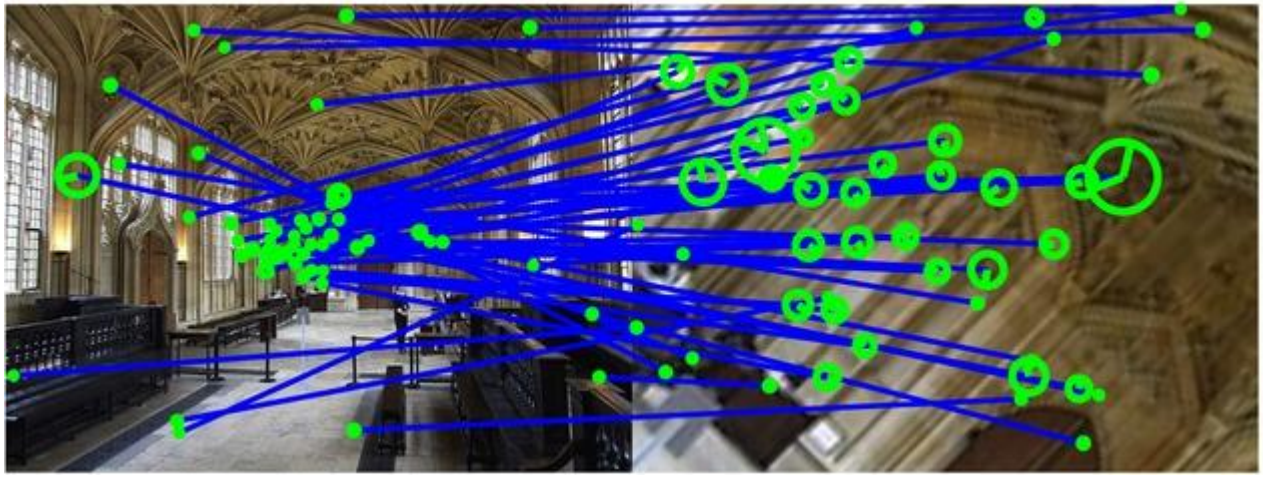
Detektált MSER területek + illesztett ellipszisek

MSER detektor

- Működése
  - Különböző küszöbölési szintek
    - Növelhető vagy csökkenthető a küszöbérték
  - Összefüggő komponensek méretnövekedési ütemének lokális minimumai (összeolvadásuk figyelésével)
  - Ellipszis illesztése
    - Régió súlypont + optimális ellipszis

SIFT detektor + leíró

- „Scale-invariant feature transform”
  - Lowe, 1999
- SIFT jellemző
  - SIFT detektor és SIFT leíró (deszkriptor)



Forrás: http://www.vlfeat.org/overview/sift.html

SIFT detektor

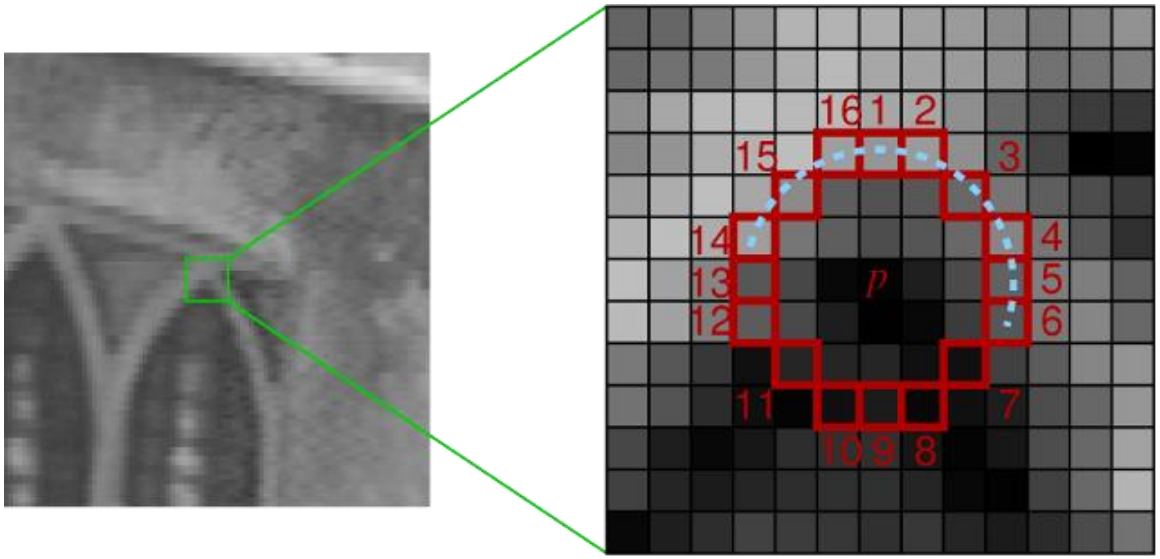
- Detektált kör alakú terület (blob)+ orientáció
  - Keresés DoG skálatérben
    - Több skálán is előforduló lokális extrémumok
    - Invariáns (kovariáns) eltolásra, forgatásra, skálázásra
  - Orientáció meghatározása: helyi képkörnyezet
    - Elforgatás kovariáns, de szimmetrikus kulcspont-környezet problémát okoz
  - Nem stabil pontok visszautasítása
    - Alacsony kontraszt
    - Élek közelében

SURF detektor

- Hatékony módosítása a SIFT-nek
  - Doboz-szűrők Gauss simítás helyett
    - Integrál kép-alapú számítás (gyors)
  - Laplace operátor közelítése Hessian-különbség reprezentációval
  - Skála-tér: szűrők felskálázása (kép leskálázás helyett) 9x9, 15x15, 21x21, 27x27 méretre
  - Blob-válasz térkép
    - Hesse-féle mátrix determinánsnak közelítése
    - Ezek lokális maximuma különféle skálákon

FAST detektor

- SUSAN kritérium specializált változata
  - 16 pont vizsgálata a pont körül r=3 sugarú diszkrét körön
  - Ha legalább *n*=9 pont világosabb vagy sötétebb, mint *t* küszöbérték, akkor detekció
  - Pontok vizsgálati sorrendje optimalizálható
  - Nagyon gyors futásidő



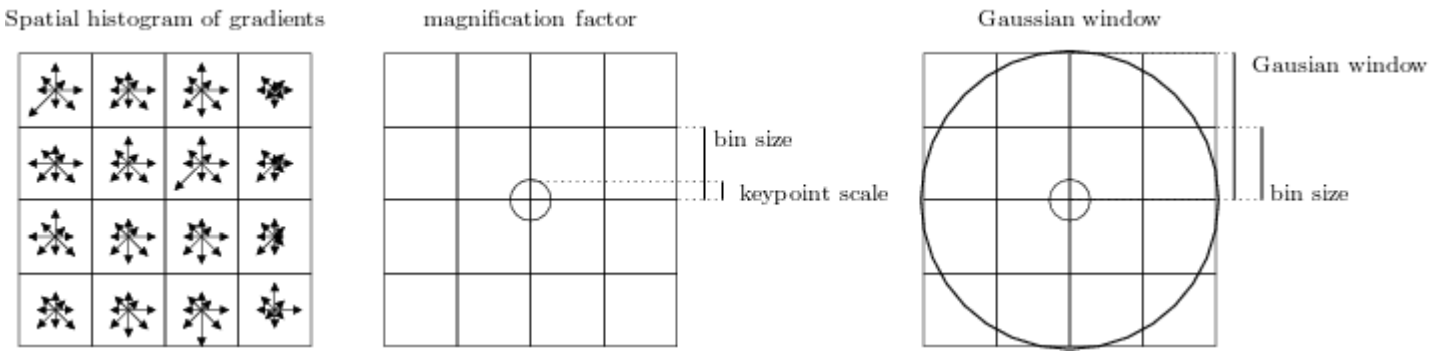
Forrás: Edward Rosten

GFTT detektor

- **Good Features to Track (Shi-Tomasi detektor)**
  - Videósorozaton detekció, mozgáskövetés (kis elmozdulások)
  - 2x2 Harris mátrix kisebb sajátértéke

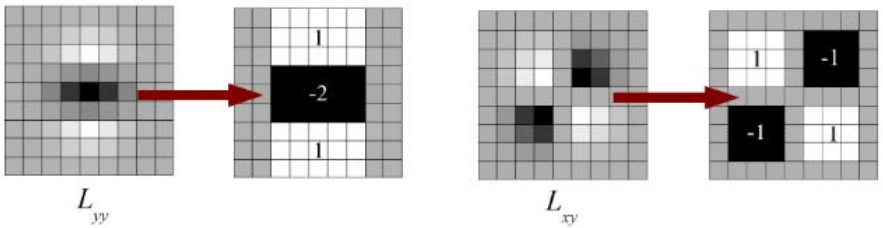
SIFT leíró

- Kulcspontok megjelenésének karakterizálására
  - 3D jellemző vektor (képpont helye + gradiens orientáció)
  - Normalizálás + 3D összegzőtömb, Gauss-súlyozás a távolabbiak hatásának csökkentésére
  - 128 elemű skalár vektor

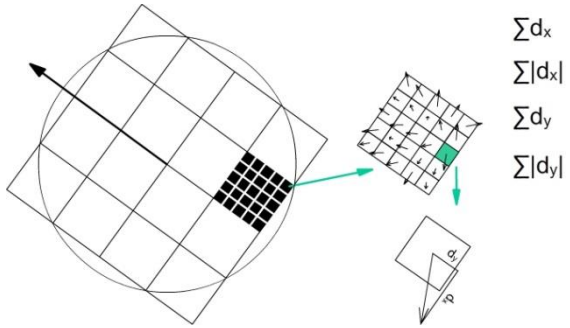


SURF leíró

- Hatékony módosítása a SIFT-nek
  - Intenzitásértékek figyelembe vétele a kulcspont körül
  - Elsőfokú Haar-wavelet válaszok eloszlása
    - Integrálkép-reprezentáció használata
  - Opcionálisan orientációs ablak
  - 64 vagy 128 elemű skalár vektor



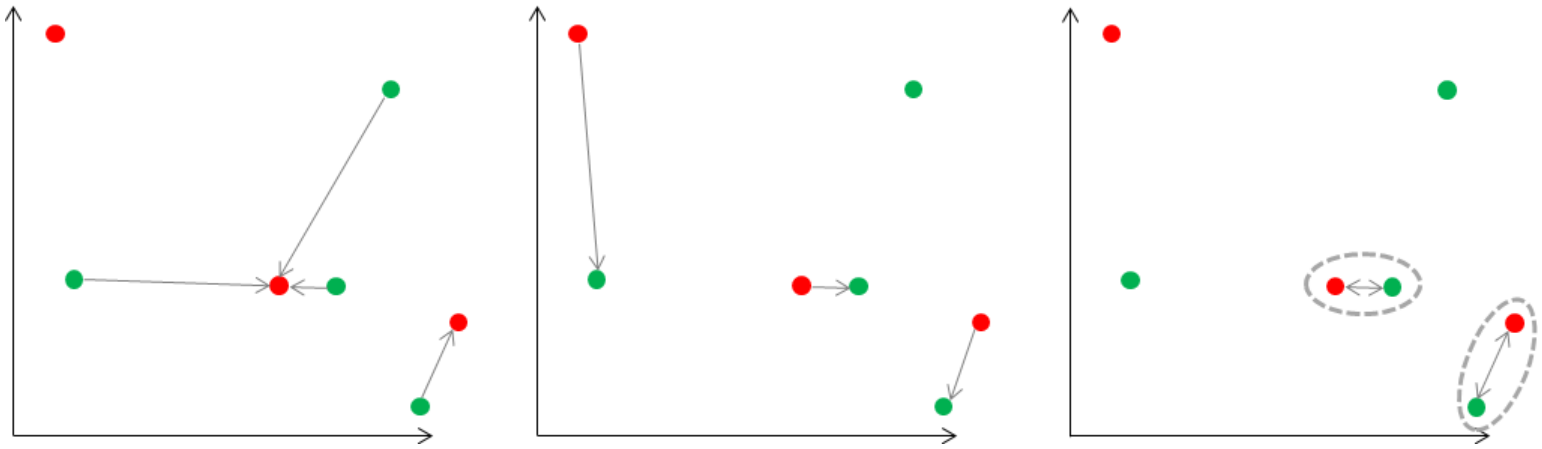
SURF detektor (doboz-szűrők)



SURF leíró számítása

Leíró-alapú kulcspont párosítás

- Képpárok között
  - Képenként kulcspontok + leírók meghatározása
- Párosítás jellemzővektor-hasonlóság alapján
  - Teljes keresés nem hatékony
  - FLANN k-nn algoritmus, keresztvalidációval
  - Geometriai-alapú szűrés RANSAC-kal



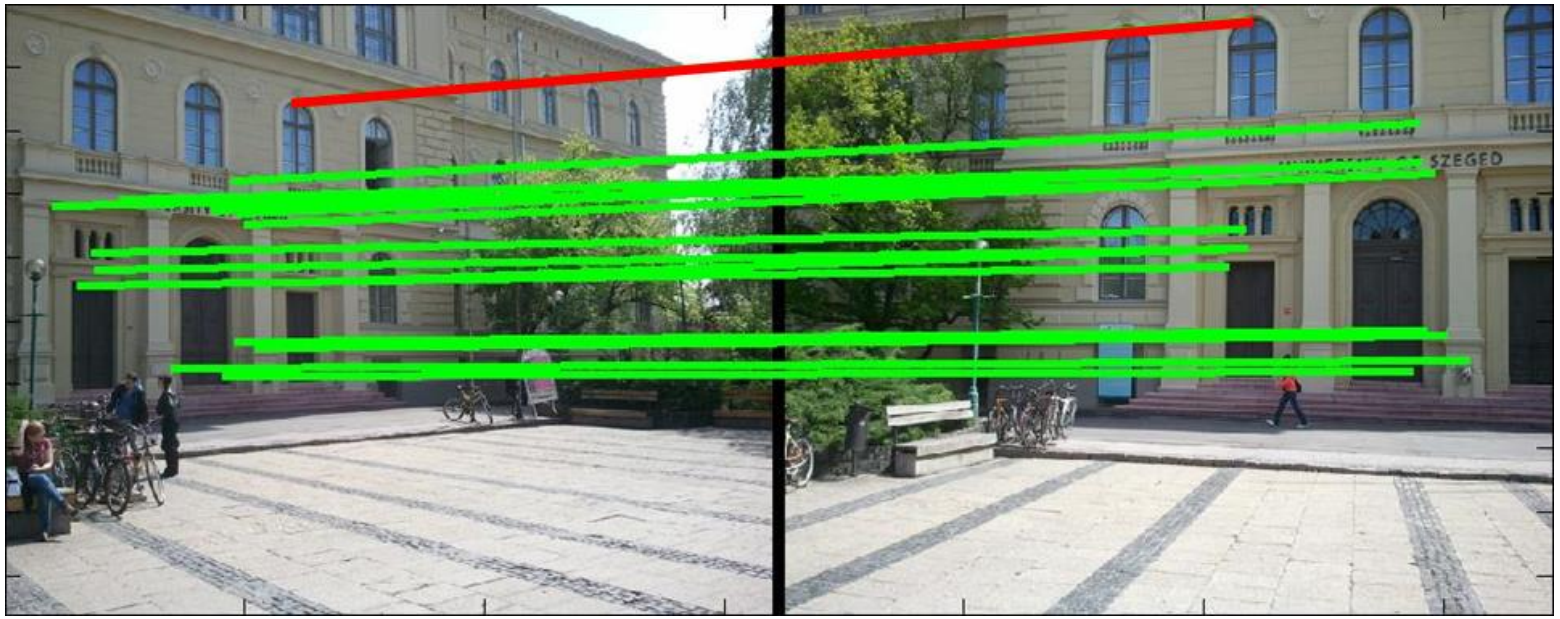
1-nn zöldeket a vörösökhöz

1-nn vöröseket a zöldekhez

Keresztvalidáció

Alkalmazás példa

- Széles bázisvonalú sztereo megfeleltetés-keresés outlier szűréssel
  - Visszavetítési hiba, mint geometriai kényszer
  - Kalibrált kamera szükséges



OpenCV lehetőségek

- **Megvalósított detektorok**
  - Harris
  - SIFT, SURF (szabadalmaztatott, alkalmazására figyelni kell!)
  - FAST, GFTT, MSER, STAR, ORB, BRISK, Dense, SimpleBlob
- **Megvalósított leírók**
  - SIFT, SURF (szabadalmaztatott, alkalmazására figyelni kell!)
  - BRIEF, BRISK, ORB, FREAK
- **Megvalósított párosítók**
  - Teljes keresés (brute force)
  - FLANN-alapú: legközelebbi kulcspont hatékonyabb, közelítő keresése.
  - *k*-nn és rádiusz
    - *k* darab, illetve adott távolságon belül található összes pontpár

OpenCV Harris

- Használata

```
cv::Mat cornerStrength;
cv::cornerHarris( image, cornerStrength,
                  3, //neighbour size
                  3, // aperture size
                  0.01 ); // Harris parameter
cv::Mat harrisCorners;
double threshold = 0.0001;
cv::threshold( cornerStrength, harrisCorners, threshold, 255,
cv::THRESH_BINARY );
```

OpenCV GFTT

- Használata

```
std::vector<cv::Point2f> corners;
cv::goodFeaturesToTrack( image, corners,
            500, // maximum number of corners
            0.01, // quality level
            10 ); // minimum allowed distance between points
```

OpenCV 2 interfész

- Keypoint osztály
  - Kulcspont koordináták, orientáció, oktáv, méret, erősség
  - Nem minden mező definiált minden detektor esetén
- cv::FeatureDetector absztrakt osztály (interfész)

```
void detect( const Mat& image, vector<KeyPoint>& keypoints,
                  const Mat& mask=Mat() ) const;

void detect( const vector<Mat>& images,
             vector<vector<KeyPoint> >& keypoints,
             const vector<Mat>& masks=vector<Mat>() ) const;

virtual void read(const FileNode&);
virtual void write(FileStorage&) const;

static Ptr<FeatureDetector> create( const string& detectorType );
```

OpenCV 2 GFTT

- cv::GoodFeaturesToTrack wrapper osztály

```
// Vector of Keypoints
std::vector<cv::KeyPoints> keypoints;
// Detector construction
cv::GoodFeaturesToTrackDetector gftt(
            500, // maximum number of corners
            0.01, // quality level
            10 ); // minimum allowed distance between points
// Detection
gftt.detect( image, keypoints );
```

OpenCV 2 kulcspont kirajzolás

- FAST detektor + eredmények kirajzolása

```
std::vector<cv::KeyPoints> keypoints;
cv::FastFeatureDetector fast(
            40); // threshold for detection
fast.detect( image, keypoints );

cv::drawKeypoints( image, // original image
            keypoints, // vector of keypoints
            image, // output image
            cv::Scalar(255,255,255), // keypoint color
            cv::DrawMatchesFlags::DRAW_OVER_OUTIMG );

// DRAW_RICH_KEYPOINTS: skála információ megjelenítése is
```

OpenCV 2 leírók

- SURF leíró

```
cv::SurfDescriptorExtractor surfDesc;
cv::Mat descriptors1;
surfDesc.compute( image1, keypoints1, descriptors1);
```

  - **descriptors1**
    - Sorok száma: kulcspontok száma
    - Minden sorban N dimenziós leíró vektor
      - N = 64 SURF alapbeállítás esetén

- Párosítás példa

```
cv::BruteForceMatcher<cv:L2<float>> mtcher;
std::vector<cv::DMatch> matches;
matcher,.match( descriptors1, descriptors2, matches );
```

Élek detektálása (Ismétlés)

- **Gradiens operátorok**
  - Képfüggvény deriváltjának diszkrét közelítése
  - Gradiens: (kép)függvény legnagyobb csökkenésének iránya adott pontban
  - ÉI: gradiensre merőleges irány adott pontban
  - Módszerek
    - Roberts, Prewitt, **Sobel**, Scharr, …
    - Konvolúciós maszkokkal megadhatók

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

- **Laplace operátor**
  - (Kép)függvény másodrendű deriváltjának diszkrét közelítése
  - ÉI: 0-váltások helyén
    - Dupla él detekció, nagyon zajérzékeny
  - Konvolúciós maszkkal megadható

$$\mathbf{D}^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Élek detektálása

- OpenCV-ben Sobel operátor

```
cv::Sobel( image,  // input
    result,        // output
    image_depth,   // image type
    xorder, yorder, // kernel direction: 1,0 (X); 0,1 (Y)
    kernel_size,   // kernel size
    alpha, beta ); // intensity scale and offset

cv::Sobel( image, sobelX, CV_32F, 1, 0 );
cv::Sobel( image, sobelY, CV_32F, 0, 1 );
sobel = abs( sobelX ) + abs( sobelY ); // L1 norm
cv::Mat norm, dir;
cv::cartToPolar( sobelX, sobelY, norm, dir );
```

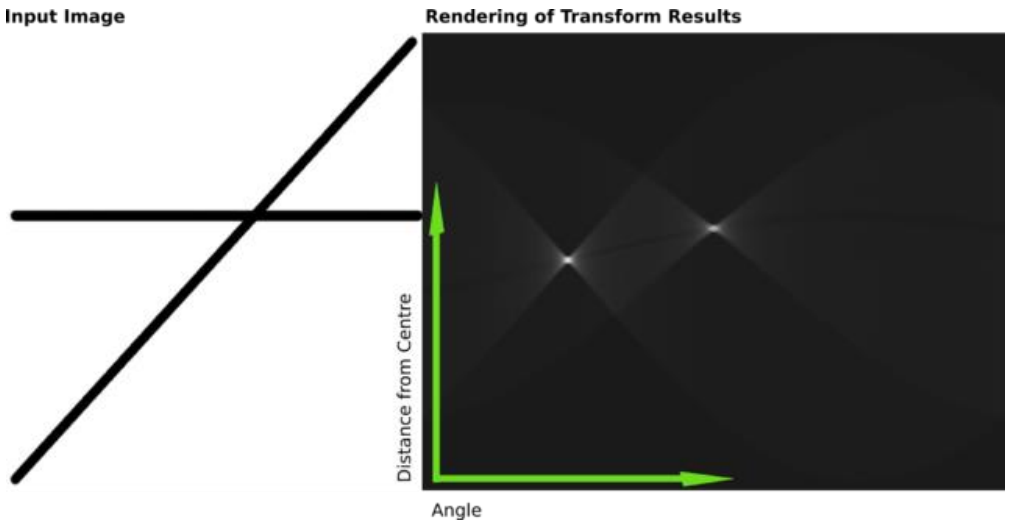
Élek detektálása

- Canny éldetektor
  - Sobel gradiens operátor (más is használható)
  - Hiszterézis küszöbölés
    - Alsó küszöb: minden élpont detektálva legyen + más egyéb pontok
    - Felső küszöb: csak az erős élpontok; szakadozott eredmény
    - Alsó küszöbölt képről csak azon pontok megtartása, amelyek erős komponenseket kötnek össze
  - Nem maximális elnyomás
    - Gradiens irányban csak a maximális érték megtartása
  - OpenCV használat

```
cv::Mat contours;
cv::Canny( image, contours, low_threshold, high_threshold );
```

Hough transzformáció

- Egyenes vonalak detektálása
  - Vonal reprezentáció: $\rho = x \cos\theta + y \sin\theta$
  - Élkeresést kell végrehajtani a képen és binarizálni
  - Minden objektumpontra összegzőtömb növelés (mely egyeneseknek lehet pontja)
  - A maximális tömbértékek adják a legvalószínűbb egyenesek paramétereit (ρ, θ)



Forrás: Wikipedia

Hough transzformáció

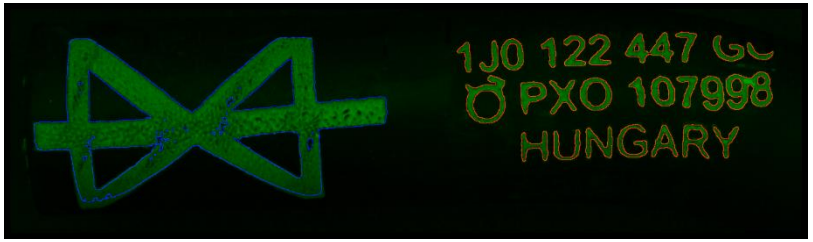
- OpenCV használat

```
std::vector<cv::Vec2f> lines;
cv::HoughLines( binary_im, lines,
    1, PI / 180, // step sizes
    80 ); // minimum number of votes

cv::HoughLinesP( binary_im, lines, 1, PI / 180,
    minLength, maxGap );
```

  - HoughLinesP()
    - Legalább mekkora legyen egy vonalszegmens (minLength)
    - Mekkora lehet a vonalszakaszok közötti távolság (maxGap)
  - Kiterjeszthető az algoritmus adott sugarú kör detektálására
    - $r^2 = (x - x_0) + (y - y_0)$

Komponensek detektálása

- Hasonló intenzitású, összefüggő területek a képen
  - Betűk, számok, szimbólumok, …
- Megoldási lehetőségek
  - Küszöbölés
  - Régiónövelés megadott seed pont(ok)ból
  - Összetettebb szegmentálás
- Összefüggőség
  - 4- vagy 8-szomszédság
  - Összefüggő komponens esetén bármely pontjából bármelyik pontjába el lehet jutni 4- vagy 8-sétával
- Zajszűrés
  - Pl. simítás, morfológiai műveletek

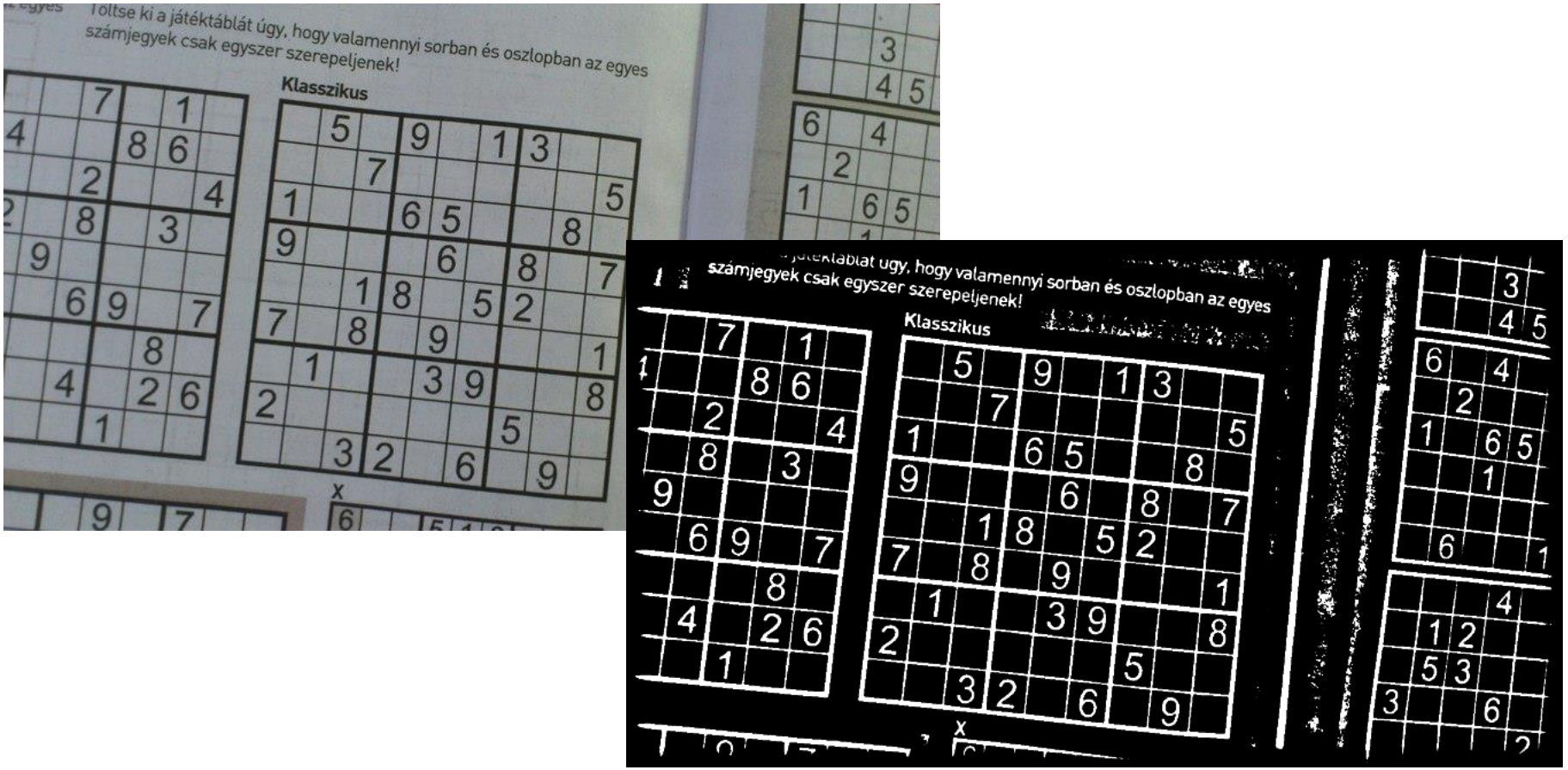


Küszöbölés

- Globális küszöbölés
  - Minden képpontra ugyanazon küszöbérték használata
  - Optimális globális küszöbérték meghatározása
    - Kétcsúcsú hisztogram esetén a kettő között
    - Csak él-képpontok alapján számolva
    - Iteratív kiválasztás: indulás az átlag intenzitásértékből; új küszöb = küszöb alatti és feletti intenzitások átlagának fele;amíg van változás
    - Varianciák számításával (Otsu-módszer)
    - Entrópia alapján
- Adaptív küszöbölés
  - Küszöbérték meghatározása a képpont lokális környezete alapján
  - Niblack: $T(x,y) = m(x,y) + k \cdot s(x,y)$, ahol *T* a küszöb, *m* az átlagintenzitás, s pedig a szórás egy ablakon belül, *k* súly (pl. 0,5)
  - Sauvola: $T(x,y) = m(x,y) \cdot \left[1 + k \cdot \left(\frac{s(x,y)}{R} - 1\right)\right]$, ahol R pl. 128

Küszöbölés

- Niblack küszöbölés eredmény



Küszöbölés

- OpenCV függvények
  - `cv::threshold( input, output, thresh, maxval, type );`
    - type: THRESH_BINARY, THRESH_BINARY_INV, THRESH_TRUNC, THRESH_TOZERO, THRESH_TOZERO_INV
    - THRESH_OTSU: optimális küszöbérték számítása Otsu algoritmussal
  - `cv::adaptiveThreshold( input, output, maxValue, adaptiveMethod, thresholdType, blockSize, C );`
    - adaptiveMethod: ADAPTIVE_THRESH_MEAN_C vagy ADAPTIVE_THRESH_GAUSSIAN_C
    - thresholdType: THRESH_BINARY vagy THRESH_BINARY_INV
    - C: lokális (súlyozott) átlagból kivonandó érték
  - Niblack és Sauvola kód
    - http://answers.opencv.org/question/22941/sauvola-thresholding-for-android/

Komponens kontúrok

- OpenCV

```
std::vector<std::vector<cv::Point>> contours;
cv::findContours( image, contours, mode, method, offset );
```

  - Mode
    - CV_RETR_EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP, CV_RETR_TREE
  - Method
    - CV_CHAIN_APPROX_NONE, CV_CHAIN_APPROX_SIMPLE, CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS
  - Offset
    - ROI-n belüli objektum esetén koordináta eltolási érték adható meg

```
cv::drawContours( image, contours, contourIdx, color,
thickness, lineType, hierarchy, maxLevel, offset );
```

Alakjellemzők

- Fourier-leírók
  - A körvonal 2D koordinátáit komplex számokként értelmezzük

$$\{(x_j, y_j) : j = 0, \ldots, N-1\}$$

$$z_j = x_j + i \cdot y_j, \quad i = \sqrt{-1}$$

  - A komplex számsorozat Fourier transzformáltját használjuk fel

$$c_k = \frac{1}{N} \cdot \sum_{j=0}^{N-1} z_j \cdot e^{\frac{-i \cdot 2 \cdot \pi \cdot k \cdot j}{N}}, \quad k = 0, \ldots, N-1$$

  - A Fourier transzformált magnitúdója eltolás és forgatás invariáns jellemző, függetlenül a kontúrmegadás kezdőpontjától

Alakjellemzők

- Momentumok
  - (p+q) rendű (centrális) momentumok

$$m_{pq} = \sum_{i=0}^{N-1} x_i^p \cdot y_i^q \cdot f_i$$

$$u_{pq} = \sum_{i=0}^{N-1} (x_i - \bar{x})^p \cdot (y_i - \bar{y})^q \cdot f_i$$

$$\bar{x} = \frac{\sum_{i=0}^{N-1} x_i \cdot f_i}{\sum_{i=0}^{N-1} f_i} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{\sum_{i=0}^{N-1} y_i \cdot f_i}{\sum_{i=0}^{N-1} f_i} = \frac{m_{01}}{m_{00}}$$

  - Centrális momentumok: eltolás invariánsak, de nem forgatás invariánsak

Alakjellemzők

- Magasabb rendű momentumok
  - Eltolás és elforgatás invariáns, másod- és harmadrendű momentumok (Hu momentumok: 7 darab)

$$a_1 = u_{20} + u_{02}$$

$$a_2 = (u_{20} - u_{02})^2 + 4 \cdot u_{11}^2$$

$$a_3 = (u_{30} - 3 \cdot u_{12})^2 + (3 \cdot u_{21} + u_{03})^2$$

$$a_4 = (u_{30} + u_{12})^2 + (u_{21} + u_{03})^2$$

$$a_5 = (u_{30} - 3 \cdot u_{12}) \cdot (u_{30} - u_{12}) \cdot [(u_{30} - u_{12})^2 - 3 \cdot (u_{21} + u_{03})] +$$
$$+ (3 \cdot u_{21} - u_{03}) \cdot (u_{21} + u_{03}) \cdot [3 \cdot (u_{30} - u_{12})^2 - (u_{21} + u_{03})^2]$$

...

Alakjellemzők

- Invariáns momentumok
  - Alakjellemzésre egyszerre több momentumot célszerű használni
  - Hasonlóság: momentumvektorok távolsága, korrelációja
  - Eltolás, forgatás és skálázás invariánsak is lehetnek

További alakjellemzők

- Képfeldolgozás kurzus anyagában
  - Határvonal-alapú
    - Normalizált lánckód
    - Kompaktság, cirkularitás, szignatúrák, görbület, energia, …
  - Régió-alapú
    - Befoglaló téglalap, rektangularitás, fő- és melléktengely, excentricitás, átmérő, konvex burok, vetületek, …
    - Topológiai leírás, Euler szám, összefüggőségi fa, váz, …
  - Transzformációs
    - Skála tér, Fourier leíró, …

Alakjellemzők

- Kapcsolódó OpenCV függvények
  - Kontúr reprezentáció alapján
  - boundingRect()
  - minEnclosingCircle()
  - approxPolyDP()
  - convexHull()
  - moments()
  - minAreaRect()
  - contourArea()
  - pointPolygonTest()
  - matchShapes()

Szimbólumok felismerés

- **Feltételezzük**
  - Felismerendő objektum szegmentálva van
  - Alakjellemzők kinyerésre kerültek

- **Szimbólum felismerés**
  - Tanító halmazon osztályok definiálása, jellemzővektorok összeállítása
  - Felismerendő objektumra jellemzővektor számítása
  - Osztályozás
    - Vektorok „hasonlósága” alapján melyik osztályba tartozik
      - Legközelebbi elem, SVM, …
    - Neuronhálók betanítása
    - **Számtalan további lehetőség!**

Betanító és teszt halmazok

- Betanító halmaz
  - Adatokban előforduló változatosságot jól jellemző adatok
  - Objektumok kinyerése, jellemzővektorok előállítása
  - Osztály címkék hozzárendelése

- Teszt halmaz
  - Betanításban nem szereplő adatok
  - Osztályozás sikeressége mérhető
  - Kereszt-validáció
    - N darab partícióba osztás
    - 1 partíció betanító, a többi teszt
    - Dinamikusan végezve a betanító választást

Szimbólumok felismerése

- Nehézségek
  - Nagyon ritkán elegendő egy jellemző érték
  - 1 kép alapján ritkán lehet döntési kritériumot meghatározni
    - Más képeken változhat a méret, a szín, …

Legkisebb távolság osztályozók

- A tanító halmaz legközelebbi elemének a címkéje
- „Távolság” definíciók
  - Euklideszi távolság $((p_1 - q_1)^2 + (p_2 + q_2)^2 + \cdots + p_N + q_N)^2)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{N}(p_i - q_i)^2}$
  - Jellemzővektorok bezárt szöge
  - Korreláció
  - Mahalanobis távolság
    - Figyelembe veszi az egyes jellemző értékek statisztikáit a különböző skálák kiegyenlítésére

$$d_M(P, Q) = \sqrt{(P - Q)^T S^{-1} (P - Q)}$$

    - S: kovariancia mátrix

$$COV = \frac{\sum_{i=1}^{n}(P_i - \mu_i)(Q_i - \mu_i)}{n - 1}$$

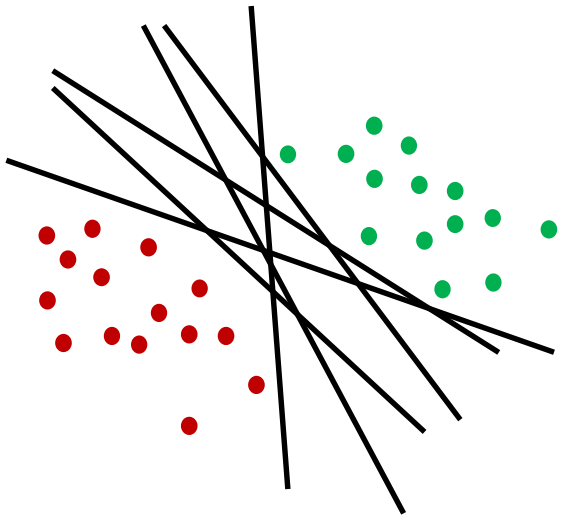
Legkisebb távolság osztályozók

- *k*-legközelebbi szomszéd
  - Nem csak a legközelebbit veszi figyelembe
  - A *k* darab legközelebbi többségi szavazását veszi

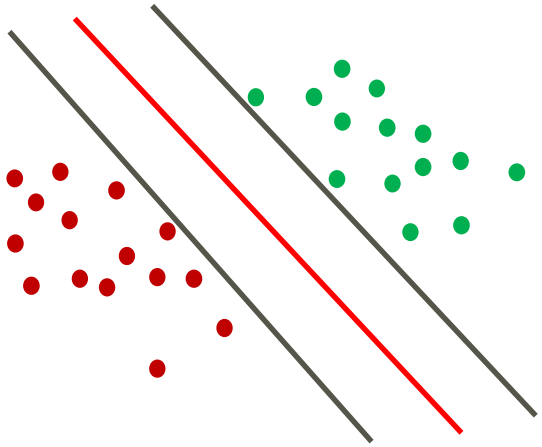
- Legközelebbi centroid osztályozó
  - Az egy osztályba tartozó objektumok jellemző értékeinek átlagához számolja a távolságot (gyorsabb kiértékelés)

Lineáris osztályozó, SVM

- Jellemzők
  - Döntés két osztály között, többféle megoldás is lehet
  - Több osztály esetén páronkénti betanítás kell



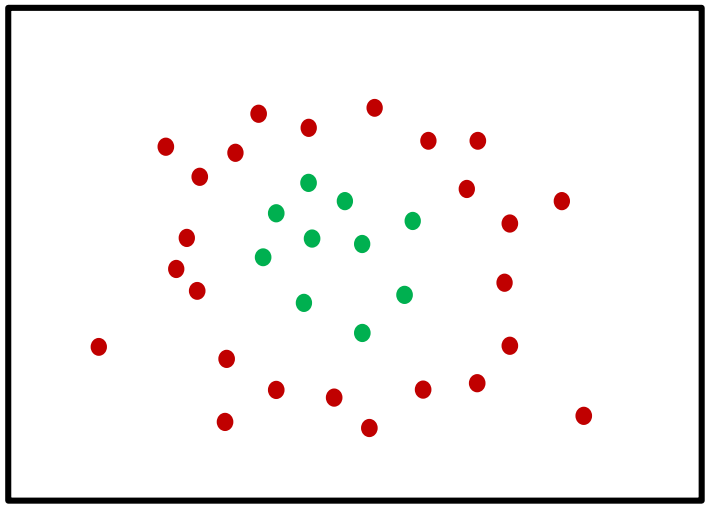
Osztályozók



Osztályozó a
legnagyobb távolsággal

Nem-lineáris osztályozó

- Jellemzővektorok nem-lineáris transzformációjával
  - Lineárisan nem szétválasztható esetre
  - Magasabb dimenzióba kell transzformálni megfelelő kernel függvénnyel, ahol a lineáris szétválasztás megoldható
    - A példában: 2D→3D, ahol a vörös és zöld jellemzők más-más síkra esnek



Együttes osztályozók

- Több osztályozó eredményének egyesítése
  - Sok osztály, sok vektor esetére
- Osztályozó válasz típusok
  - 1-es típus: csak osztály címke
  - 2-es típus: osztály címkék rangsora, de valószínűségek nélkül
  - 3-as típus: osztály címkékhez valószínűség, vagy megbízhatósági szint rendelése
- Elvárások
  - Különböző válasz típusok összerendelése egységes válasszá
  - Az egyesített osztályozás eredménye jobb legyen, mint az osztályozók különválasztva

OpenCV lehetőségek

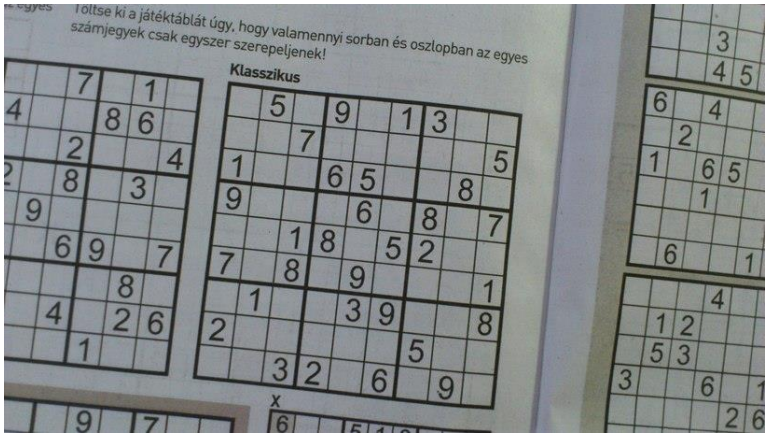
- Erős támogatás a tanuló algoritmusokhoz
  - http://docs.opencv.org/2.4/modules/ml/doc/ml.html
  - Statisztikai modellek
  - Bayes-osztályozó
  - K-nn osztályozó
  - Neurális hálózatok: MLP (Multi-layer perceptrons)
  - SVM
  - Döntési fák
  - Boosting
  - E-M (expectation maximization)
  - ...

Esettanulmányok

- Sudoku megoldó
  - Kerekes Gábor diplomamunkája
  - Realtime Webcam Sudoku Solver (CodeProject)
- Franciakártya típus (káró, treff, kőr, pikk) felismerése
  - Mészáros Balázs szakdolgozata
- Gépkocsi típus és modell felismerés
  - Varjas Viktor diplomamunkája

Sudoku megoldó

- Részfeladatok
  - Tábla helyének megkeresése
  - Tábla típusának meghatározása
  - Orientáció detektálása és korrekciója
  - Számjegy régiók detektálása
  - Számjegyek felismerése
    - Zónázás
    - Oldalirányú profilok
  - Sudoku feladat megoldása
- Leírás
  - CodeProject



Kártyatípus felismerés

- Részfeladatok
  - Kamera élőkép beolvasása
  - Előfeldolgozás: háttérlevonás, Niblack küszöbölés
  - Alakjellemzők: Hu-momentumok
  - Osztályozás: neuronhálóval
    - 7 bemeneti érték
    - 4 kimenet
    - Rejtett neuronok száma: felismerendő objektum osztályok kétszerese
    - Betanítással (Matlab toolbox)

Gépkocsi gyártó és típus felismerés

- **Részfeladatok**
  - ROI detekció: a rendszámtábla környezete informatív
    - Rendszámtábla detekció
    - ROI normalizálás (orientáció, méret)
  - SMG (Square Mapped Gradient) jellemző számítása
  - Osztályozás
    - Korreláció, mint távolság
    - 3-NN többségi szavazás

Gyártó és típus felismerés →

Peugeot 307

Tesztelés

- Tesseract OCR
  - **OCR Test** alkalmazás a Google Play áruházban
    - Forráskód letölthető
    - https://github.com/rmtheis/tess-two
    - https://github.com/rmtheis/android-ocr
- Doctor Mole alkalmazás
  - Anyajegyek elemzése
- Android Application Programming with OpenCV könyv
  - 5. fejezet: ismert kép keresése kamera élőképen
- OpenCV tutorialok
  - http://docs.opencv.org/doc/tutorials/features2d/table_of_content_features2d/table_of_content_features2d.html
  - Android OpenCV „Color Blob Detection” példaprogramja
    - Egészítsük ki úgy, hogy csak egy adott intervallumba eső méretű komponenseket tartson meg!

MOBIL KÉPELEMZÉS ÉS GRAFIKA

MOBIL 3D GRAFIKA ANDROID PLATFORMON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

3D grafikai csomagok

- Direct3D, DirectX
  - Csak Microsoft platformon
- OpenGL
  - Silicon Graphics: IRIS GL (zárt kód)
  - → OpenGL (1992)
  - Nyílt szabvány; konzorciumi kezelésben (ma: Khronos Group)
  - C nyelvű függvénykönyvtár asztali PC-kre
- Unity 3D
  - Multiplatform környezet, „játékmotor”
  - Magasabb szintű modellezés
  - A választott platformon elérhető API-t használja
    - OpenGL, DirectX, …

Új grafikus API-k

- Metal
  - iOS és MacOS készülékekre (Apple)
  - Hatékonyabb, mint az OpenGL
- Vulkan
  - OpenGL melletti új nyílt szabvány
    - Alacsonyabb szintű hozzáférés a hardverhez
    - Hatékonyabb, de több programozói munkát igényel
  - Asztali, mobil és egyéb platformokra
    - Samsung Galaxy S7 már implementálja
  - 2016. februárjában jelent meg az első verzió a szabványból

OpenGL ES

- OpenGL ES (Embedded Systems)
  - Nyílt szabvány; 2000-es évek eleje
  - OpenGL redukált változata beágyazott rendszerekre
  - EGL: közös interfész réteg
  - Sok platform támogatja
    - Android, iOS, Java ME, Symbian, QNX, PlayStation, Blackberry, webOS, ...
    - WebGL (3D grafika böngészőkben) is erre épül!

- Specifikáció
  - The Standard for Embedded Accelerated 3D Graphics

OpenGL ES verziók

- **OpenGL ES 1.0**
  - OpenGL 1.3 verzió alapján (ezt tanítjuk BSc-n)
  - **Embedded profilok**
    - **Common lite profile**: csak fixpontos adattípus (nincs lebegőpontos)
      - Ha nincs lebegőpontos CPU támogatás (beágyazott környezetben előfordul)
    - **Common profile**: fixpontos és lebegőpontos adattípus is van
  - **Főbb különbségek az asztali változathoz képest**
    - Nincs közvetlen modellezési lehetőség
      - glBegin() … glEnd() helyett tömbökkel kell dolgozni
    - Fixpontos számábrázolás megjelenése
    - Több funkció nem érhető el
      - Csak háromszögek használhatók fel!
      - texgen; line stipple; poligon stipple; polygon_mode
      - Bitmap műveletek; előtér és összegző pufferek
      - Megjelenítési listák és feedback mód
      - Hátlapok anyagjellemzői; felhasználói vágó síkok; …

OpenGL ES verziók

- OpenGL ES 1.1
  - OpenGL 1.5 verzió alapján
  - Újdonságok
    - Multitexture támogatás; mipmap generálás
    - VBO (vertex buffer object)
    - Állapotlekérések; felhasználói vágósíkok
- OpenGL ES 2.0 (2007. március)
  - OpenGL 2.0 verzió alapján
  - Programozható műveletsor megjelenése
    - Vertex és fragment shader-ek
    - Nincs alapértelmezett transzformáció sorozat és megvilágítás típusok!
    - Nagyobb programozói szabadság – több munka!
  - Nincs visszafelé kompatibilitás az 1.1 verzióval!

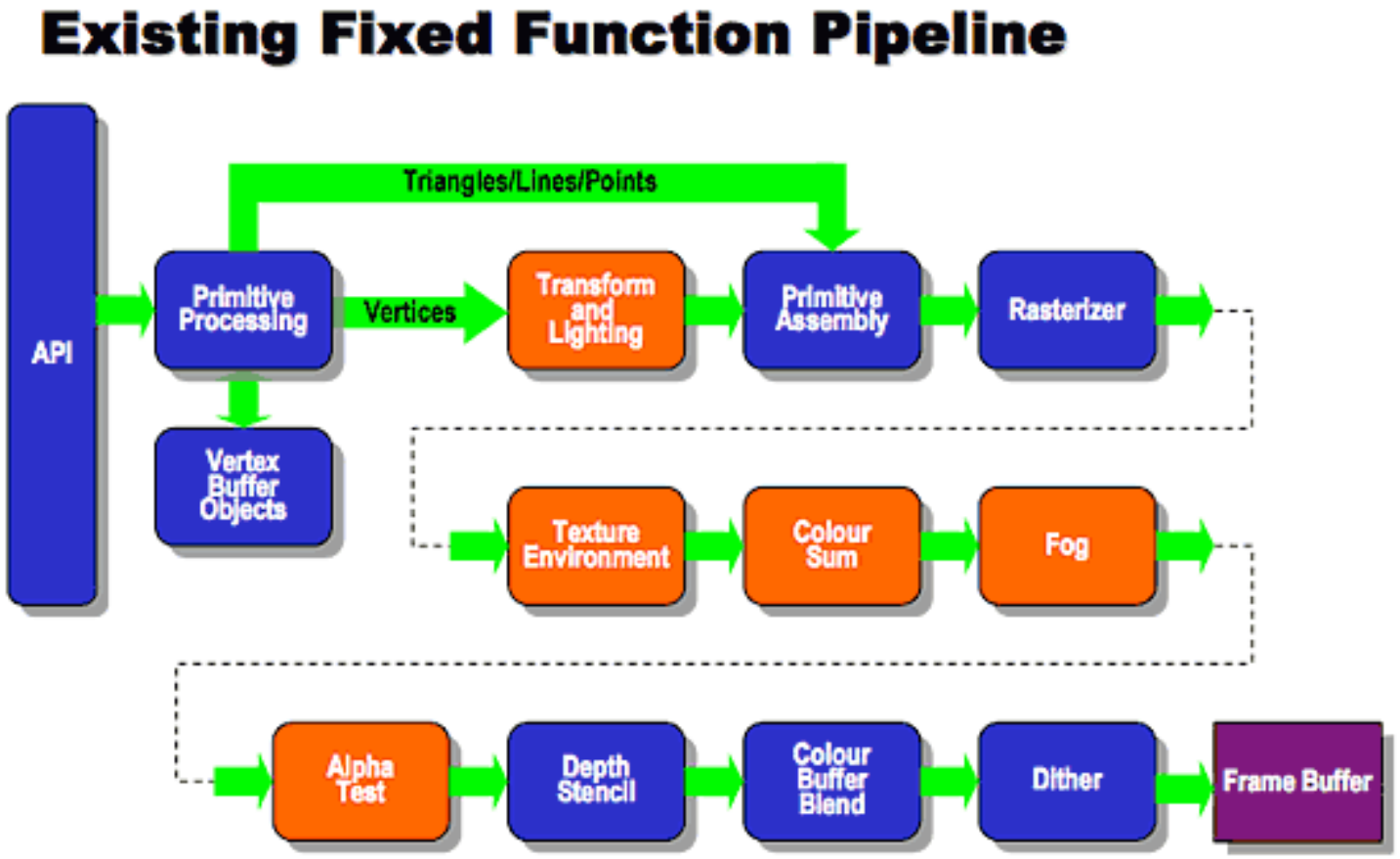
OpenGL ES verziók

- OpenGL ES 3.0 (2012. augusztus)
  - Teljes kompatibilitás az OpenGL 4.3 verzióval
  - Visszamenőlegesen kompatibilis az OpenGL ES 2.0 verzióval
  - Újdonságok
    - Új textúra formátumok, tömörítés
    - Hatékonyabb megjelenítés
    - Új GLSL ES shader programozási nyelv
- OpenGL ES 3.1 (2014. március)
  - Indirekt rajzoló parancsok, shader frissítések
  - Visszamenőlegesen kompatibilis az OpenGL ES 3.0 és 2.0 verziókkal
- OpenGL ES 3.2 (2015. augusztus)
  - Geometria és tesszelláló shader-ek, lebegőpontos render, ...

Modellezés elve

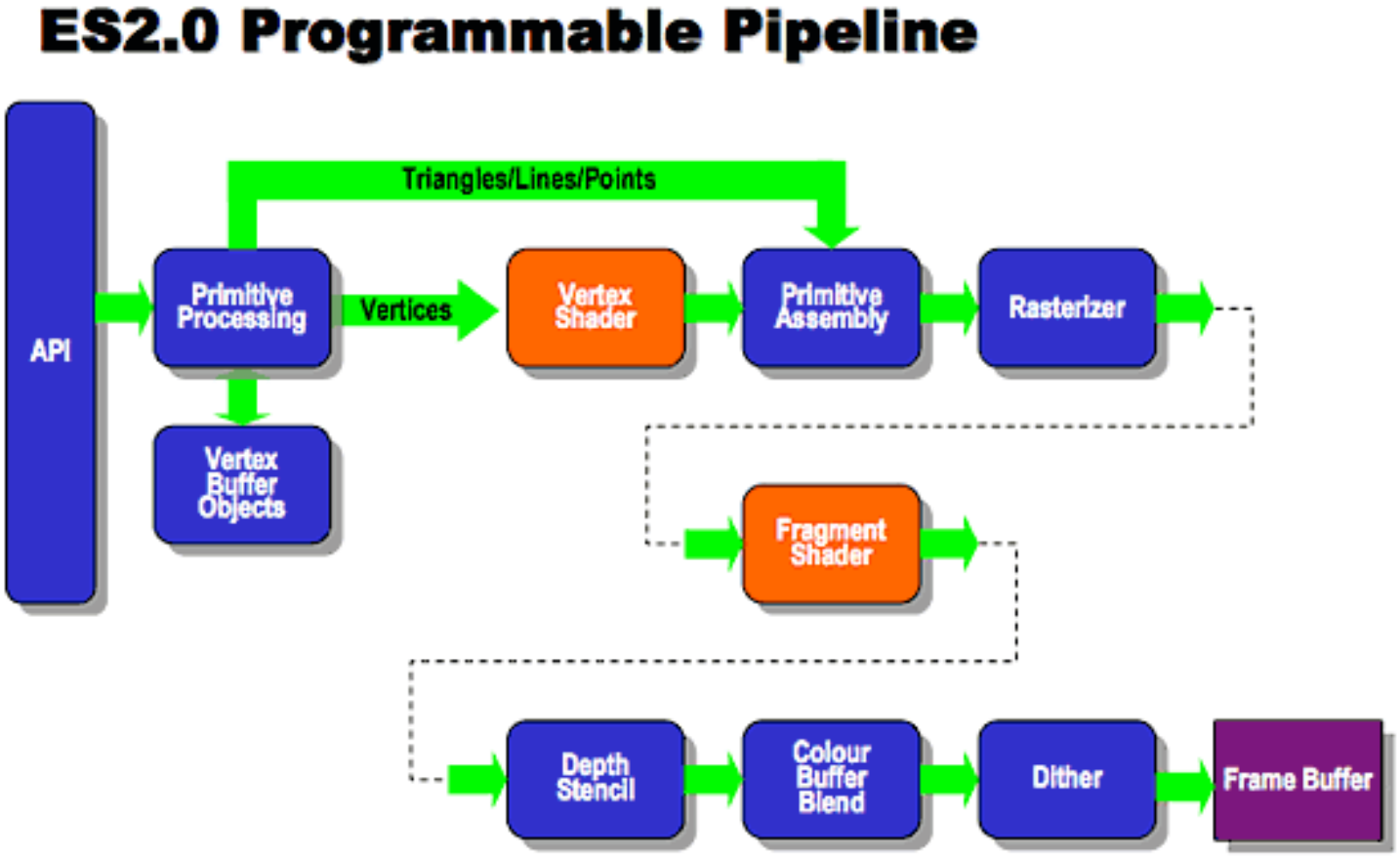
- 3D világ koordináta-rendszerben
  - Jobb sodrású, X-Y-Z rendszer
- A térbeli objektumokat felszínük síkidomokkal való közelítésével modellezzük
  - Csak az XY síkra modellezve 2D grafika is készíthető
  - A síkidomok belső területe színezhető, mintával kitölthető
  - Síkidom takarások kezelhetők
- Elhelyezhetünk fényforrásokat
  - Anyag és fény kölcsönhatása adja a végleges színt
- Kamera vetít 3D-ből 2D-be
  - Megadható térbeli pozíció, vetítési irány, vetítés típusa
- 2D vetített eredmény megjelenítése ablakban

Rögzített műveletsor



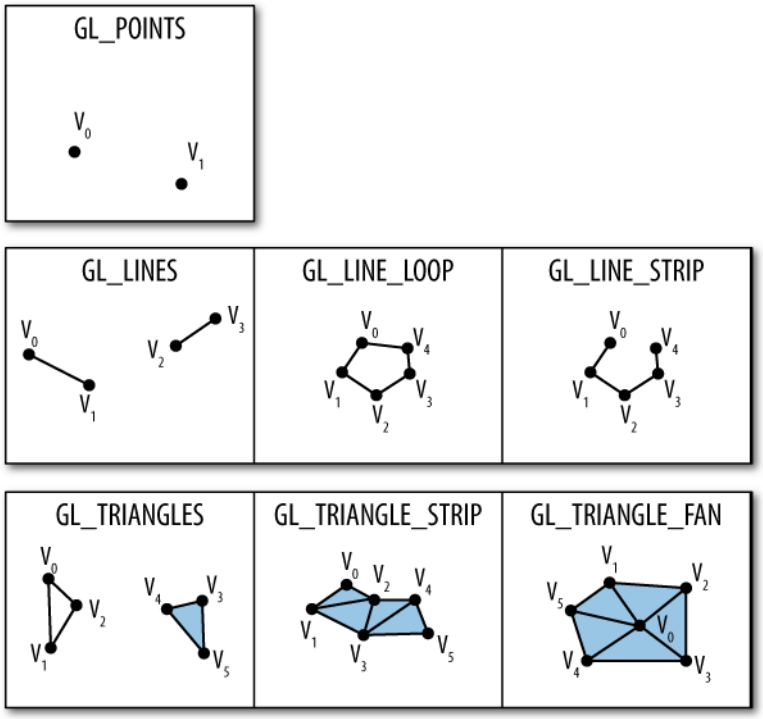
Kép forrása: Khronos Group

Programozható műveletsor



Kép forrása: Khronos Group

Grafikus primitívek



Kötelező

- Csúcspont koordináták

Opcionális

- Csúcspont színek
- Csúcspont normálvektorok
- Csúcspont textúra koordináták

Minden csúcspontra meg kell adni őket, így ismétlődések elő fognak fordulni.

Tömbben adjuk meg őket

- Attribútumonként lehet külön tömb.
- Egy tömbben is elhelyezhetők.

Alapértelmezett vetítés

- Koordináták a [-1, 1] tartományban X, Y, és Z esetén is
- A kamera a Z-tengely negatív irányába néz, párhuzamosan vetít

GLSL nyelv (GPU programozás)

- Kötelező megírnunk a csúcspontok és a síkidom belső pontok kezelésére szolgáló programrészeket!
  - Shader programok, árnyalók
  - Csúcspont (vertex) és fragmens (fragment) árnyalók
- A shaderek ESSL/GLSL nyelven íródnak
  - Embedded System / OpenGL Shading Language
    - Alacsony szintű, C-szerű szintaktika
- A programunk futása közben fordítódnak és töltődnek a GPU-ra
  - A GPU párhuzamosan futtatja őket

Shader-ek

- Vertex shader
  - Csúcspontokra vonatkozó transzformációk
    - Geometriai transzformációk alkalmazása
    - Megvilágítási egyenletek kiértékelése
    - …

Legegyszerűbb megvalósítás: megkapott koordináta továbbadása változatlanul.

- Fragmens shader
  - Poligonok belső pontjaira vonatkozó transzformációk
    - Színezés, textúrázás, fényhatások, …

Legegyszerűbb megvalósítás: megkapott szín továbbadása változatlanul.

Shader specifikus típusok

- Alaptípusok
  - `float, bool, int`
  - `vec2, vec3, vec4`
  - `bvec2, bvec3, bvec4`
  - `ivec2, ivec3, ivec4`
  - `mat2, mat3, mat4`
  - `sampler1D, sampler2D, sampler3D, samplerCube, sampler1DShadow, sampler2DShadow`

Vektor- és mátrixművelet specifikus típusok.

Textúra mintavételezéshez.

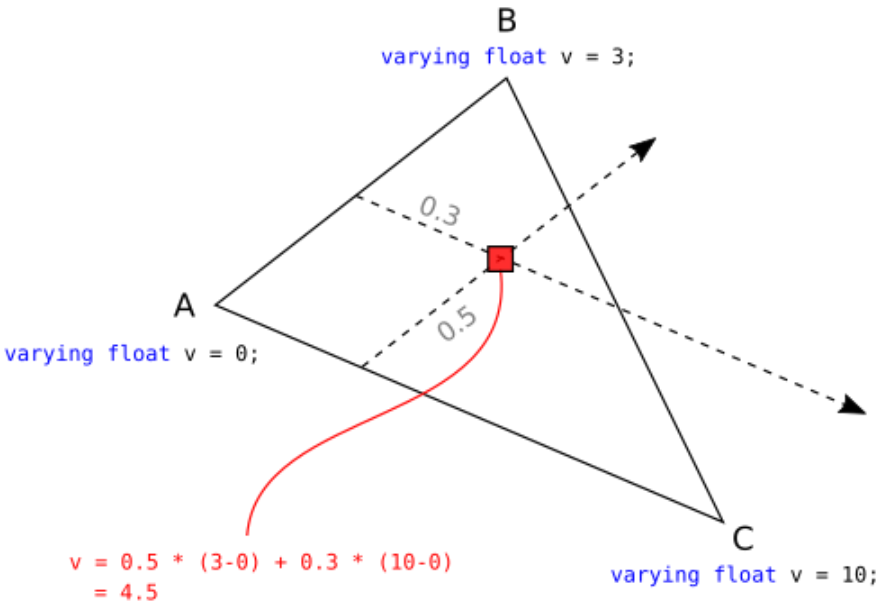
- Struktúrák
  - `struct`

Shader speciális módosító jelzők

- `const`
  - Shader fordítási időben definiált konstans.
- `attribute`
  - Csúcspontonkénti adatok (pl. csúcspont tömbök) tárolása.
  - Csak a vertex shaderben érhető el.
- `uniform`
  - Primitívre vonatkozóan egységesen használt, konstans érték.
- `varying`
  - Shader interpolált kimenete fragmes shader felé. Vertex shaderben írható, fragmens shaderben csak olvasható.
- `sampler`
  - Textúrák mintavételezéséhez.

varying változók

- Vertex shader által a csúcsponthoz rendelt **`varying`** adatok bilineáris interpoláltja lesz elérhető a fragmens shaderben az aktuális belső pontra vonatkozóan



Forrás: https://stackoverflow.com/questions/17537879/in-webgl-what-are-the-differences-between-an-attribute-a-uniform-and-a-varying

Shader előre definiált változók

- Vertex shader
  - **`vec4 gl_Position`**
    - Vertex shader által továbbadott csúcspont pozíció érték, például geometriai transzformáció alkalmazása után (kimenet)

- Fragmens shader
  - **`vec4 gl_FragCoord`**
    - Fragmens pont ablak koordinátái (input)
  - **`bool gl_FrontFacing`**
    - Előlapi pont-e? (input)
  - **`vec4 gl_FragColor`**
    - Fragmens eredmény színe (kimenet)

OpenGL ES Android-on

- OpenGL ES 1.0
  - android.opengl (API Level 1; Android 1.0)
  - Rögzített műveletsor
- OpenGL ES 1.1
  - android.opengl.GLES11 (API Level 4; Android 1.6)
  - Rögzített műveletsor
- OpenGL ES 2.0
  - android.opengl.GLES20 (API Level 8; Android 2.2)
  - Programozható műveletsor (shader-ek)
- OpenGL ES 3.0
  - (API Level 18; Android 4.3 – ha a gyártó támogatja)
- OpenGL ES 3.1
  - (API Level 21; Android 5.0 – ha a gyártó támogatja)
- Elérhető Java API-n vagy NDK-n keresztül (C nyelven)

Android OpenGL ES információk

- OpenGL ES (Android Developer)
- Grafika OpenGL ES 2.0-val (Android Developer)
  - OpenGL ES környezet beállítása
  - Alakzatok definiálása
  - Alakzatok rajzolása
  - Projekciós és kamera nézeti transzformációk megadása
  - Animáció
  - Reagálás érintés eseményekre
- Rick Boyer. **Android 9 Development Cookbook.** Third Edition. Chapter 11. Packt Publishing, 2018.
- Kevin Brothaler. **OpenGL ES 2 for Android. A Quick Start Guide.** The Pragmatic Programmers, LLC, 2013.
- Fountain OpenGL Application Walkthrough (CodeProject)
  - http://www.codeproject.com/Articles/269727/Fountain-OpenGL-Application-walkthrough

Android OpenGL ES

- Fő irányelvek
  - `<uses feature>` a `Manifest.xml` fájlban
    - `<uses-feature android:glEsVersion="0x00010000" />`
    - `<uses-feature android:glEsVersion="0x00020000" android:required="true" />`
  - Rajzolás `SurfaceView`-ra, összekapcsolás OpenGL ES kontextussal
  - Modellezés külön szálon fusson!

- Megvalósítás
  - Android segédosztályokkal (`GLSurfaceView` és társai)
    - Egyszerűbb, de kevesebb kontroll
  - Kontextus létrehozása manuálisan
    - Összetettebb, nem tárgyaljuk

GLSurfaceView + Renderer

- **`GLSurfaceView`**
  - `GLSurfaceView` osztály példányosítása (interakció nélküli esetben)
  - Leszármaztatás (ha szükséges interakció)
  - Az objektumpéldány `setRenderer()` függvényének hívása, amelynek átadjuk a saját `Renderer` modellező osztályunk objektumpéldányát
- **`GLSurfaceView.Renderer`** interfész implementálása
  - `onSurfaceCreated(GL10 unused, EGLConfig config)`
    - Kezdeti beállítások
  - `onSurfaceChanged(GL10 gl, int width, int height)`
    - Vetítő mátrix definiálása a beérkező méretek alapján
  - `onDrawFrame(GL10 gl)`
    - Modellezés helye
    - `FloatBuffer`, `ByteBuffer`, `IntBuffer` osztályok: Java tömbből OpenGL ES tömbökbe konverzió!
  - `onPause()`, `onResume()` függvények kezelése!
    - A `GLSurfaceView` objektumpéldány `onPause()`, `onResume()` függvényeit is meg kell hívni, mivel a modellezés külön szálon fut!

SetupOpenGL példaprogram

```
package com.packtpub.setupopengl;

import android.content.Context;
import android.opengl.GLES20;
import android.opengl.GLSurfaceView;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
```

08_AndroidBookOpenGLSamples.zip

Kontextus létrehozása

```
public class MainActivity extends AppCompatActivity {
    private GLSurfaceView mGLSurfaceView;

    class GLRenderer implements GLSurfaceView.Renderer {
        public void onSurfaceCreated(GL10 unused, EGLConfig config) {
            GLES20.glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
        }
        public void onDrawFrame(GL10 unused) {
            GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);
        }
        public void onSurfaceChanged(GL10 unused, int width, int height) {
            GLES20.glViewport(0, 0, width, height);
        }
    }

    class CustomGLSurfaceView extends GLSurfaceView {
        private final GLRenderer mGLRenderer;

        public CustomGLSurfaceView(Context context){
            super(context);
            setEGLContextClientVersion(2);
            mGLRenderer = new GLRenderer();
            setRenderer(mGLRenderer);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mGLSurfaceView = new CustomGLSurfaceView(this);
        setContentView(mGLSurfaceView);
    }
}
```

ShapesWithOpenGL példaprogram

```
private Triangle mTriangle;

class GLRenderer implements GLSurfaceView.Renderer {
        public void onSurfaceCreated(GL10 unused, EGLConfig config) {
            mTriangle = new Triangle();
            GLES20.glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
        }
        public void onDrawFrame(GL10 unused) {
            GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);
            mTriangle.draw();
        }
        public void onSurfaceChanged(GL10 unused, int width, int height) {
            GLES20.glViewport(0, 0, width, height);
        }
    }
```

08_AndroidBookOpenGLSamples.zip

2D háromszög rajzolása

```
private final String vertexShaderCode = "attribute vec4 vPosition;" +
            "void main() {" +
            "  gl_Position = vPosition;" +
            "}";

   private final String fragmentShaderCode = "precision mediump float;" +
            "uniform vec4 vColor;" +
            "void main() {" +
            "  gl_FragColor = vColor;" +
            "}";
```

Shader kódok megadása egyszerű sztringként.
Le kell fordítani futás közben a használathoz!

A Java vagy C++ kódból át kell adni a vPosition és a vColor paramétereket.
Ezek 4 elemű vektorok, vagy tömbök is lehetnek.

2D háromszög rajzolása

```
final int COORDS_PER_VERTEX = 3;
   float triangleCoords[] = {
           0.0f,  0.66f, 0.0f,
           -0.5f, -0.33f, 0.0f,
           0.5f, -0.33f, 0.0f
   };

   float color[] = { 0.63f, 0.76f, 0.22f, 1.0f };

   private final int mProgram;
   private FloatBuffer vertexBuffer;
   private int mPositionHandle;
   private int mColorHandle;
   private final int vertexCount = triangleCoords.length / COORDS_PER_VERTEX;
   private final int vertexStride = COORDS_PER_VERTEX * 4;
```

2D háromszög rajzolása

```
public Triangle() {
        int vertexShader = loadShader(
                GLES20.GL_VERTEX_SHADER,
                vertexShaderCode);
        int fragmentShader = loadShader(
                GLES20.GL_FRAGMENT_SHADER,
                fragmentShaderCode);
        mProgram = GLES20.glCreateProgram();
        GLES20.glAttachShader(mProgram, vertexShader);
        GLES20.glAttachShader(mProgram, fragmentShader);
        GLES20.glLinkProgram(mProgram);

        ByteBuffer bb = ByteBuffer.allocateDirect(
                triangleCoords.length * 4);
        bb.order(ByteOrder.nativeOrder());

        vertexBuffer = bb.asFloatBuffer();
        vertexBuffer.put(triangleCoords);
        vertexBuffer.position(0);
    }
```

Shader program létrehozása

Háromszög csúcspontok bájt tömbbe helyezése.
Java tömb közvetlenül nem adható át a shader kódnak!

2D háromszög rajzolása

```
public int loadShader(int type, String shaderCode){
        int shader = GLES20.glCreateShader(type);
        GLES20.glShaderSource(shader, shaderCode);
        GLES20.glCompileShader(shader);
        return shader;
    }

    public void draw() {
        GLES20.glUseProgram(mProgram);
        mPositionHandle = GLES20.glGetAttribLocation(mProgram, "vPosition");
        GLES20.glEnableVertexAttribArray(mPositionHandle);
        GLES20.glVertexAttribPointer(mPositionHandle,
                COORDS_PER_VERTEX,
                GLES20.GL_FLOAT, false,
                vertexStride, vertexBuffer);
        mColorHandle = GLES20.glGetUniformLocation(mProgram, "vColor");
        GLES20.glUniform4fv(mColorHandle, 1, color, 0);

        GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, vertexCount);
        GLES20.glDisableVertexAttribArray(mPositionHandle);
    }
```

Shader kód fordítása

vertexBuffer koordináták átadása a shadernek.

Szín átadása.

Rajzolás

Program továbbfejlesztése

- `ProjectionAndCamera` példaprogram
  - Kamera vetítési paramétereinek megadása
  - 4x4 méretű projekciós mátrix használata
- `RotateWithUserInput` példaprogram
  - Érintés esemény hatására forgatás
  - 4x4 méretű transzformációs mátrix használata
- `CreatingMovement` példaprogram
  - Objektum mozgatása
- Feladat
  - Egészítsük ki négyszög rajzolására!

Geometriai transzformációk mátrix alakban történő reprezentációjával a számítógépes grafika kurzusokon találkozhattak.

Geometriai transzformációk

- Homogén koordináták
  - Dimenziónövelés 3D→4D
  - Valós térbeli koordinátákat úgy kapunk, hogy a negyedik elemmel elosztjuk az első hármat
  - Részletek a Számítógépes grafika előadás anyagban
- Mátrixok
  - Lineáris transzformációk reprezentációjára képesek
  - Összeadás, kivonás
  - Szorzás
    - Nem kommutatív, a sorrend nem cserélhető fel!

Egység mátrix

- A homogén koordinátás térbeli pontot nem mozgatja

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Eltolás

- *x*, *y*, és *z* koordinátákhoz eltolási érték adása

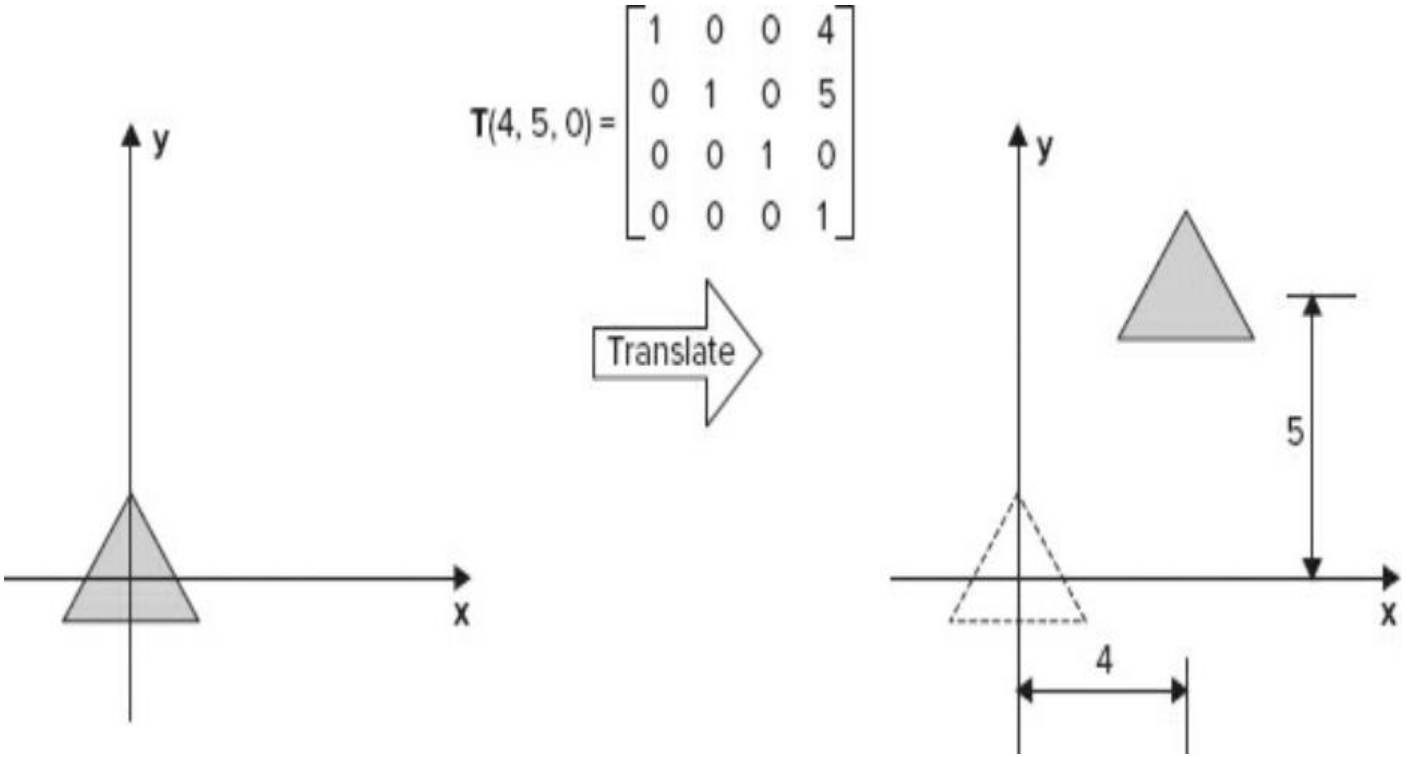
$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$Tp = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \\ 1 \end{bmatrix}$$

Eltolás

- Példa



Forgatás

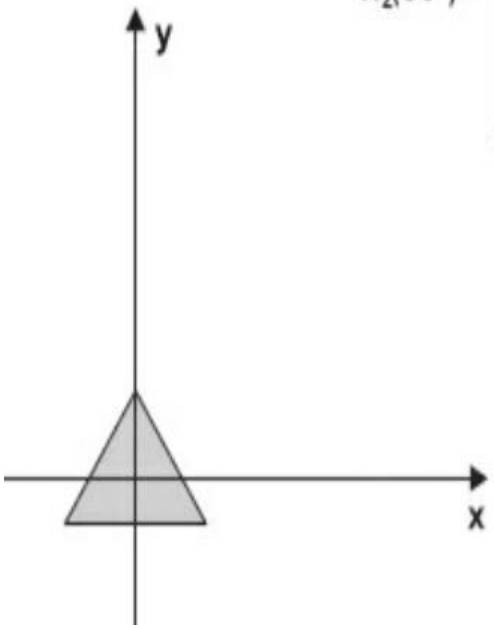
- Tengelyek mentén

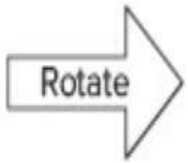
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

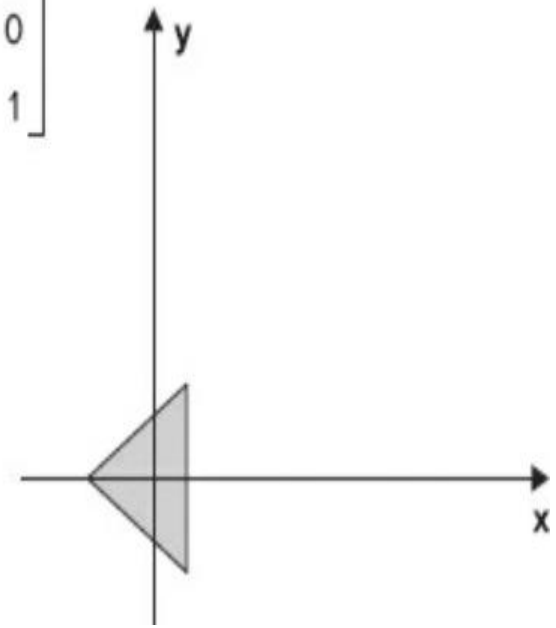
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(90°) = \begin{bmatrix} \cos 90° & -\sin 90° & 0 & 0 \\ \sin 90° & \cos 90° & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





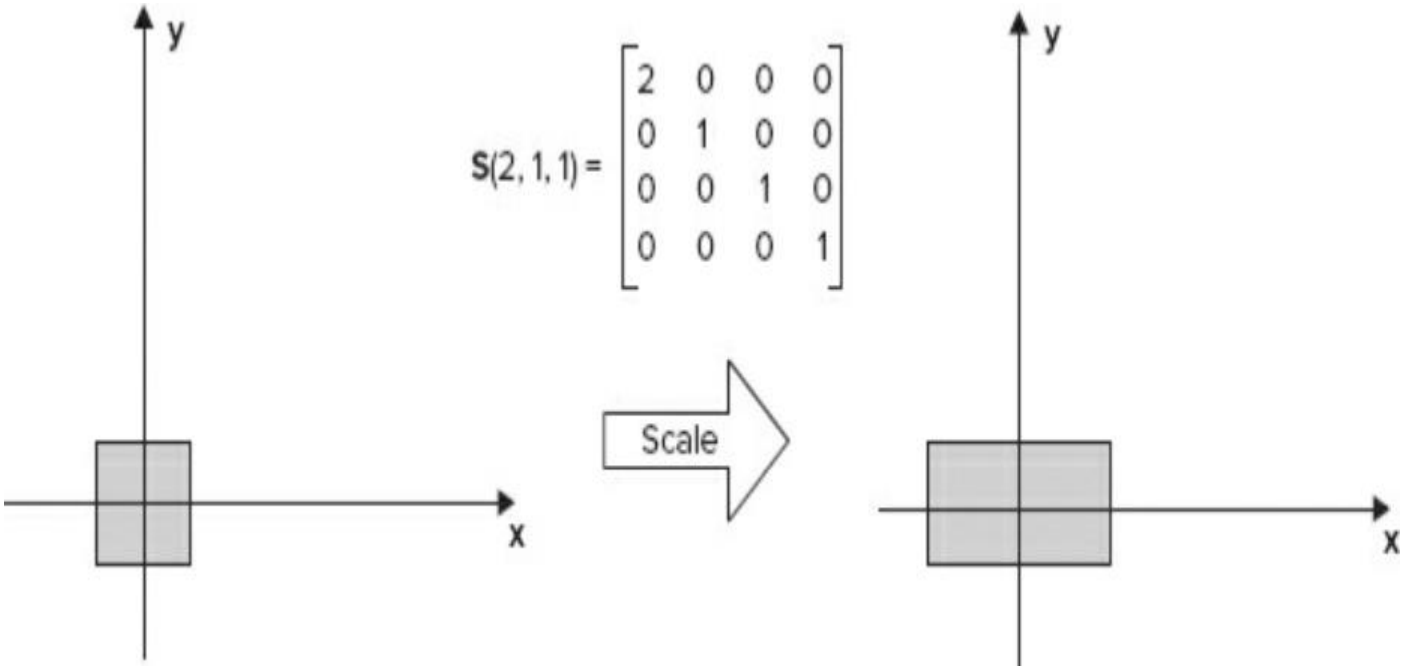


Skálázás

- Tengelyek mentén

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$S(2,1,1) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Tulajdonságok

- A mátrix műveletek **kompozíciója** képezhető
  - Mátrixszorzás segítségével
    - Csoportosítható (asszociatív)
    - Nem kommutatív (sorrend nem cserélhető fel)
- **Mátrix vizsgálata** (*k*-dimenzióban)
  - *t*: eltolás
  - det(*R*)=1, tiszta forgató (merev-test)
  - det(*R*)≠1, skálázás, nyírás is
  - det(*R*)<0, tükrözés is
  - *p*: perspektív transzformáció esetén nem 0

$$T = \begin{pmatrix} R_{k\times k} & t_{k\times 1} \\ p_{1\times k} & 1 \end{pmatrix}$$

ModelView transzformáció

- Térbeli világ koordináta-rendszer
- Ebben elhelyezkedő
  - objektum pontok
  - kamera
- Az objektumok térben pozícionálhatók (*Model* transzformáció)
  - Eltolás, forgatás, skálázás sorozattal
- Kamera térben elhelyezhető (*View* transzformáció)
  - Pozíció, vetítési irány, orientáció
- Vetítés eredménye
  - A kamera hogyan látja a színteret?

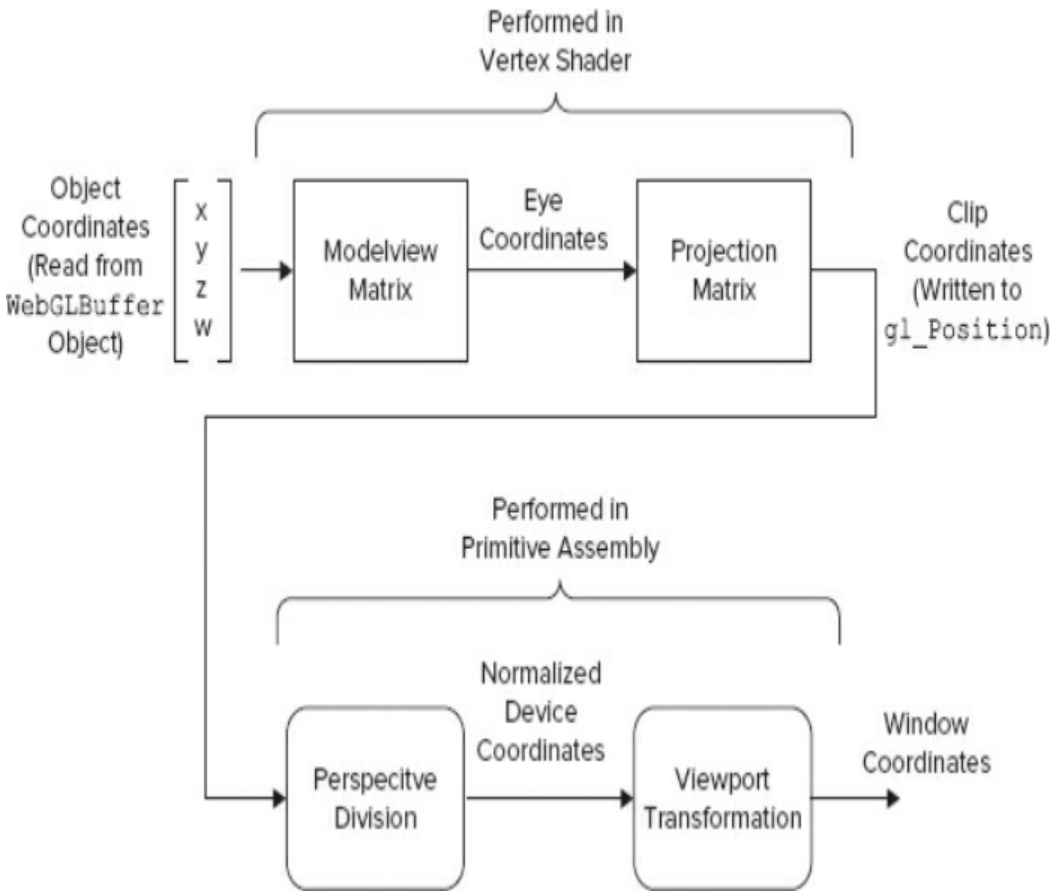
ModelView transzformáció

- Szem (eye) koordináták
  - Az objektum pontok a kamera saját koordináta-rendszerében hol helyezkednek el.
- A végeredmény szempontjából mindegy, hogy a kamera mozog a színtérben, vagy a színtér objektumai mozognak a kamerához képest.
  - Úgy adjuk meg, ahogy kényelmesebb.
  - A *Model* és a *View* egy közös 4x4-es mátrixban megadható. Ennek neve *ModelView*.

Vetítés (projekció)

- 4x4-es mátrixszal megadható
- Perspektív vagy párhuzamos vetítések az általánosak
- Jellemzően segédfüggvényekkel definiáljuk

Teljes műveletsor



android.opengl.matrix

- Android segédkönyvtár OpenGL ES mátrix műveletekhez
  - https://developer.android.com/reference/android/opengl/Matrix
- `Matrix`
  - 16 elemű, oszlop sorrendben megadott `float` vektor egy 4x4 mátrix reprezentálására
- Koordináta
  - 4 elemű `float` oszlopvektor
- Műveletek
  - Elemi mátrix műveletek
    - `invertM()`, `length()`, `multiplyMM()`, `multiplyMV()`, `transpose()`
  - Kamera funkciók
    - `frustumM()`, `orthoM()`, `perspectiveM()`, `setLookAtM()`
  - Geometriai transzformációk
    - `rotateM()`, `scaleM()`, `translateM()`, `setRotateM()`, `setIdentityM()`, `setRotateEulerM()`

Transzformáció átadása a shader-eknek

```
private final float[] mMVPMatrix = new float[16];
private final float[] mProjectionMatrix = new float[16];
private final float[] mViewMatrix = new float[16];

...

float ratio = (float) width / height;
Matrix.frustumM(mProjectionMatrix, 0, -ratio, ratio, -1, 1, 3, 7);

...

Matrix.setLookAtM(mViewMatrix, 0, 0, 0, -3, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
Matrix.multiplyMM(mMVPMatrix, 0, mProjectionMatrix, 0, mViewMatrix, 0);

mTriangle.draw(mMVPMatrix);
```

Transzformáció átadása a shader-eknek

```
private final String vertexShaderCode = "attribute vec4 vPosition;" +
                "uniform mat4 uMVPMatrix;" +
                "void main() {" +
                "  gl_Position = uMVPMatrix * vPosition;" +
                "}";

private final String fragmentShaderCode = "precision mediump float;" +
        "uniform vec4 vColor;" +
        "void main() {" +
        "  gl_FragColor = vColor;" +
        "}";
```

Transzformáció átadása a shader-eknek

```
public void draw(float[] mvpMatrix) {
    GLES20.glUseProgram(mProgram);
    mPositionHandle = GLES20.glGetAttribLocation(mProgram, "vPosition");
    GLES20.glEnableVertexAttribArray(mPositionHandle);
    GLES20.glVertexAttribPointer(mPositionHandle,
            COORDS_PER_VERTEX,
            GLES20.GL_FLOAT, false,
            vertexStride, vertexBuffer);
    mColorHandle = GLES20.glGetUniformLocation(mProgram, "vColor");

    mMVPMatrixHandle = GLES20.glGetUniformLocation(mProgram, "uMVPMatrix");
    GLES20.glUniformMatrix4fv(mMVPMatrixHandle, 1, false, mvpMatrix, 0);

    GLES20.glUniform4fv(mColorHandle, 1, color, 0);
    GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, vertexCount);
    GLES20.glDisableVertexAttribArray(mPositionHandle);
}
```

Feladat

- Alakítsuk át a megoldást úgy, hogy egy iránytű modellt kapjunk
  - Egy vörös és egy kék háromszög mutassa az irányokat.
  - Ezt a mutatót forgassuk körbe az X-tengely körül!

- Megoldás
  - `AndroidBookOpenGLSamples / Compass3D`

Handler-ek

- Kommunikáció GUI és OpenGL szál között
  - Példák
    - OpenGL szál adatot írna egy Android vezérlőelembe (pl. FPS szöveg)
    - GUI szál által fogadott interakció továbbítása 3D vezérléshez
  - `Handler` objektumokkal

```
public final Handler mHandler = new Handler();
public void toggleFPSDisplay() {
        showFPS = !showFPS;
        if (!showFPS) {
            mHandler.post(new Runnable() {
                public void run() {
                    mFPSText.setText("FPS off (press 'f' to turn on)");
                }
            });
        }
    }
```

További példák

- Android SDK samples
  - Hello GL2 (C++), Sensor – Graph (C++)
    - Próbáljuk ki a Hello GL2-t!
    - Nézzük át a forráskódot, figyelembe véve a C++ részeket is!
      - Linkeléshez meg kell adni az EGL és a GLESv2 könyvtárakat
- CodeProject
  - FountainGLProject (Eclipse)
    - Szenzor + érintés

Összetett 3D felszínmodellek

- Célszerű külső modellező programmal készíteni
  - Pl. Blender, 3D Studio, Maya, ...
- Nyílt modell formátum
  - Wavefront OBJ
  - Könnyen olvasható saját kódból is

Magasabb szintű modellezés

- Unity 3D
  - Multiplatform 3D játékmotor
    - iOS, Android, Windows Store, Windows Phone, BlackBerry, asztali PC és web platformokra kódgenerálás
    - Telepítéskor ki kell választani a cél platformokat
    - Kb. 14-20 GB helyet foglal (Visual Studio Community-vel)!
  - Fejlesztőeszköz Windows és MacOS rendszeren érhető el
  - Ingyenes verzió + Pro előfizetés ($75 havonta)
  - Jellemzők
    - Fejlett grafikus funkciók
    - 2D és 3D képek/modellek importálása
      - 3DS Max, Maya, Blender, …
    - Shader programozás
    - Fizikai modellezés
      - PhysX motor

Magasabb szintű modellezés

- Unity 3D anyagok
  - AngryBots példaprogram
    - Unity 4.6, vagy az alatti verziókhoz
  - Standard Assets Example
    - Unity 5 felett
    - Telepítéskor ki kell jelölni
  - Learn with Unity
  - UnityStudent: Unity és UnityScript tananyagok

MOBIL KÉPELEMZÉS ÉS GRAFIKA

SZENZOR VEZÉRELT GRAFIKA ANDROID-ON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Szenzorok + grafika

- Okostelefon szenzorok típusok
  - *Lokáció szenzorok:* eszköz helyzete a Földön
    - GPS, hálózat-alapú (Wifi/mobil) lokáció
    - **Ezekkel külön anyagban foglalkozunk majd!**
  - *Pozíció szenzorok:* eszköz fizikai helyzete
    - Magnetométer, orientáció, …
  - *Mozgás szenzorok:* tengelyek menti gyorsulás, forgás
    - Gyorsulásmérő, gravitáció szenzor, giroszkóp, …
  - *Környezeti szenzorok:* környezeti mért paraméterek
    - Barométer, hőmérő, fénymérő, közelség, …

- Hardver/szoftver szenzorok
  - *Hardveres szenzor:* beépített chip
  - *Szoftveres szenzor:* egy vagy több hardver szenzor mérése alapján új, számított érték

Szenzorok + grafika

- Tulajdonságok
  - Opcionálisak!
    - Gyakoriak: gyorsulásmérő, magnetométer, GPS
    - Ritkábbak: barométer, hőmérő, …
    - Használat előtt ellenőrizni kell a meglétüket!
  - Bizonyos típusból többféle szenzor is elérhető lehet
    - Pl. kétféle gravitációs szenzor
- Grafika szenzoros vezérlése
  - Eszköz forgatásával 3D modell forgatása
  - Gyorsulásmérő segítségével játék irányítása
  - Digitális iránytű
  - Térkép (pozíció-alapú)
  - …

VR – AR – MR

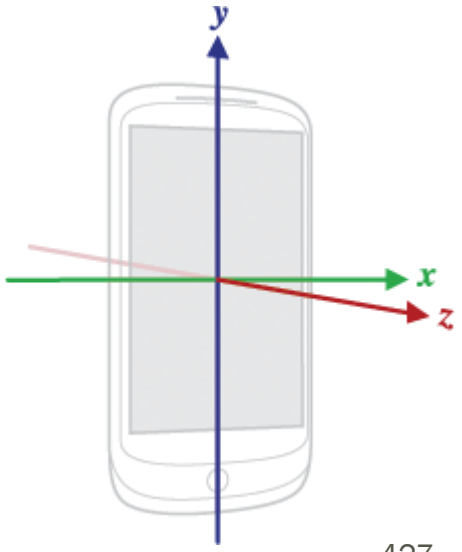
- **Virtuális valóság (Virtual Reality – VR)**
  - Korábbi koncepció, de manapság új lendületet kapott
  - Képernyőn mesterségesen létrehozott 3D látvány
  - Interaktívan manipulálható
  - *William Gibson elájult a virtuális valóságtól*
- **Kiterjesztett valóság (Augmented Reality – AR)**
  - Kamera élőkép + 2D/3D grafika elegyítése
  - Elsősorban információk elhelyezésére
    - *Lokáció-alapú AR:* a készülék helye a Földön, orientációja számít
    - *Jelölő-alapú AR:* a kamera képen felismert speciális objektum szolgáltat információt a 3D modell elhelyezéséhez
    - *Statikus:* rögzített modell elhelyezése a kamera élőképen
    - *Dinamikus:* adatbázisból (helyi vagy hálózati) adatok letöltése és megjelenítése
- **Kevert valóság (Mixed Reality – MR)**
  - Interaktív kiterjesztett valóság
    - Fejlesztés alatt (Microsoft Hololens)

Következő óra témaköre!

Android szenzor kezelés

- Android szenzor API
  - Milyen szenzorok érhetők el a rendszeren?
  - Az egyes szenzorok képességeinek felmérése
    - Mérési tartomány, gyártó, energiaigény, …
  - Szenzor frissítés időközének megadása
  - Eseményfigyelők definiálása

- Android szenzor koordináta-rendszer
  - Készülékhez rögzített
  - Készülék orientáció váltásakor sem változik!
  - A készülék alapértelmezett orientációjának megfelelően értelmeződik
    - Telefon: álló, táblagép: fekvő!



Android szenzor típusok

- *Gyorsulásmérő* (hardveres) – `TYPE_ACCELEROMETER`
  - X, Y, Z tengelyek menti gyorsulás (m/s^2)
  - Gravitációs gyorsulással együtt értelmezendő!
- *Giroszkóp* (hardveres) – `TYPE_GYROSCOPE`
  - X, Y, Z tengelyek körüli szögsebesség (rad/s)
  - Relatív elforgást mér, kiválasztott pozícióhoz képest!
  - Az idő előrehaladtával a mérési hibák összegződnek!
- *Gravitációs szenzor* (szoftveres vagy hardveres) – `TYPE_GRAVITY`
  - X, Y, Z tengelyek menti gravitációs gyorsulás (m/s^2)
- *Lineáris gyorsulás* (szoftveres vagy hardveres) – `TYPE_LINEAR_ACCELERATION`
  - X, Y, Z tengelyek menti gyorsulás (m/s^2)
  - Gravitáció nélkül

Android szenzor típusok

- *Mágneses mező* (hardveres) – `TYPE_MAGNETIC_FIELD`
  - X, Y, Z tengely menti mágnesség érték (μT) – digitális iránytű
- *Orientáció* (szoftveres) – `TYPE_ORIENTATION`
  - X, Y, Z tengelyek körüli elforgás – elavult, használata nem javasolt!
- *Közelség érzékelő* (hardveres) – `TYPE_PROXIMITY`
  - Hívás közben fülhöz emelés – távolság érték (cm), vagy közel/távol

Szenzor fúzió

- Többféle fizikai szenzor értéke alapján
  - Pl. Orientáció mérési pontossága kicsi, giroszkóp pontos (egy ideig)
  - Nyugalmi helyzetben detektáljuk a gravitációs irányt (gyorsulásmérővel) és inicializáljuk ezzel a giroszkópot
    - Nyugalmi helyzet = gyorsulás nagysága 9,81 m/s^2 körüli, vagyis nincs mozgás
  - Mozgás közben forgassuk el a giroszkóp adatai alapján és vonjuk ki a gyorsulásmérő adataiból
- Anyagok
  - *Better motion control using accelerometer/gyroscope sensor fusion*
  - *Android Sensor Fusion Tutorial (CodeProject)*
  - *Sensor Fusion on Android Devices: A Revolution in Motion Processing*
    - *(YouTube videó)*

Szenzor alapú fitnesz alkalmazások

- Mozgástípusok elkülönítése szenzor mérések alapján
  - Gyaloglás, futás, felülés, …
  - Egyedi „mintázatok”
- Különálló eszközök (karkötők, okosórák) vagy telefon?
  - Melyik jobb?
    - Study says phones are just as good as wearables at tracking fitness
  - Viszont
    - A karkötők sokkal kevesebb energiát fogyasztanak (+ Bluetooth 4, 5)
    - Kézen viselve pontosabb a mérés
    - Új biztonsági funkciók valósíthatók meg velük
      - Riasztás, ha a készülék kilép a hatósugárból
      - Nem kell feloldóminta/jelszó, ha a közelben van
      - …
    - Többféle platformmal kompatibilisek lehetnek (Android, iOS)

Érdekességek

- Mozgás felismerés szenzor alapon (*gesture recognition*)
  - Egyszerű: készülék rázása, lefelé fordítása
  - Összetett: tanulást igénylő (pl. séta, futás)
    - *Android Gesture Recognition Tool*
- Külső szenzorok
  - Alcohoot smartphone breathalyzer shipping now for $119
- Beltéri navigáció
  - Meridian launches indoor GPS SDK, Macy's first to launch with it embedded
  - Qualcomm releases proximity beacons that track Android and iOS shoppers
    - Short-range location services are all the rage these days, but they're nothing without the Bluetooth beacons that make them possible. Qualcomm is fulfilling that behind-the-scenes need today by releasing its Gimbal proximity beacons.

Android szenzor API elemei

- `android.hardware.SensorManager`
  - Szenzor szolgáltatások osztálya
    - Szenzorok elérése és figyelése; eseménykezelők hozzárendelése; orientáció információ biztosítása; pontosság, mérési időköz, kalibráció
- `android.hardware.Sensor`
  - Szenzor példány reprezentációja
- `android.hardware.SensorEvent`
  - Mérési eredmény továbbítása az eseménykezelőnek
  - Nyers mérési adatok; szenzor típusa; adat pontossága; időbélyeg
- `android.hardware.SensorEventListener`
  - Interfész mérési eredmények fogadásához

Android szenzorok

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a[1]	n/a[1]
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes[2]	Yes[2]	Yes[2]	Yes
TYPE_PRESSURE	Yes	Yes	n/a[1]	n/a[1]
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes[2]	Yes	Yes	Yes

[1] This sensor type was added in Android 1.5 (API Level 3), but it was not available for use until Android 2.3 (API Level 9).

[2] This sensor is available, but it has been deprecated.

http://developer.android.com/guide/topics/sensors/sensors_overview.html

Android szenzor API

- `SensorManager` példány lekérése

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- Szenzor lista lekérése

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

- Alapértelmezett szenzor lekérése adott típus esetén

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  // Success! There's a magnetometer.
  }
else {
  // Failure! No magnetometer.
  }
```

Android szenzor API

- `Sensor` objektum függvényei
  - `getResolution()`
    - Mérési felbontása
  - `getMaximumRange()`
    - Maximális mérési tartománya
  - `getPower()`
    - Áramigénye
  - `getVendor()` és `getVersion()`
    - Szenor gyártója, verziószáma
  - `getMinDelay()`
    - Milyen gyorsan tudja a mérési adatot szolgáltatni

Szenzor események figyelése

- `SensorEventListener` interfész
  - `onAccuracyChanged()`
    - Szenzor pontosság változik
    - `SENSOR_STATUS_ACCURACY_LOW`, `SENSOR_STATUS_ACCURACY_MEDIUM`, `SENSOR_STATUS_ACCURACY_HIGH`, vagy `SENSOR_STATUS_UNRELIABLE`
  - `onSensorChanged()`
    - Új mérési eredmény
    - Adatok `SensorEvent` objektumpéldányban érkeznek
  - Eseménykezelés vezérlése
    - `onResume()` függvényben hozzárendelés
      - `SensorManager.registerListener()`
    - `onPause()` függvényben lekapcsolódás
      - `SensorManager.unregisterListener()`

```java
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

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

    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
  }
  @Override
  public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Do something here if sensor accuracy changes.
  }
  @Override
  public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    float lux = event.values[0];
    // Do something with this sensor value.
  }
  @Override
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
  }
  @Override
  protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
  }
}
```

Képernyő orientáció figyelése

```
private float mSensorX;
private float mSensorY;
private float mSensorZ;
private long mSensorTimeStamp;

@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() != Sensor.TYPE_ACCELEROMETER)
        return;

    switch (mDisplay.getRotation()) {
    case Surface.ROTATION_0:
        mSensorX = event.values[0];
        mSensorY = event.values[1];
        break;
    case Surface.ROTATION_90:
        mSensorX = -event.values[1];
        mSensorY = event.values[0];
        break;
    case Surface.ROTATION_180:
        mSensorX = -event.values[0];
        mSensorY = -event.values[1];
        break;
    case Surface.ROTATION_270:
        mSensorX = event.values[1];
        mSensorY = -event.values[0];
        break;
    }
    mSensorZ = event.values[2];
    mSensorTimeStamp = event.timestamp;
}
```

Szenzor események figyelése

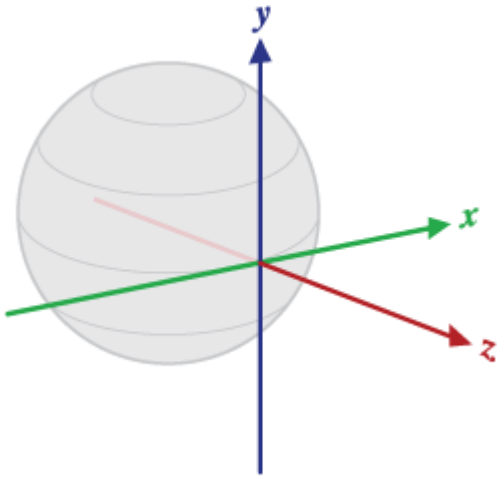
- Mérési gyakoriság
  - Időközök
    - `SENSOR_DELAY_NORMAL` (200.000 mikroszekundum várakozás)
    - `SENSOR_DELAY_GAME` (20.000 mikroszekundum várakozás)
    - `SENSOR_DELAY_UI` (60.000 mikroszekundum várakozás)
    - `SENSOR_DELAY_FASTEST` (nincs várakozás)
  - Nem érdemes túl sok mérési eredményt kérni
    - Gyorsabban merül az akkumulátor
    - Nagyon lefogja a készüléket a számítás

- `Manifest` szűrés (példa)

```
<uses-feature android:name="android.hardware.sensor.accelerometer"
              android:required="true" />
```

Grafika vezérlése

- Világ koordináta-rendszer
  - Különbözik a készülékétől!
  - **Y**: északi irány; **Z**: égbolt felé; **X**: **Y.Z** keresztszorzat



- `SensorManager.getRotationMatrix (float[] R, float[] I, float[] gravity, float[] geomagnetic)`
  - `R` forgató mátrix (kimenő)
  - `I` inklinációs mátrix (kimenő)
  - `gravity`, `geomagnetic` (bemenő): gravitációs és mágneses szenzor mért értékek
  - Definíció szerint
    - `[0 0 g] = gravity * R` (**g** = gravitációs magnitúdó)
    - `[0 m 0] = geomagnetic * R * I` (**m** = mágneses mező magnitúdója)

Gravitáció ellensúlyozás Z-irányba mutasson!

Inklináció: északi irányba forgatás!

Grafika vezérlése

- `getRotationMatrix()` folytatás
  - `R` és `I` mátrixok: 3x3 vagy 4x4 mátrix (átadott mátrix alapján)
  - 4x4 esetén:

$$\begin{pmatrix} M[0] & M[4] & M[8] & M[12] \\ M[1] & M[5] & M[9] & M[13] \\ M[2] & M[6] & M[10] & M[14] \\ M[3] & M[7] & M[11] & M[15] \end{pmatrix}$$

  - Közvetlenül átadható az OpenGL ES `glLoadMatrixf()` függvényének (ES 1.1), vagy shader kódnak
    - Esetleg transzponálni kell a forgató részt (=inverze)
  - 3x3 esetén:

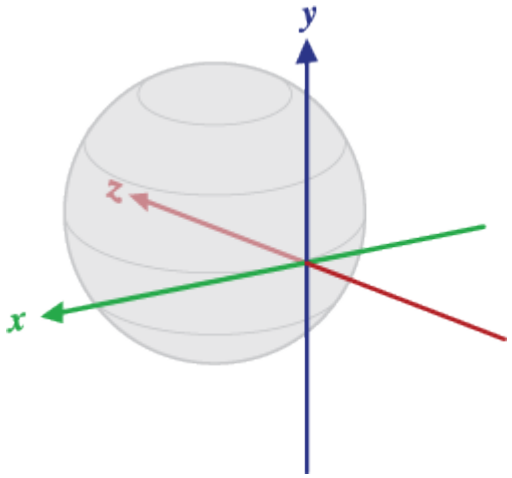
$$\begin{pmatrix} M[0] & M[3] & M[6] \\ M[1] & M[4] & M[7] \\ M[2] & M[5] & M[8] \end{pmatrix}$$

Grafika vezérlése

- `SensorManager.remapCoordinateSystem(float[] inR, int X, int Y, float[] outR)`
  - Koordináta-rendszer váltása, új **X** és **Y** irányt adhatunk meg
  - AR alkalmazások esetén gyakori
    - `remapCoordinateSystem(inR, AXIS_X, AXIS_Z, outR);`
  - Iránytű alkalmazás 90 fokos készülék elforgatás esetén
    - `remapCoordinateSystem(inR, AXIS_Y, AXIS_MINUS_X, outR);`
    - `android.view.Display.getRotation()` eredménye alapján
  - Tovább leírás a készülék orientációjának figyelembe vételéről
    - http://android-developers.blogspot.hu/2010/09/one-screen-turn-deserves-another.html

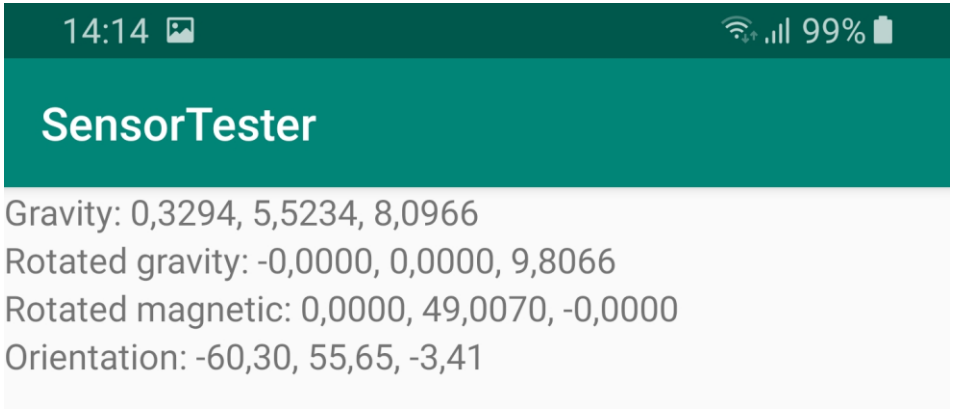
Grafika vezérlése

- Orientáció lekérése a `SensorManager`-től
  - `R` mátrix alapján számol
  - `float[] SensorManager.getOrientation(float[] R, float[] values)`
  - Világ koordináta-rendszer elforgatva **Y** körül!
    - **Y**: északi irány; **Z**: Föld középpontja felé; **X**: **Y**.**Z** keresztszorzat
  - Visszaadott értékek (radiánban!)
    - `values[0]`: azimuth (**Z** körüli forgatás)
    - `values[1]`: pitch (**X** körüli forgatás)
    - `values[2]`: roll (**Y** körüli forgatás)



09_SensorTester példaprogram

- A példaprogram a szenzoradatok alapján számolt forgatómátrix adatokat jeleníti meg
- Láthatjuk, hogy a gravitációs értékekre alkalmazott rotációs mátrix olyan koordináta-rendszerbe forgatja a vektort, ahol a **Z**-tengely a gravitáció irányába mutat, a harmadik érték a gravitációs gyorsulás magnitúdója
- A mágneses adatoknál az északi irányba forgatott esetben csak az **Y** irányban kapunk 0-tól különböző értéket, ahogyan várjuk is



Példa (műhorizont)

```
public class AHActivity extends Activity {
            float[] aValues = new float[3];
            float[] mValues = new float[3];
            HorizonView horizonView;
            SensorManager sensorManager;

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

              horizonView = (HorizonView)this.findViewById(R.id.horizonView);
              sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
              updateOrientation(new float[] {0, 0, 0});
            }

    private void updateOrientation(float[] values) {
      if (horizonView!= null) {
                horizonView.setBearing(values[0]);
                horizonView.setPitch(values[1]);
                horizonView.setRoll(values[2]);
                horizonView.invalidate();
              }
            }
```

Pro Android Augmented Reality könyv (javított) példájának szenzorkezelő része.

Példa (műhorizont)

```
private float[] calculateOrientation() {
     float[] values = new float[3];
     float[] R = new float[9];
     float[] outR = new float[9];

     SensorManager.getRotationMatrix(R, null, aValues, mValues);
     SensorManager.remapCoordinateSystem(R,
                                         SensorManager.AXIS_X,
                                         SensorManager.AXIS_Z,
                                         outR);
     SensorManager.getOrientation(outR, values);

     values[0] = (float) Math.toDegrees(values[0]);
     values[1] = (float) Math.toDegrees(values[1]);
     values[2] = (float) Math.toDegrees(values[2]);

     return values;
   }

   private final SensorEventListener sensorEventListener = new SensorEventListener() {
     public void onSensorChanged(SensorEvent event) {
       if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
         aValues = event.values;
       if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
         mValues = event.values;

       updateOrientation(calculateOrientation());
     }
```

Példa (műhorizont)

```
 public void onAccuracyChanged(Sensor sensor, int accuracy) {}
            };

@Override
protected void onResume() {
     super.onResume();

     Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
     Sensor magField = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);

     sensorManager.registerListener(sensorEventListener,
                                    accelerometer,
                                    SensorManager.SENSOR_DELAY_FASTEST);
     sensorManager.registerListener(sensorEventListener,
                                    magField,
                                    SensorManager.SENSOR_DELAY_FASTEST);
   }

   @Override
   protected void onStop() {
     sensorManager.unregisterListener(sensorEventListener);
     super.onStop();
   }
}
```

Szenzor kezelés

- Használati javaslatok
  - `onSensorChanged()` függvényben csak minimális kód!
    - Ne blokkolja a végrehajtást
  - Elavult szenzortípusok használatának kerülése!
    - Pl. `TYPE_ORIENTATION`
  - Használat előtt ellenőrizzük a rendelkezésre állást!
  - Zajszűrés fontos lehet a mérési pontatlanságok miatt
    - Alul- vagy felüláteresztő szűrő
    - Utolsó mérések átlagolása
    - Kálmán-szűrő
    - ...

Szenzor felhasználási lehetőségek

- Képernyő orientáció állítása
  - Orientációs szenzor
- Játékok irányítása
  - Accelerometer Play (régi API samples)
  - Rengeteg további a Play áruházban
- Telefon némítás
  - Pl. kijelzővel lefelé fordítás érzékelése
- Digitális iránytű; mesterséges horizont; vízimérték
  - Orientációs és mágneses szenzorok
- Fémdetektor
  - Kezdetleges; a mágneses mező zavarainak detektálása alapján
- Csillagtérkép
  - Lokáció + orientáció + modell
- Távolság és magasság mérése
  - Orientációs szenzor + elemi geometria

Feladatok

- Keressünk szenzor adatokat listázó alkalmazásokat a Play Store-ban!
  - Keressünk a `Sensor List` kulcsszavakra
- Modellezünk északi irányba mutató, vízszintes helyzetű nyilat!
  - Az eszköz forgatásának kompenzálásával
- Marine Compass tesztelése
  - https://play.google.com/store/apps/details?id=net.pierrox.mcompass
  - http://www.pierrox.net/cmsms/open-source/marine-compass-2.html
- Próbáljuk ki és módosítsuk a Pro Android AR 4. fejezetének példaprogramját!
  - `09_Pro_Android_AR_4_Non_AR_Demo_Rot_mod.tar.gz`
- FountainGL példaprogram
  - Menü: `Use Tilt Angle` bekapcsolása
- Készítsünk szenzor-támogatott kameraszoftvert!
  - Forgassuk az előnézeti képet és az elkészülő fotót vízszintes helyzetbe!
  - Jelezzük a kijelzőn az egyenes helyzetbe hozáshoz szükséges forgatások irányát!

3D iránytű megoldása

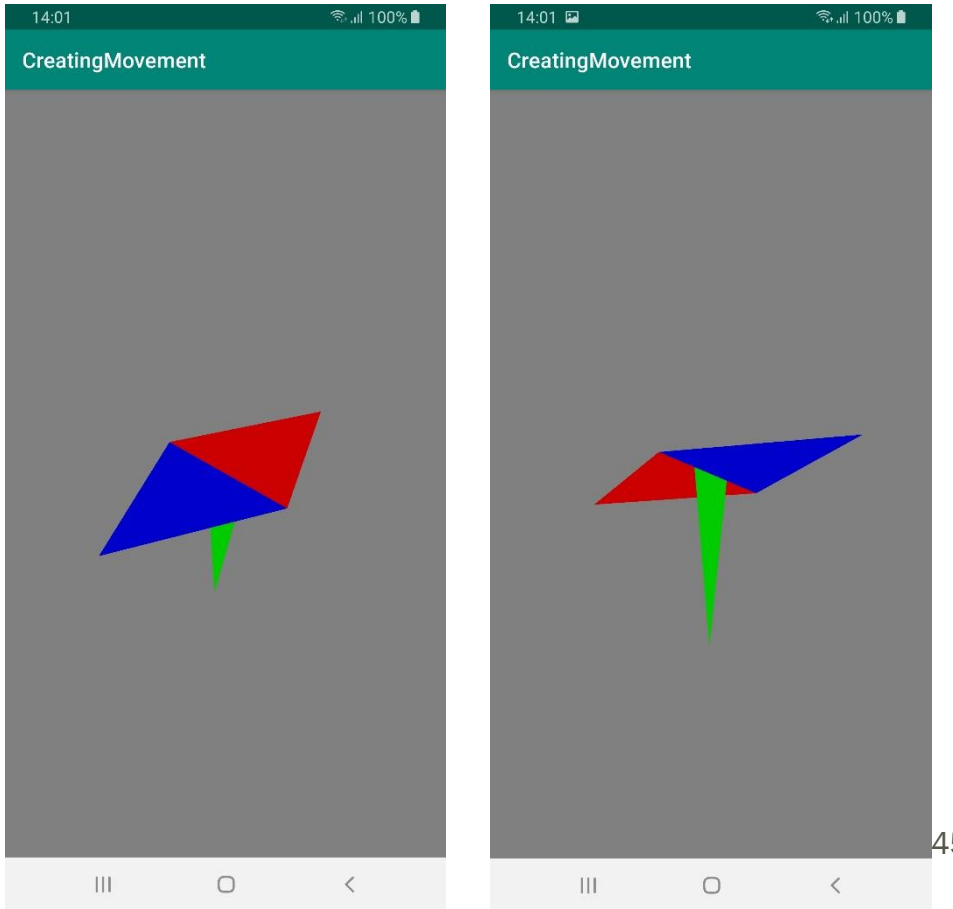
- Induljunk ki a `08_AndroidBookOpenGLSamples / Compass3D` példaprogramból!
  - Ez három darab háromszöget modellez. A vörös az Y-tengely pozitív, a kék a negatív irányába mutat, a zöld a negatív Z-tengely irányába.
- Emeljük át a `09_SensorTester` programból a szenzorfigyelő részeket!
  - 9 elemű tömb helyett 16 eleműeket használjunk a mátrixoknál! Így közvetlenül használhatjuk a grafika 4x4 méretű mátrixaiként.
    - `float[] Rot = new float[16];`
    - `float[] Incl = new float[16];`
    - `float[] outR = new float[16];`
- Az iránytű térbeli mozgatását a szenzor adatokból számolt `outR` mátrix fogja vezérelni.
  - `Matrix.multiplyMM(tempMatrix, 0, mMVPMatrix, 0, outR, 0);`

3D iránytű megoldása

- `outR` meghatározása
  - Gravitációs és mágneses mérési értékekből forgató és inklinációs mátrixok számítása
    - `SensorManager.getRotationMatrix(Rot, Incl, gValues, mValues);`
  - A forgató mátrix bázisán módosítani kell. A 3D grafikus modellt úgy építettük fel, hogy a kamera a `(0, 0, -3)` pontból az origó felé néz. A kamera felé mutat a zöld háromszög, ami a gravitáció irányát szemlélteti. Ha készülék vízszintesen fekszik, akkor a képernyőt nézve a negatív Z-tengely irányába nézünk. Ehhez kell igazítani a modellt, ami az Y-tengely körüli 180 fokos forgatást jelent. Ezt megvalósíthatjuk úgy, hogy az új X-tengely a régi negatív X-tengely irányába nézzen, az Y-tengely pedig ne változzon.
    - `SensorManager.remapCoordinateSystem(Rot, SensorManager.AXIS_MINUS_X, SensorManager.AXIS_Y, outR);`

3D iránytű megoldása

- A teljes megoldást a `09_02_SensorCompass3D.zip` program mutatja be.



MOBIL KÉPELEMZÉS ÉS GRAFIKA

LOKALIZÁCIÓ ANDROID-ON

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

Földrajzi pozíció meghatározása

- **GNSS (GPS, GLONASS, Galileo, Beidou)**
  - Műhold-alapú lokalizáció (*Global Navigation Satellite System*)
    - Műholdaktól mért távolság alapján háromszögeléssel
  - Akár pár méteres pontosság, de függ a terepviszonyoktól (égbolt láthatósága) és az időjárástól is
    - Épületen belül jellemzően nem működik!
  - Az első pozíció meghatározása percekbe is telhet
    - Le kell töltenie az almanachot (műhold pályaadatok) a műholdakról
    - Utána frissítés akár másodpercenként
  - Sebességet is mér Doppler effektussal (elég pontos)
  - Magasság viszont pontatlan
    - WGS84 geoid feletti magasságot ad, geoinduláció nélkül (?)
  - Energiaigényes a használata
  - Nincs szükség internet kapcsolatra, „bárhol” működik

Földrajzi pozíció meghatározása

- GPS
  - NEM NAVIGÁCIÓ!
  - NINCS TÉRKÉP!
  - Csak hosszúsági, szélességi és geoid feletti magasság adatokat ad!
  - Sajnos a köznyelvben nagyon pontatlan a jelentése!

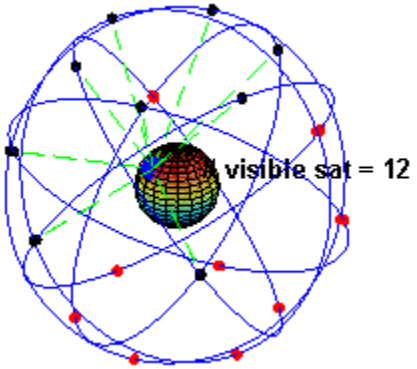
- Megbízhatóság fokozása
  - WAAS, EGNOS rendszer
    - Geostacionárius műholdak, földi kiszolgáló állomások
    - Hibás műhold adatok rövid időn belül megjelennek, így kiküszöbölhetők

Földrajzi pozíció meghatározása

- GPS (rövid) története
  - 1973: Kiépítés kezdete (USA Védelmi Minisztériuma)
    - Katonai célok; indoklás nélkül korlátozhatja az USA
    - „Selective Availability”
      - Katonai célok: méter alatti pontosság
      - Polgári célok: 100 méter nagyságrendű pontatlanság
  - 1995: teljes funkcionalitás 24 műholddal
  - 2000. május 2.: „Selective Availability” megszüntetése
    - Polgári célokra is méteres pontosság elérhető
- Galileo
  - EU vezetésével (2003)
  - GPS függőség csökkentésére
  - Tervezési fázisban (és csúszásban) van
    - Tervezett teljes kiépítés: 2019

Földrajzi pozíció meghatározása

- **Működési elve**
  - *Föld körül keringő műholdak (űr szegmens)*
    - 24-32 darab elérhető; 20e km távolság
    - Atomórával ellátva
    - Nagy százalékban a Föld felszínén legalább 4 műhold látható
  - *Földi vezérlés (vezérlő szegmens)*
    - Hibák detektálása; műhold atomórák szinkronizálása; pályaadatok korrekciója (USA légerő irányítja)
  - *Földi vevő (felhasználói szegmens)*
    - Legalább 4 műholddal kapcsolat; eltelt idő alapján háromszögel



- **Adatok**
  - *Ephemeris:* a műhold státusza és precíz pályaadata
  - *Almanac:* becsült pályaadat és státusz további max. 32 műholdra

Földrajzi pozíció meghatározása

- Mobil és/vagy Wifi hotspot információ alapján
  - Pozíciójuk ismert, jelerősségük mérhető
    - Netes adatbázisok
    - Adatkapcsolat szükséges!
  - Pozíció számítása háromszögeléssel
  - Gyorsabb pozícionálás
  - Nagyobb pontatlanság
    - Akár kilométeres nagyságrendű is, de durva pozícionálásra alkalmas

- AGPS (Assisted GPS)
  - Internet kapcsolat esetén a műhold pályaadatok onnan letölthetők, nem kell a műholdas lassú kapcsolatra várni

Földrajzi pozíció meghatározása

- Geokódolás és inverze
  - GPS pozíció alapján cím meghatározása és cím alapján GPS pozíció lekérése
  - Adatkapcsolat szükséges!
  - Google is ad ilyen szolgáltatást

- Google Maps API
  - Információk megjelenítése Google Térkép felett
  - Saját adatok térképre vetíthetők (overlay)
  - Az újabb verziók már Google Play Services hozzáférést követelnek
  - Google regisztráció kell
  - Ingyenes, de nagymértékű használat esetén fizetős
    - Emiatt pl. a FourSquare áttért az OpenStreetMap használatára

Android lokációs API-k

- Android lokációs API

  Több készüléken érhető el (Nyílt)

  - `android.location` csomag
  - `LocationManager` rendszerszolgáltatás
    - Utolsó ismert lokáció lekérése (`LocationProvider` példányoktól)
    - Eseménykezelő beállítása folyamatos pozíció követésre
    - Intent értesítés kérhető megadott koordináta közelébe érve
      - *Proximity alert*

- Google Play Services API

  Bővebb szolgáltatáskészlet (Zárt)

  - Csak Google szolgáltatásokat kínáló készülékeken érhető el!
  - Aktuális lokáció; lokációfrissítés; geokódolás; geofencing
  - Mozgástípus felismerése
    - Gyaloglás, futás, ...
  - Tárolt adatok használata teszteléshez, hibakereséshez

Android lokációs API

- Rendszerszolgáltatás elkérése

```
LocationManager locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
```

- Pozíciókövetés beállítása

```
LocationManager.requestLocationUpdates(String provider, long
minTime, float minDistance, LocationListener listener)
```

  - `provider`: Adatszolgáltató kiválasztása
    - `NETWORK_PROVIDER`: mobil és/vagy Wifi háromszögelés
    - `GPS_PROVIDER`: műhold-alapú
    - Akár mindkettő kérhető; két `requestLocationUpdates()` függvényhívással
  - `minTime`: minimum időtartam két frissítés között (`0`: leggyakoribb)
  - `minDistance`: minimum távolág két frissítés között (`0`: leggyakoribb)
  - `listener`: lokáció frissülését kezelő objektum

Android lokációs API

- Pozíciókövetés beállítása
  - Választás kritériumok beállítása alapján; eredmény átadása `requestLocationUpdates()` függvénynek

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setSpeedRequired(false);
String bestProvider =
        locationManager.getBestProvider(criteria, true);
```

Android lokációs API

- `LocationListener` interfész
  - `onLocationChanged(Location location);`
    - Új pozíció feldolgozása
    - Adatok a `location` objektumpéldányban
  - `onStatusChanged(String provider, int status, Bundle extras)`
    - Lokációszolgáltató állapotváltozása (pl. GPS vétel megszűnt, vagy helyreállt)
  - `onProviderEnabled(String provider);`
    - `provider` helymeghatározó mód engedélyezésre került
  - `onProviderDisabled(Strind provider);`
    - `provider` helymeghatározó mód letiltásra került

- Pozíciókövetés leállítása
  - `locationManager.removeUpdates(locationListener);`

Android lokációs API

- Pozíciókövetés beállítása (alternatív technika)
  - NMEA 0183 formátumú adatfolyam fogadása
    - Szabványos ASCII formátum, soros porti kommunikációra (külső vevő esetén), adatok „mondatokban” érkeznek, soronként
    - `addNmeaListener(GpsStatus.NmeaListener listener)`
    - `onNmeaReceived(long timestamp, String nmea)` függvénye

```
$GPGGA,092750.000,5321.6802,N,00630.3372,W,1,8,1.03,61.7,M,55.2,M,,*76
$GPGSA,A,3,10,07,05,02,29,04,08,13,,,,,1.72,1.03,1.38*0A
$GPGSV,3,1,11,10,63,137,17,07,61,098,15,05,59,290,20,08,54,157,30*70
$GPGSV,3,2,11,02,39,223,19,13,28,070,17,26,23,252,,04,14,186,14*79
$GPGSV,3,3,11,29,09,301,24,16,09,020,,36,,,*76
$GPRMC,092750.000,A,5321.6802,N,00630.3372,W,0.02,31.66,280511,,,A*43
```

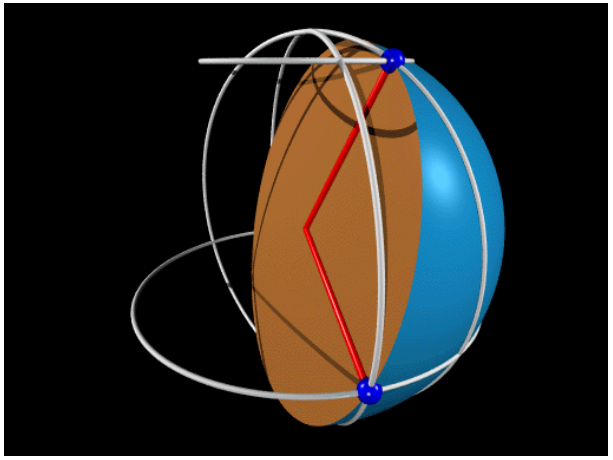
    - Leírás az NMEA mondatokról
      - http://www.kh-gps.de/nmea.faq

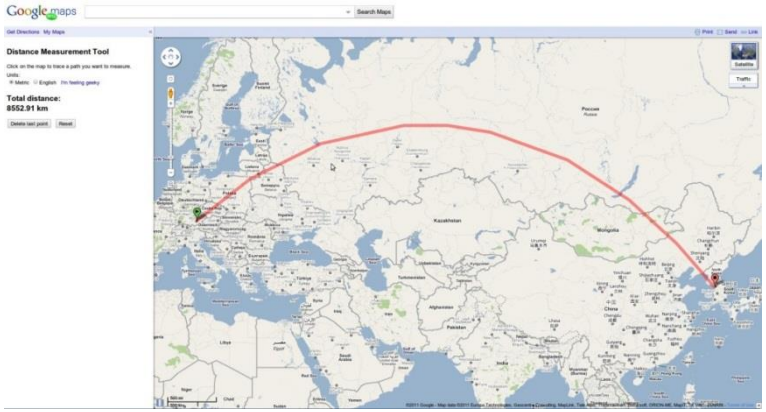
Android lokációs API

- `Location` osztály
  - Lokációfrissítésre reagáló objektumunk kapja
  - Szélesség, hosszúság, magasság, becsült pontosság (méterben), sebesség, haladási irány, idő, szolgáltató, … lekérése tagfüggvényekkel
    - `getLatitude(); getLongitude(); getAltitude(); getAccuracy(); getSpeed(); getBearing(); getTime(); getProvider();`
    - Beállítás a megfelelő `set…()` függvényekkel
  - Haladási irány megadott célhoz
    - `bearingTo(Location dest);`
  - Távolság a megadott célhoz (méterben)
    - `distanceTo(Location dest);`
    - „Great Circle” távolság (ívhossz gömb felszínén!)
  - Konverziók sztring és lebegőpontos ábrázolás között
    - `FORMAT_DEGREES; FORMAT_MINUTES; FORMAT_SECONDS`

Android lokációs API

- Távolság számítása (Great Circle Distance)
  - $\Delta\sigma = \arccos(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos\Delta\lambda)$
  - $d = r\Delta\sigma$
  - Ahol
    - (ϕ_1, λ_1) és (ϕ_2, λ_2) a két koordináta
    - r a Föld átlagsugara (6371 km)
    - A távolságot méterben kapjuk





- Calculate distance, bearing and more between Latitude/Longitude points

Android lokációs API

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
      // Called when a new location is found by the network location provider.
      makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider, int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
  };

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);

// …

locationManager.removeUpdates(locationListener);
```

Android lokációs API

- `AndroidManifest` engedély
  - Hálózati lokáció használata (pontatlanabb)

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

  - GPS lokáció használata (pontosabb)

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

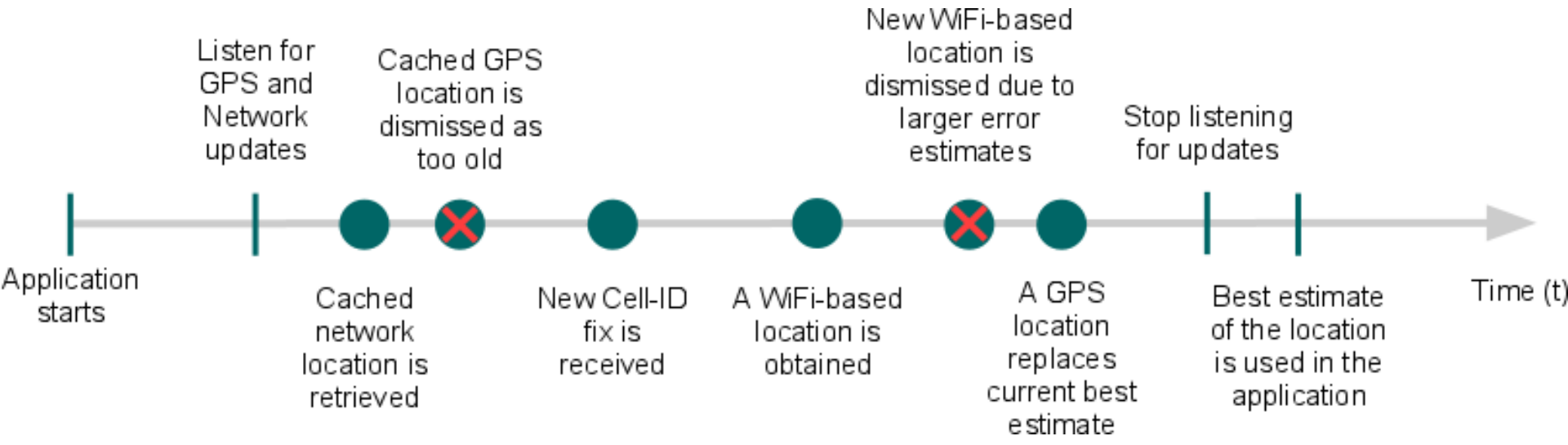
- 23-as API szint felett engedély ellenőrzés a kódban!

Android lokációs API

- Stratégia
  - Kapjunk mielőbb lokációs adatot, még ha az nem is nagyon pontos

```
Location lastKnownLocation =
locationManager.getLastKnownLocation(locationProvider);
```

  - Ahogy egyre pontosabb forrásból jön adat, frissítsünk



Geokódolás és inverze

- Átalakítás földrajzi koordináta és postacím között
  - Google szolgáltatás, internet kapcsolat kell
    - Manifest!
  - `Geocoder` osztály használata

- Címhez koordináta

```
Geocoder gc = new Geocoder(this, Locale.ENGLISH);
String address = ”Árpád tér 2, Szeged”;
List<Address> locations = null;
int maxNumOfResults = 5;
locations = gc.getFromLocationName(address, maxNumOfResults);
```

  - Használható befoglaló téglalappal (foktrapézzal) is
    - +4 paraméter a függvénynek

Geokódolás és inverze

- Koordinátához cím

```
double lat = 47.0;
double lon = 19.0;
Geocoder gc = new Geocoder(this, Locale.getDefault());
List<Addresses> addresses = null;
int maxNumOfResults = 5;
addresses = gc.getFromLocation(lat, lon, maxNumOfResults);
```

Térkép

- Google Maps Android API V1
  - Térkép csempék, pontok, útvonalak rajzolása, ...
  - Támogatása megszűnt; `MapView` osztály már nem használható!
  - A régebben kiadott kulcsok még működnek, de újat nem adnak ki

- Google Maps Android API V2
  - Google Play Services SDK telepítése
    - Felvétele a `Manifest` fájlba
  - Projekt regisztráció a `Google APIs Console` rendszerben
  - Google Maps API kulcs kérése
    - Szintén a `Manifest`-be kell bemásolni
    - Nagy mennyiségű adathozzáférés esetén fizetni kell!
  - Példák
    - Example of Using Google Maps Service and GPS in Android
    - Flirting with Google Maps on Android

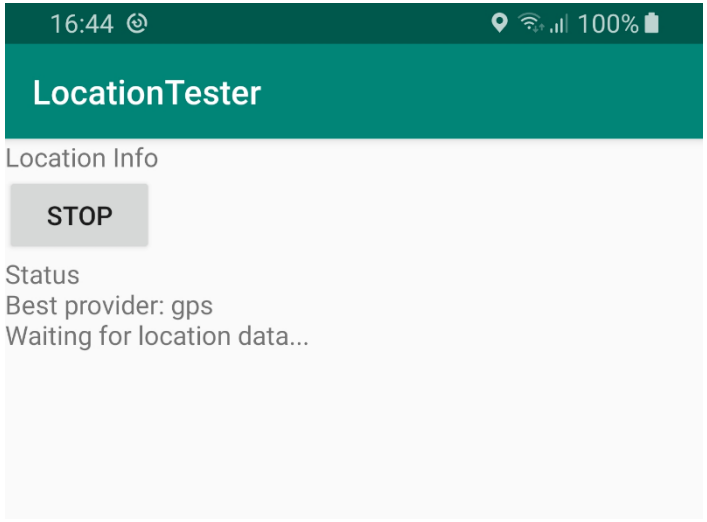
Térkép

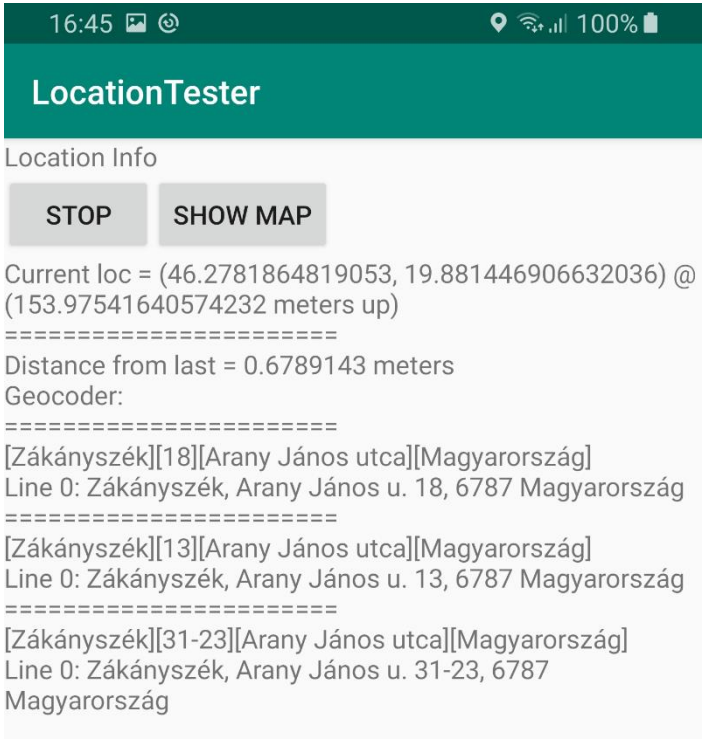
- OpenStreetMaps
  - Közösségi szerveződésű világtérkép, bárki csatlakozhat
  - MapView V1-szerű funkcionalitást is elérhető
  - Hozzáférés a vektoros, nyers adatokhoz is
  - Használata Android programokban: pl. `OSMDroid` könyvtár
    - https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library
  - OpenStreetMaps keretrendszerek
    - http://wiki.openstreetmap.org/wiki/Frameworks

Példaprogram

- Lokalizáció
  - `10_01_LocationTester` példaprogram

LocationTester





Feladatok

- Próbáljuk ki a Google Play Services lokációs példaprogramjait!
  - Making Your App Location-Aware
  - https://github.com/googlesamples/android-play-location
  - https://www.youtube.com/watch?v=S8sugXgUVEI

MOBIL KÉPELEMZÉS ÉS GRAFIKA

VR – AR– MR: VALÓSÁGOK ÉS ALKALMAZÁSAIK

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Előadás témakörök

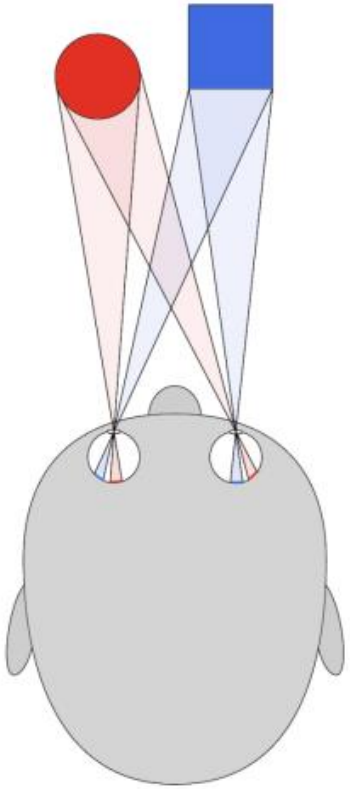
- Fogalmak
  - Mit jelentenek a különféle „valóságok”?
- AR és MR
  - Kategóriák, eszközök, keretrendszerek és alkalmazások
- VR működési elv, technológiai háttér, eszközök
  - Hogyan csapják be az agyunkat?
  - Szenzorok + számítógépes grafika
  - Headset-ek, okostelefonok, szemüvegek
- Tartalom
  - Milyen típusú élmények érhetők jelenleg el?
- Jelen és jövőbeli alkalmazások
  - Milyen fejlesztések várhatók?
  - Áttörés lesz vagy sem?

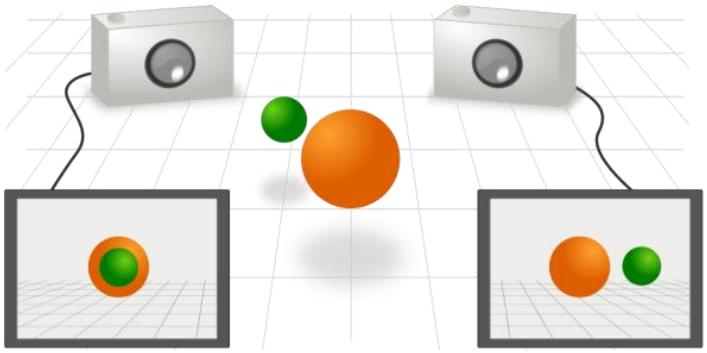
VR – AR – MR

- Virtuális valóság (Virtual Reality – VR)
  - Képernyőn mesterségesen létrehozott 3D látvány
  - Interaktívan manipulálható
  - Korábbi koncepció (1960-as évek), de manapság új lendületet kapott
- Kiterjesztett valóság (Augmented Reality – AR)
  - Kamera élőkép + 2D/3D grafika elegyítése
  - Elsősorban információk elhelyezésére
  - Okostelefonokon, önálló szemüvegek (Google Glass, Intel, …)
- Kevert valóság (Mixed Reality – MR)
  - Interaktív kiterjesztett valóság
    - Valós látvány + szemüvegre vetített információ
    - Microsoft Hololens (fejlesztés alatt), Magic Leap (előzetes verzió)

Térlátás működési elve

- Emberi (binokuláris) térlátás alapja
  - Két „képalkotó berendezés”
    - Szem = Lencserendszer + érzékelő (retina)
    - Kicsit más nézőpontból képzik le a világot
    - Diszparitás: közeli és távolabbi tárgyak leképződése eltérő
  - Fénymérés idegpályákon az agyba jut, ahol az érzet létrejön





Kép: Arne Nordmann

A 3D VR alapelve

- Készítsünk két képet, ahogyan a bal és a jobb szemünk látná a tájat
  - Kissé eltérő nézőpontból a *teljes elmerülés* eléréséhez
- Juttassuk ezeket a képeket a külön-külön a két retinára
- Kövessük a felhasználót és frissítsük a látványt
  - Fej fordulása, térbeli mozgás (séta), akciók (gesztusok)

Virtuális valóság (VR)

- Speciális szemüveg szükséges
  - Bal és jobb szemünk elé eltérő kamerapozícióból generált 3D kép juttatásához
- Nagyon erős számítási háttérkapacitás kell!
  - Jellemzően PC + erős GPU
  - Vezetékes kapcsolat a szemüveggel (2018)
- A fejmozgatásra nagyon gyorsan reagálni kell!
  - A fejmozgatásra nagyon gyorsan reagálni kell!
  - Korai VR próbálkozások legnagyobb hibája a nagy késleltetés volt. Az agy „nem értette” az okát.
- Látvány realisztikus és folyamatos előállítása
  - Kieső képkockák, perem menti torzulások hányingert okozhatnak!
    - Idegméreg hatásához hasonló élmény, amire a szervezet reagál)
  - Fárasztó az agynak/szemnek

Virtuális valóság (VR) eszközök

- PC + vezetékes kapcsolat
  - Oculus Rift, HTC Vive
  - Sony Playstation VR (Project Morpheus)
- (Erős) okostelefon a szemüvegben
  - Samsung Gear VR (Oculus technológia)
  - Google Daydream
- Olcsó alternatíva: Google Cardboard
  - Pár dollárból kihozható
    - Kartonlapból hajtogatható doboz az okostelefon tartására
    - Lencsék a látvány szemre vetítéséhez
    - Mágnes nyomógombként
  - **Az okostelefon számítási teljesítménye, szenzorainak minősége és kijelzőjének felbontása befolyásolja az élményt!**

Hardver háttér

- **Kijelző**
  - Nagy felbontás (2160x1200)
  - Gyors megjelenítés (~90 FPS)
- **Lencserendszer**
  - Állítható szemtávolság
  - Dioptria korrekció
  - Nagy látómező (110 fok)
- **Fejmozgás követése**
  - Giroszkóp + gyorsulásmérő + magnetométer
- **Térbeli mozgás követése**
  - Nehezebb megvalósítani!
  - Pl. Infravörös LED fények a szemüvegen, szenzor rögzített pozícióban a szobában
- **(Kéz)mozgások, gesztusok**
  - Kontrollerek, vezérlők

Oculus Rift, HTC Vive

- Erős hardver
- Jó szenzorok
- Térbeli mozgáskövetés
- Állítható vetítési paraméterek
- Drága (PC + szemüveg)

Okostelefonok

- Olcsóbb megoldás
- Kompromisszumok!
  - Gyenge kijelző
  - Lassú
  - Gyengébb szenzorok

2019. ősze: a mobil gyártók megszüntetik a VR irányvonalat. Áldozatok: Google Cardboard API, DayDream, Samsung Gear VR

Hardver háttér

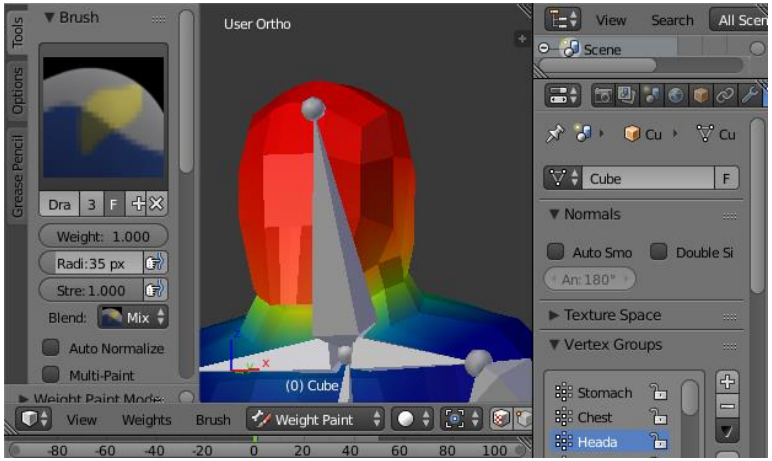
- 2019. ősze: a mobil gyártók megszüntetik a VR irányvonalat
- Áldozatok
  - Google Cardboard API: nyílt forráskódúvá tették
  - Dayream
  - Samsung Gear VR: az új Samsung készülékek már nem támogatják a szemüveget
    - John Carmack on Samsung Gear VR – „I wish we had done more”: https://www.androidcentral.com/john-carmack-samsungs-gear-vr-i-wish-we-had-done-more
- Cégek indokai
  - Az okostelefon más funkciói nem használhatók VR használat közben (de miért is kellene?)
  - Az értesítések, egyebek zavaróak VR használat közben
- Jelenlegi irányvonal
  - Kb. okostelefon képességű hardverrel rendelkező dedikált szemüvegek (pl. Oculus Quest)

VR tartalom

- 3D játékok
  - Évtizedek óta fejlődő ágazat
  - Áttérés viszonylag egyszerű
    - Két virtuális kameraképet kell generálni
    - Valós 3D modellezés kell (a 2D közelítő trükkök helyett)
  - A népszerű grafikus motorok támogatják a Rift-et, a Vive-ot és az okostelefonos VR-t is
    - Unity, Unreal, Cry Engine, …

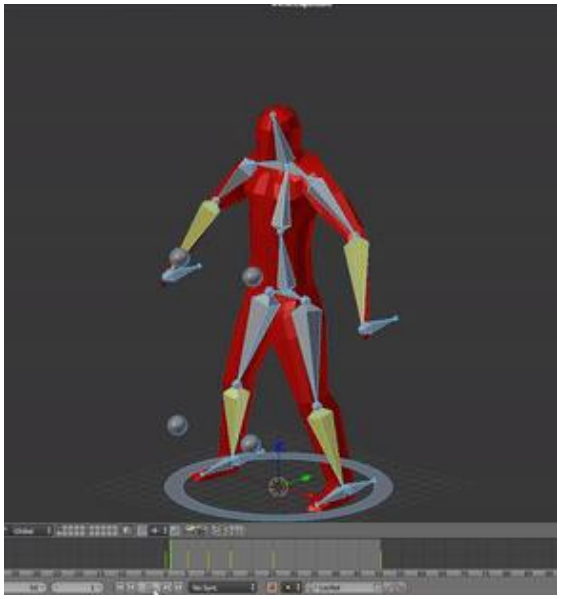
- 3D modellezés
  - CAD modellek
  - Animációk

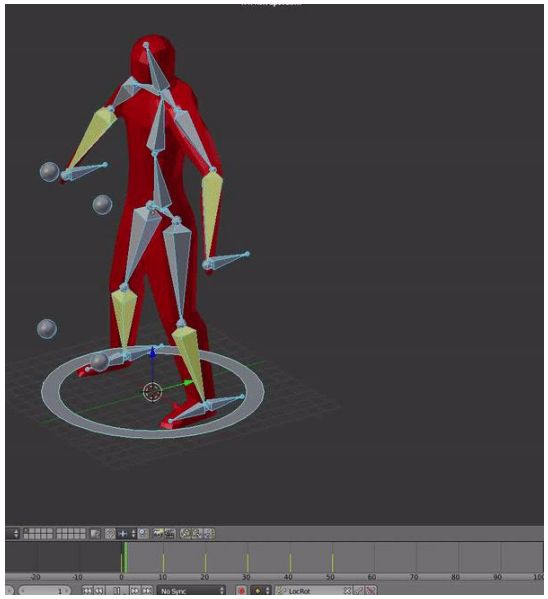
3D modellezés és animáció

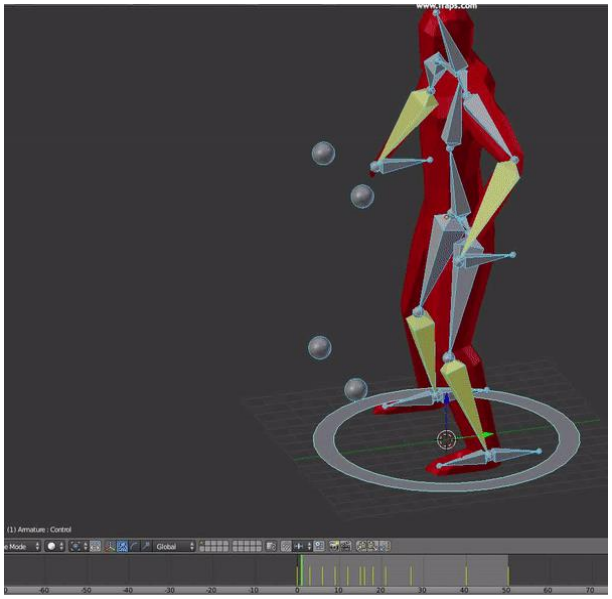


Animációk Blender-ben

Forrás: Galla Csaba (BSc szakdolgozat)







Tartalom

- 360 fokos videók
  - Kamera rendszerek
    - Két halszem optika
    - Több kamera együttese a teljes szögtartományt lefedve
  - Mostanság megjelenő alkalmazási terület!
    - Youtube, Facebook, Google Fotók már jelenleg is támogatja
  - A kamera pozíciójából bármely pillanatban szabadon körültekinthetünk
    - Nem feltétlenül 3D a hatás, a fejmozgás követés a fontosabb

További alkalmazások

- Szórakozás
  - Útikalauzok, mozifilmek, …
- Ipar
  - Valósághű betanítás
- „Távjelenlét”
  - Tárgyalások, videó hívások személyesebbé tétele
- Orvosi alkalmazások
  - Orvosi képzés segítése
    - Annotált 3D atlaszok, műtét gyakorlása, …
  - Távoli műtétvégrehajtás
  - Terápia segítése
- …

VR az élettani terápiákban

- **Célok**
  - Konvencionális terápiák segítése és kiterjesztése 3D alkalmazásokkal
  - „Komoly játékok”

- **Eszközök**
  - Oculus Rift VR
  - Unity játékmotor

- **Együttműködő partnerek**
  - Dr. Csibri Péter (Élettani Intézet, Szegedi Tudományegyetem)
  - SZTE BSc és MSc hallgatók

Átültetett vese adoptáció

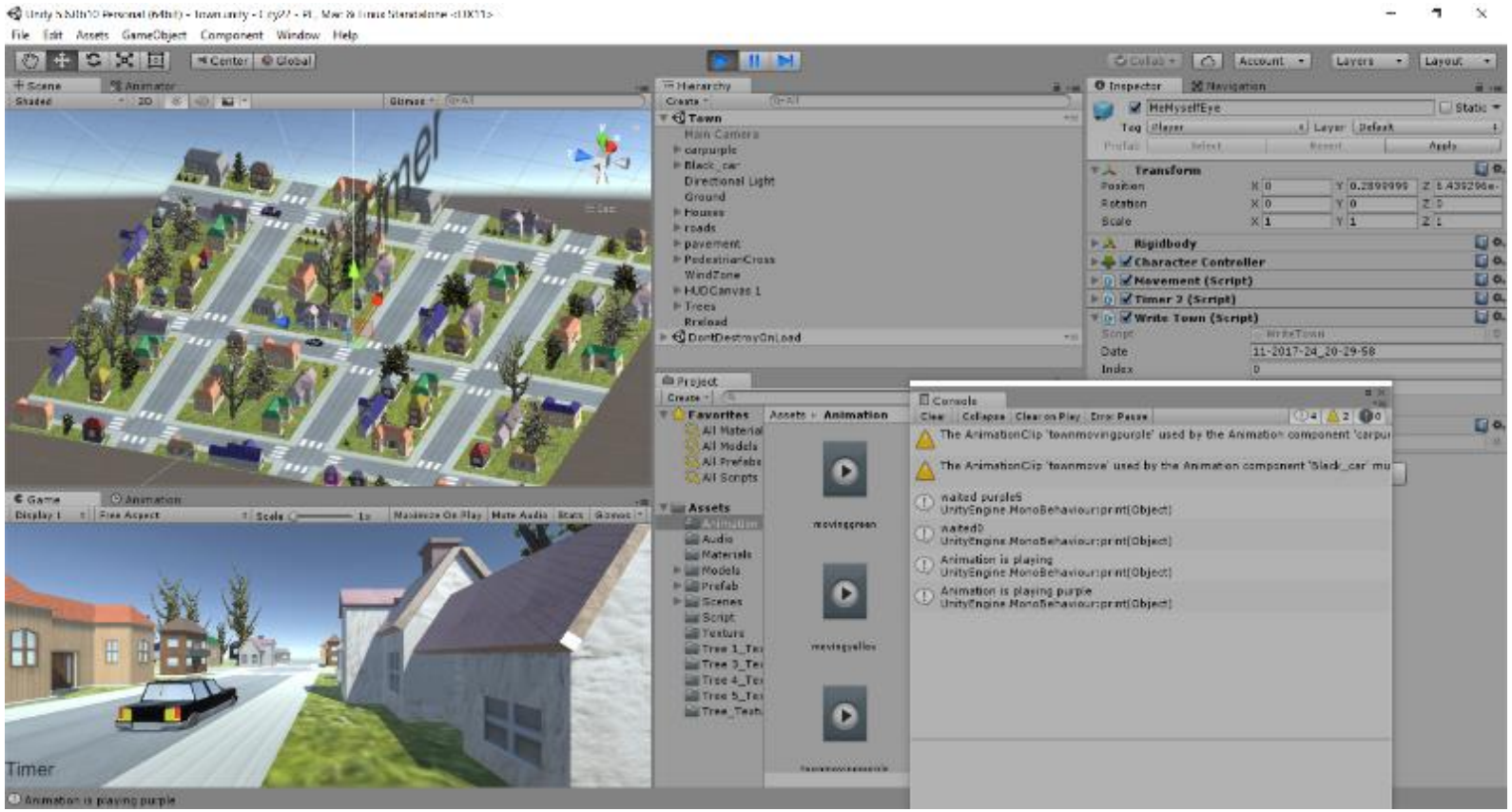
- A stressz növeli a kilökődés veszélyét
- Virtuális testen kívüli élmény használata
  - Gumi kéz illúzió kiterjesztése
  - Szimultán vizuális (VR) és érintés (valóság) élmény
    - A páciens agya sajátjaként érzékeli a virtuálisat
    - A vese megjelenítésével a stressz szint csökkenthető (elfogadás sajátként)

Sztrók terápiák

- Agyi károsodás következtében
  - A sérülés határán lévő terület stimulálása szükséges
  - A páciens (részlegesen) lebénulhatott → konvencionális terápia nem tud elindulni
    - VR: utat nyit hétköznapi szituációk gyakorlásához ágyhoz kötötten is
  - Neglektált látómező terület kialakulhat
    - A páciens nem ismeri el

- Feladatok
  - A látómező szisztematikus feltérképezése
  - VR terápia hatására
    - Tudatosítás
    - Új készségek fejlesztése a mindennapi élethez alkalmazkodáshoz

VR átkelés az úton



VR fejlődési irányok

- Google Daydream (megszűnt!)
  - Google Cardboard frissítése okostelefonokhoz
  - Előírás a minimális hardver követelményre
  - Jobb minőségű keret
  - Mozgásérzékelő távirányító (kontroller)
    - Kéz mozgásának virtuális térbe viteléhez

- Önálló kijelzős sisakok (okostelefon nélkül)
  - Google I/O 2017
    - Lenovo Mirage Solo (2018)
  - Vadiúj VR-sisakot mutatott be a HTC
    - Külső kamerával követhető a térbeli mozgás (6 szabadsági fok)
  - Oculus Go VR (2018), Oculus Quest VR (2019)

VR fejlődési irányok

- Hardver fejlődése
  - A technológia most ért arra a szintre, hogy használható legyen
    - Az ötlet ismert már az 1960-as évek óta
    - Aerosmith: Amazing videóklip (1994): https://www.youtube.com/watch?v=zSmOvYzSeaQ
  - A fejlődés várhatóan továbbra is folyamatos lesz
  - Számítási kapacitás növekedése az eszközök méretének/súlyának csökkentésével
  - Kábelek „elvágása”

- Emberi látás jobb közelítése
  - Szemmozgás követése
    - Fókusz, szemtengelyek iránya, pupillatágulat, …
    - Egyelőre ebből a szempontból statikus a VR világ!
      - Az agy nem érti, miért nem változik a kép a „paraméterek” állításával → gyors kifáradáshoz, rosszulléthez vezethet egyeseknél

Jövő?

- Erős iparági szándék (és pénz)!
  - Pl. Oculus Rift:
    - 2012: Kickstarter kampány, 2,5 millió USD támogatás
    - 2014. március: Facebook felvásárolja 2 milliárd (!) dollárért!
  - Okostelefon gyártók komplett csomagot igyekeztek adni
    - Samsung Gear VR + Gear 360 kamera + erős okostelefonok + Oculus szoftver
    - LG G5 + 360 Cam
- Hová alakul?
  - Nem egyértelmű!
    - Jelenleg még drága, és nehézkesen használható (vezetékek…)
    - A hardver fejlődésével várhatóan olcsóbb és jobb lesz
  - Kérdés: „rákap-e” elegendő számú felhasználó?
    - ~100 milliós nagyságrendben (több technológia is elbukott már korábban a kis mértékű érdeklődés következtében)
  - Speciális területeken várhatóan megmarad
    - Kiképzés, orvoslás, távjelenlét, …

Ready Player One

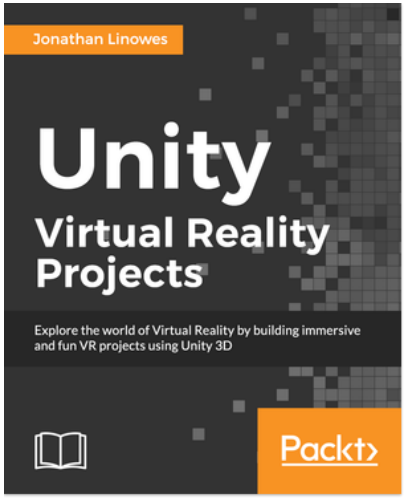
Ernest Cline könyvén alapuló mozifilm előzetese (2018)

(Mobil) VR alkalmazások fejlesztése

- Google Cardboard, DayDream
  - SDK elérhető: https://developers.google.com/vr/
  - Android, iOS, Unity és Unreal támogatás
  - Lencsetorzítás korrekció, fejmozgás követése, 3D kalibrálás, binokuláris render, térbeli hang, felhasználói interakciók kezelése
  - Példa alkalmazások natív Cardboard és Unity környezetben
    - Natív Android osztályok, interfészek
      - `CardboardActivity, onNewFrame(HeadTransform), onDrawEye(Eye)`
      - `CardboardView (GLSurfaceview kiterjesztése)`
      - `CardboardActivity.onCardboardTrigger()`
- Példaprogram
  - `11_01_CardboardVision.zip`: OpenGLES grafika, kamera élőkép megjelenítéssel

(Mobil) VR alkalmazások fejlesztése

- Unity
  - Általános és egységes 3D modellezés
  - Sokféle VR eszköz támogatása
    - Cardboard → Oculus Rift



- Lullaby (C++)
  - https://github.com/google/lullaby

Kiterjesztett valóság (AR)

- Fő fajtái
  - *Lokáció-alapú AR („abszolút”)*
    - A készülék helye a Földön (GPS) + orientációja számít.
    - Adatok helyi vagy hálózati adatbázisokból.
  - *Jelölő-alapú AR („relatív”)*
    - A kamera képen felismert speciális objektum (*jelölő, marker*) szolgáltat információt a 3D modell elhelyezéséhez.
  - *Képelemzés-alapú AR*
    - Kamera élőképen detektált objektumok alapján generált eredmény elhelyezése a képen valós időben.

- Lehet
  - *Statikus*
    - Készüléken elérhető, rögzített modell elhelyezése a kamera élőképen
  - *Dinamikus*
    - Adatbázisból (helyi vagy hálózati) adatok letöltése és megjelenítése

Pokémon Go

Lokáció-alapú (térkép, lények helye központi adatbázisból, kamera képre vetítve)

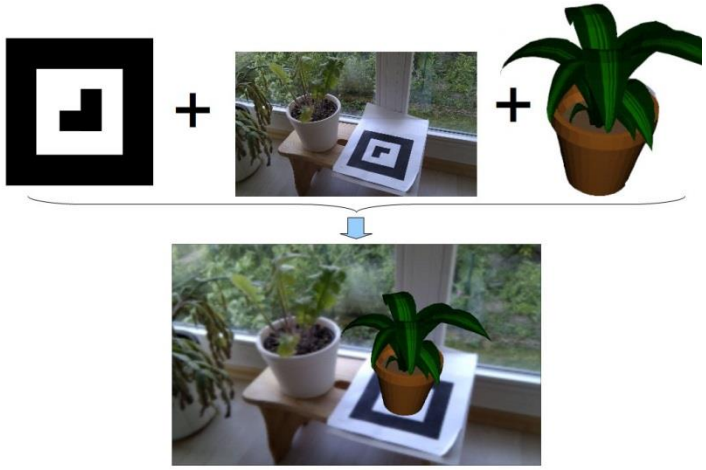
Lokáció-alapú AR – mobil alkalmazások

- Layar
  - Kamera képre információ elhelyezése
  - Első „igazi” AR; publikus API
  - Saját adatbázisokkal bővíthetjük

- Wikitude browser
  - Layar-szerű működés, publikus adatforrások

Jelölő-alapú AR

- Kamera élőképen gyorsan detektálható speciális sík minták
  - Jelölők, markerek



- Térbeli 3D koordináta-rendszer
  - Origó + térbeli irányok
  - A minta perspektív torzulásából számítható a kamera helyzete a mintához képest

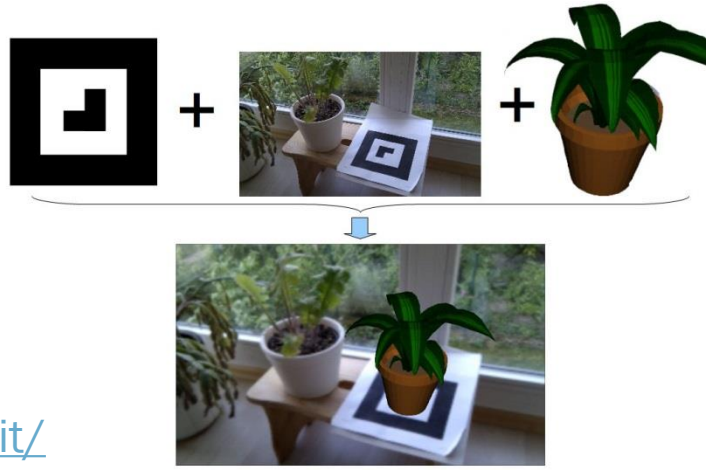
- Mesterséges 3D objektum elhelyezhető a kamera terébe transzformálva
  - Takarások kezelése nehéz!

Jelölő-alapú AR

IKEA AR katalógus (2014)

Jelölő-alapú AR

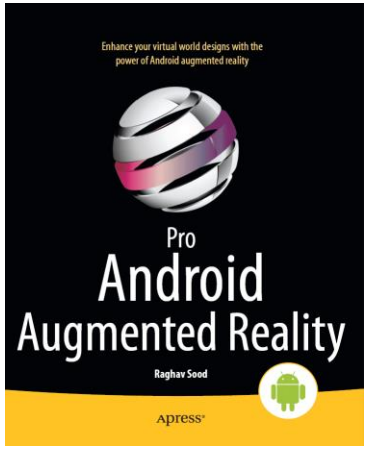
- Részletes leírás
  - Külön diasorozat!



- Programcsomagok
  - ARToolKit (1999-)
    - http://www.hitl.washington.edu/artoolkit/
    - Nyílt forráskód (SourceForge); multiplatform
  - AndAR alkalmazás (Google Play)
    - https://code.google.com/p/andar/
    - How to use your own markers?
    - svn checkout http://andar.googlecode.com/svn/trunk/ andar-read-only
  - NyARToolkit (Google Play)
  - Vuforia (PTC, korábban Qualcomm)
    - Külön diaanyag!

Kiterjesztett valóság példák

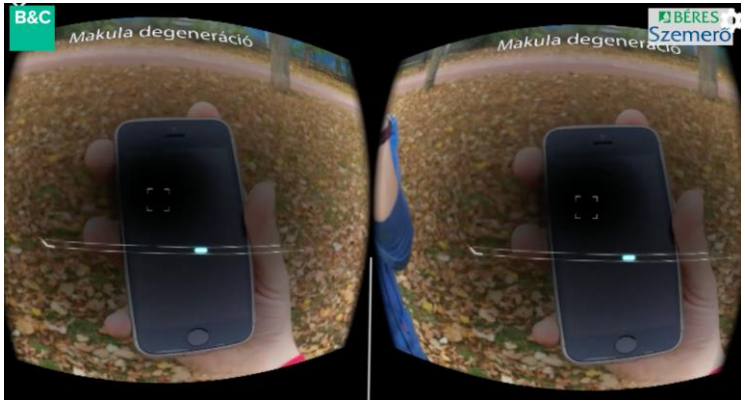
- Szenzor vezérelt grafika
  - Pro Android AR (4. fejezet és 9. fejezet)
  - Vuforia Samples
- How to create an augmented reality Earth in Unity
  - https://www.codeproject.com/Articles/1242369/How-to-create-an-augmented-reality-Earth-in-Unity
- Adding Augmented Reality to an ML Kit-Powered Android App
  - https://www.codeproject.com/Articles/5160659/Adding-Augmented-Reality-to-an-ML-Kit-Powered-Andr



Képelemzés-alapú AR példák

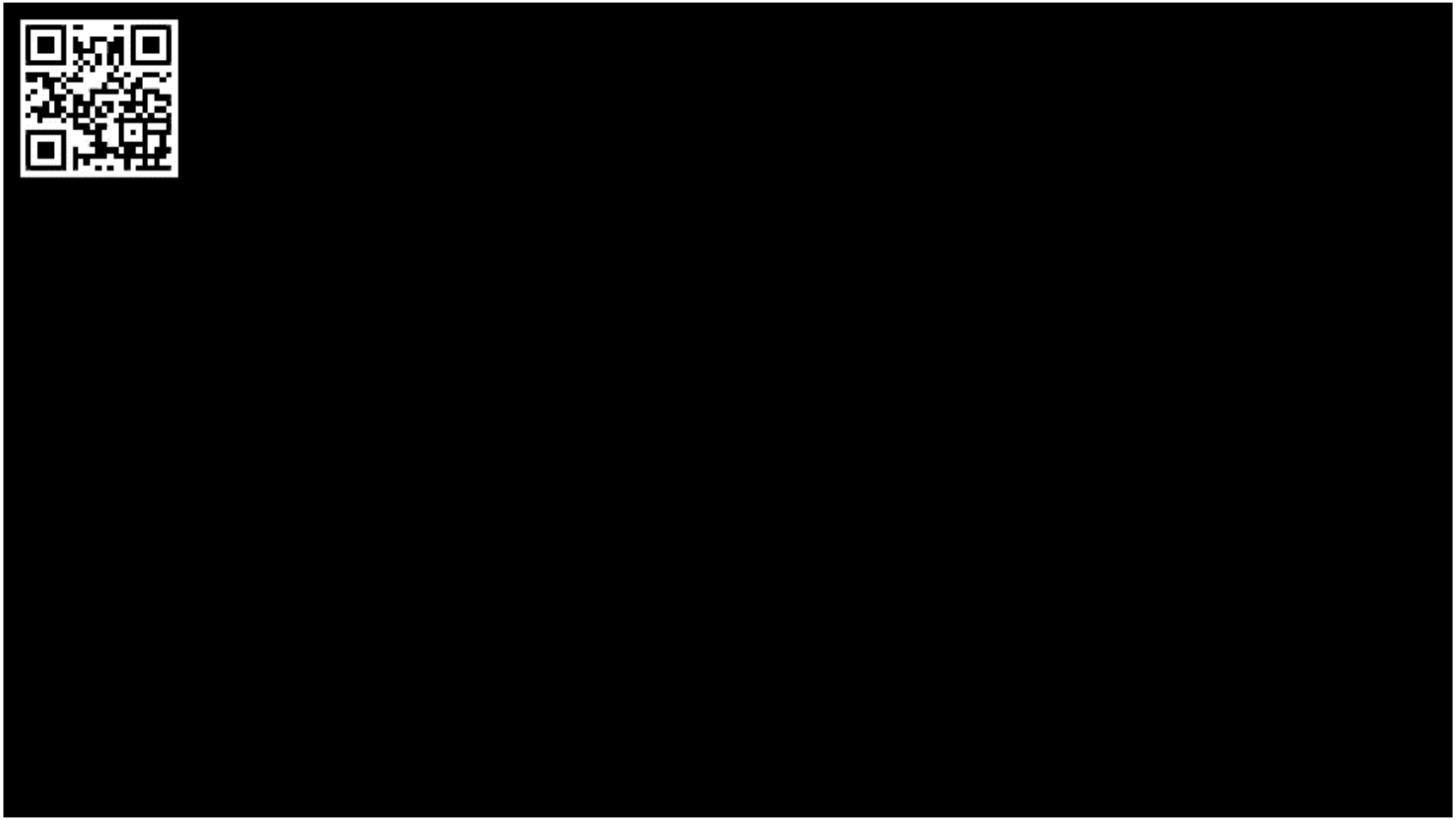
- Google Fordító
  - Korábban WordLens, a Google felvásárolta
  - Kezdetben spanyol-angol fordító
  - Képen detektált szövegek fordítása és a kamera élőképre montírozása

- Augmented Driving
  - Vezetési asszisztencia: sávelhagyás detektálása; gyorsan közeledő jármű a sávban, …



- Látás szimuláció
  - Béres látás szimuláció

WordLens



Környezet feltérképezése

- Célhardverrel
  - 2015: Project Tango RGB-D (mélységérzékelő) szenzorral
  - Belső terek feltérképezésére (néhány méteres távolságig)
  - 2018. tavaszán megszűnik → ARCore (mert az Apple bejelentette, hogy dedikált hardver helyett az ARKit-et fejleszti erre a célra)
  - 2019-től újra kezdenek megjelenni a ToF kamerák
  - 2020, Apple: LiDAR beépítése az új iPAD Pro készülékekbe

- Okostelefon kamera + szenzorok + szoftver réteg
  - ARKit (Apple), ARCore (Google)
  - Színtér értelmezés (számítógépes látás algoritmusok)
    - Vízszintes és függőleges sík objektumok detektálása (talaj, falak, …)
    - Mesterséges 3D objektum elhelyezése rajtuk
    - *Vizuális inerciális odometria:* készülék mozgás képi- és szenzor-alapú követése
    - Megvilágítás felismerés és közelítés
  - Elérhető (lesz) több milliárd okostelefonon
  - Megoldandó a takarások (valós-mesterséges objektumok) megfelelő kezelése!

ShotPro + ARKit

Kiterjesztett valóság (AR) építőelemek

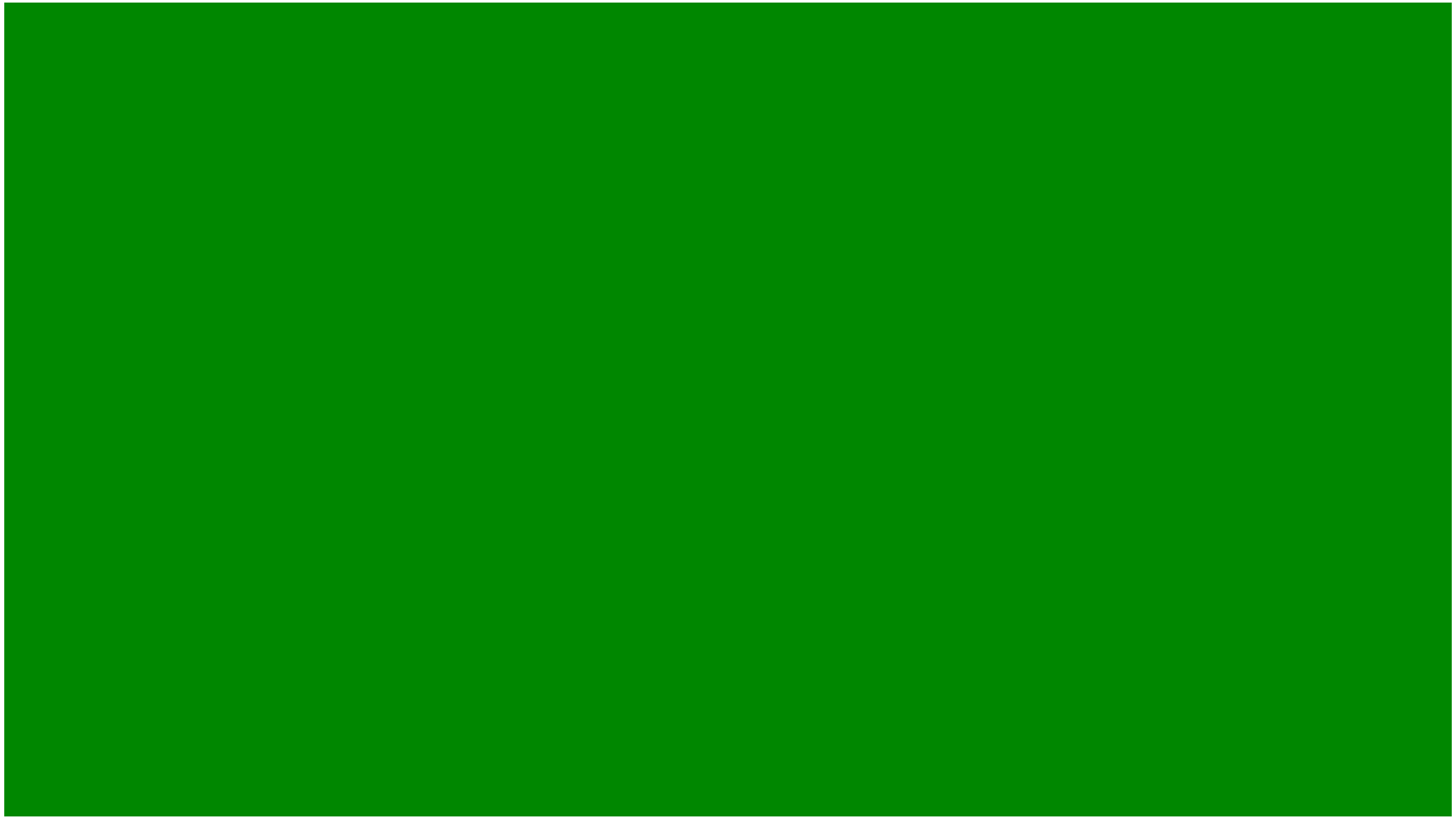
Okostelefon programozás
- Szenzorok + kamera

Számítógépes látás + mesterséges intelligencia
- Színtér értelmezés
- Fények követése

Grafikai modellezés

Forrás: https://www.youtube.com/watch?v=ssMuRqfMwKw

Just a Line AR (Google)



https://justaline.withgoogle.com/
https://github.com/googlecreativelab/justaline-android

Niantic AR (takarások kezelése)

NIANTIC

A kör…

Koncepció videó (egyelőre :D)

ARCore anyagok fejlesztőknek

- Honlap
  - https://developers.google.com/ar/
- Platformok
  - Android, Unity, Unreal, iOS
- Android
  - Minimális feltételek: Legalább 24-es API szint (Android 7), hivatalosan támogatott hardver eszköz
    - Eszköz lista: https://developers.google.com/ar/discover/supported-devices
    - Emulátorból is tesztelhető
  - Demó alkalmazás a javasolt AR irányelvekről
    - ARCore Elements
  - Javasolt kiindulás: `Sceneform` példaprogramok
    - https://github.com/google-ar/sceneform-android-sdk/releases
    - Próbáljuk ki a `hellosceneform` és a `solarsystem` példaprogramokat!

Google Glass

- Fő jellemzők
  - Google kísérleti projekt
  - Szemüveg + kis kijelző
  - Érintés- + hangvezérlés
  - Alkalmazások
  - Okostelefon kapcsolat
  - **Biztonsági és személyiségi problémák!**

- Történet
  - 2012. április: Bejelentés
  - 2013. április: Teszt készülékek elérhetők (1.500 USD)
  - 2015. január: Project leállítva
  - 2015: Project Aura (nem publikus)
  - 2019: Google Glass Enterprise Edition 2

Intel Glass (Vaunt)

- Fő jellemzők
  - Normál szemüveg kinézet: könnyű és kevésbé takar ki részeket
  - Lézeres retinára vetítés: adott irányból látható csak
  - Minimális információ közlésére (időnként a kevesebb több…)

Megszűnt!

Kevert valóság (MR)

- **Kiterjesztett valóság + interakció**
  - A valós világra rávetített mesterséges objektumok **manipulálhatók**
  - Látótérbe újra visszakerülve is az új helyen maradnak
  - Fejlesztői változatok elérhetők
    - Hololens (Microsoft), Magic Leap

Hololens promóciós videó

Magic Leap

- Titokzatos cég
  - Készülékfejlesztés 2011 óta, erős pénzügyi támogatással
  - 2018. augusztus: AR szemüveg előzetes fejlesztői változata elérhető

- Fő jellemzők
  - AR új szintje retinára vetítéssel
  - Kis súlyú headset szenzorokkal
  - Lightpack: kis méretű számítógép
  - Takarások kezelése

MOBIL KÉPELEMZÉS ÉS GRAFIKA

ARC- ÉS EMBERALAK – KÉPELEMZÉS

Tanács Attila

Képfeldolgozás és Számítógépes Grafika Tanszék
Szegedi Tudományegyetem

A tananyag az EFOP-3.5.1-16-2017-00004 pályázat támogatásával készült.

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Arcdetekció és -felismerés

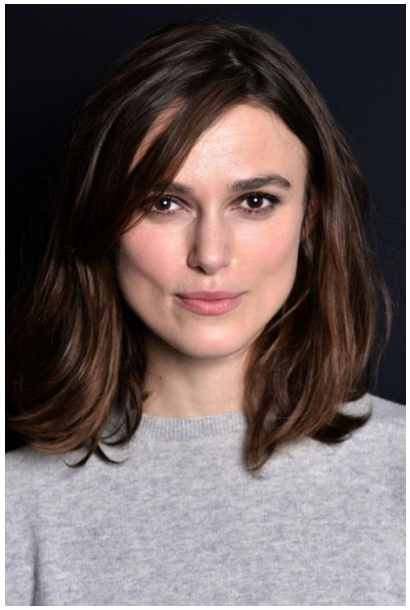
- **Felhasználás**
  - Anonimizálás (pl. StreetView)
  - Fotó „taggelés”: résztvevők detektálása, felismerése
  - Biztonsági funkciók (arc-alapú)
  - Marketing (életkor, nem, … meghatározás alapján)
  - Érzelem felismerése
- **A feladat 2 fő részből áll**
  - Arcképek ***detektálása*** fotókon
  - Az ***arckép-egyezés keresése*** egy meglévő adatbázisban

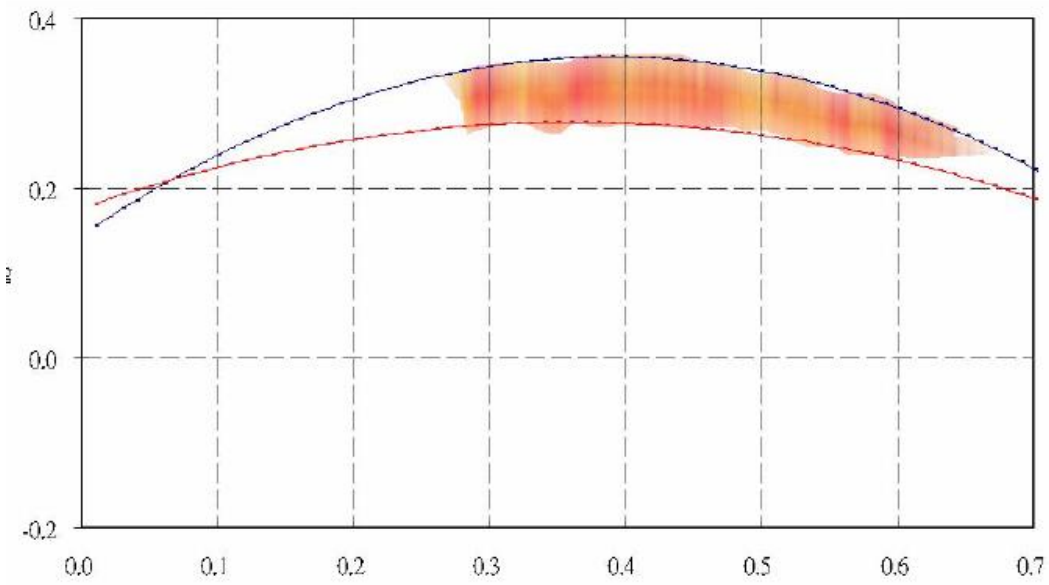
Szín-alapú bőrdetekció

- Normalizált RGB modell:

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B}$$

- A bőrszínek a kromatikus színtérnek csak egy kis részére esnek.
- Önmagában nem elegendő!





Szín-alapú detekció

- Mivel a fehér is beleesik ebbe a területbe ($r = 0{,}33 \mid g = 0{,}33$)
  - Plusz feltétel a világos pixelek szűrésére

$$\textbf{R2: } w = (r - 0{,}33)^2 + (g - 0{,}33)^2 > 0{,}001$$

- A bőr színe nem igen tartalmaz kék pixeleket, ezért egy másik szabály:

$$\textbf{R3: } R > G > B$$

- Sárga-zöld pixelek szűrése szabály:

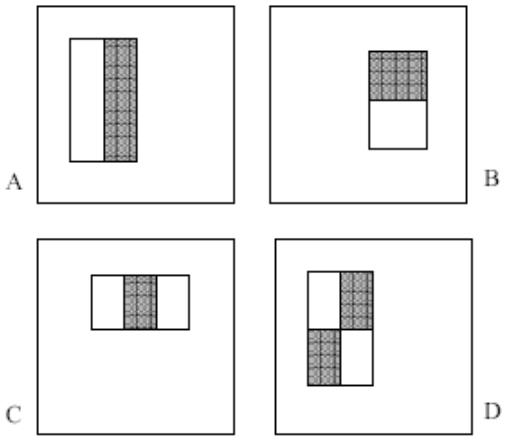
$$\textbf{R4: } R - G \geq 45$$

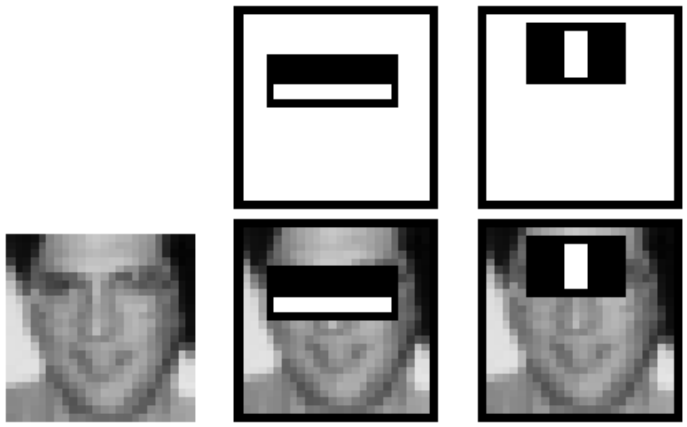
Arc detektálása

- Viola-Jones detektor (2001)
  - Nagyon gyors (valós idejű), magas találati arány
  - *Gyenge osztályozókból* felépített erős osztályozó (AdaBoost)
    - *Gyenge osztályozó* ~ Véletlentől „valamivel” jobb
  - Kaszkádolás
    - Triviális esetek gyors elvetése
  - Haar-szerű jellemzők
    - Téglalap alakú területek intenzitásösszege és -különbsége
- Publikációk
  - Viola, P. and Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Filters. Conference on Computer Vision and Pattern Recognition.
  - Viola, P. and Jones, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision* 57(2), 137-154.

Viola-Jones detektor

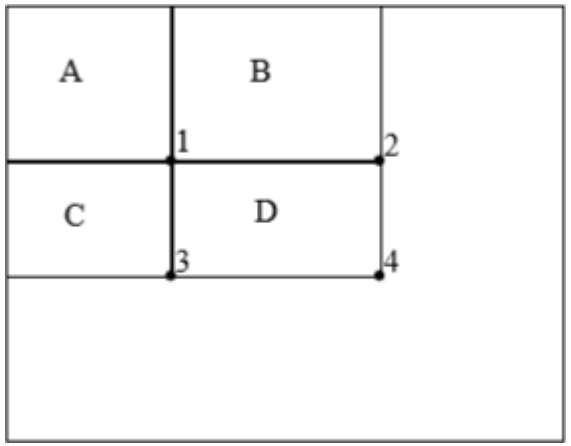
- Haar-szerű jellemzők
  - Különböző méretű, téglalap alakú ablakokban
  - Kép egy ablakában egyes területek intenzitásösszegének különbsége („feketék” – „fehérek”)
    - Szem-arc (sötét-világos); szem-orrnyereg (sötét-világos-sötét); …
  - Hatékonyan számítható
    - Integrál-kép reprezentáció
    - De nagyon nagy számú jellemző kerül kinyerésre. Melyek a jól jellemzők?





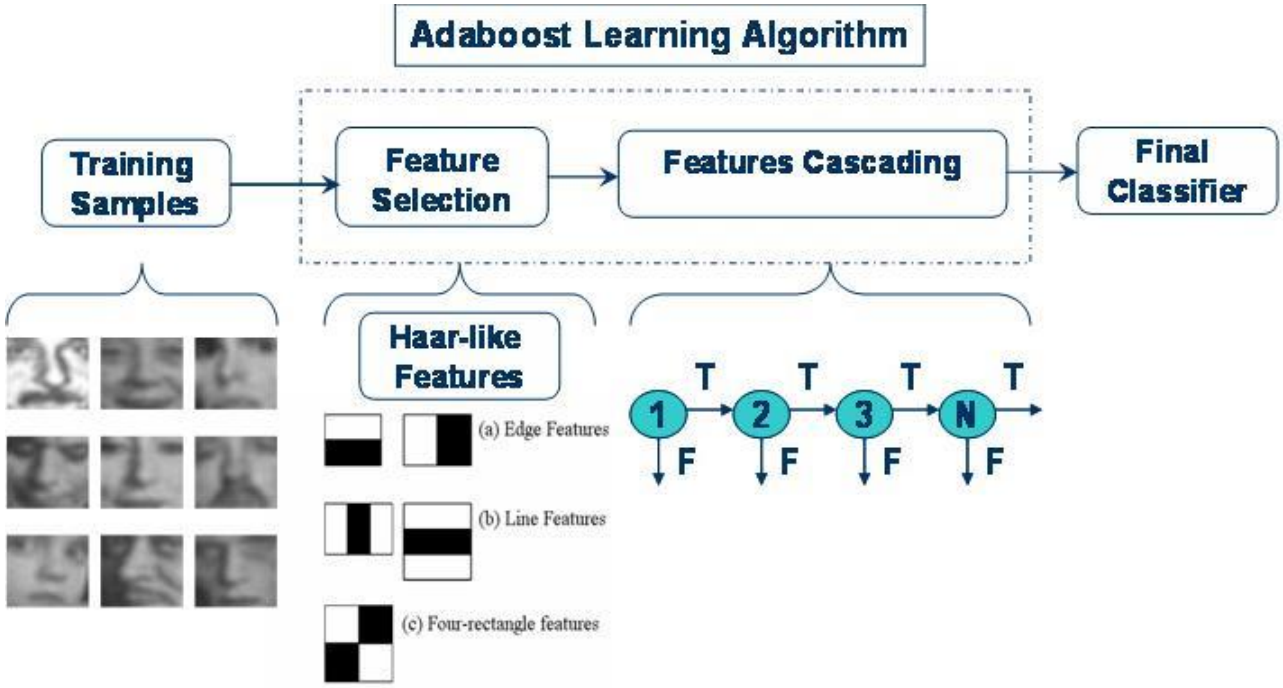
Viola-Jones detektor

- Integrál-kép reprezentáció
  - (x,y) pozíción a tőle „balra” és „felfelé” található intenzitásértékek összege
    - $ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$
    - 1 pásztázással előállítható
  - A „D” téglalapon belüli intenzitások összege 4 tömbolvasással és ezen értékek aritmetikai műveletivel meghatározható
    - $D = P_4 + P_1 - (P_2 + P_3)$

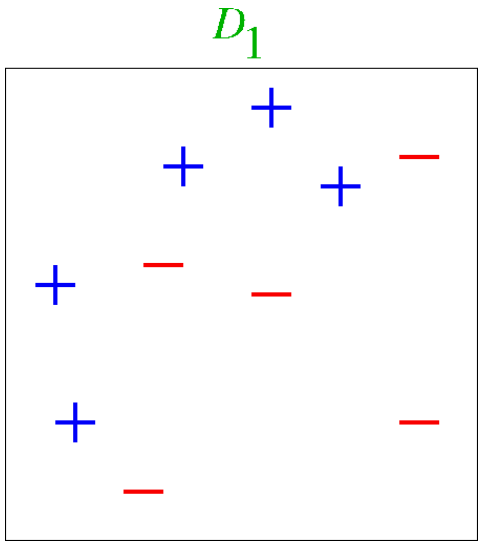


Viola-Jones detektor

- AdaBoost
  - Pozitív és negatív tanító példák (24x24 pixel)
  - Hosszú betanítási idő: napokig/hetekig tarthat
  - Kaszkádolt jellemzők: arcot nem tartalmazó ablakok gyors elvetése

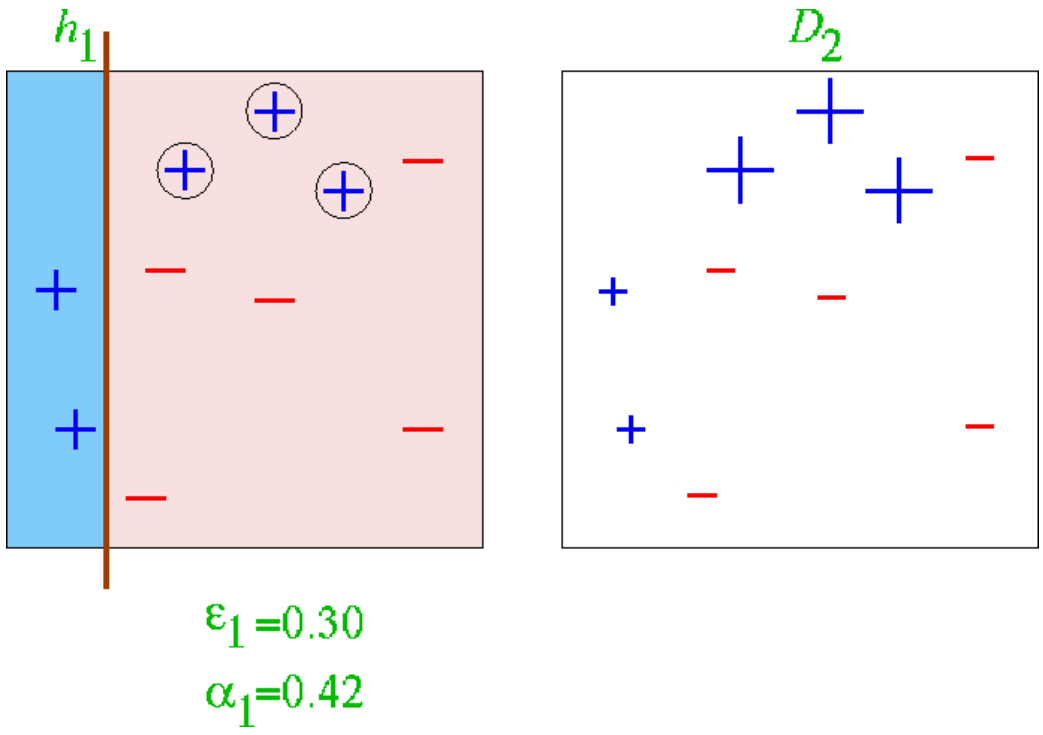


Boosting Example



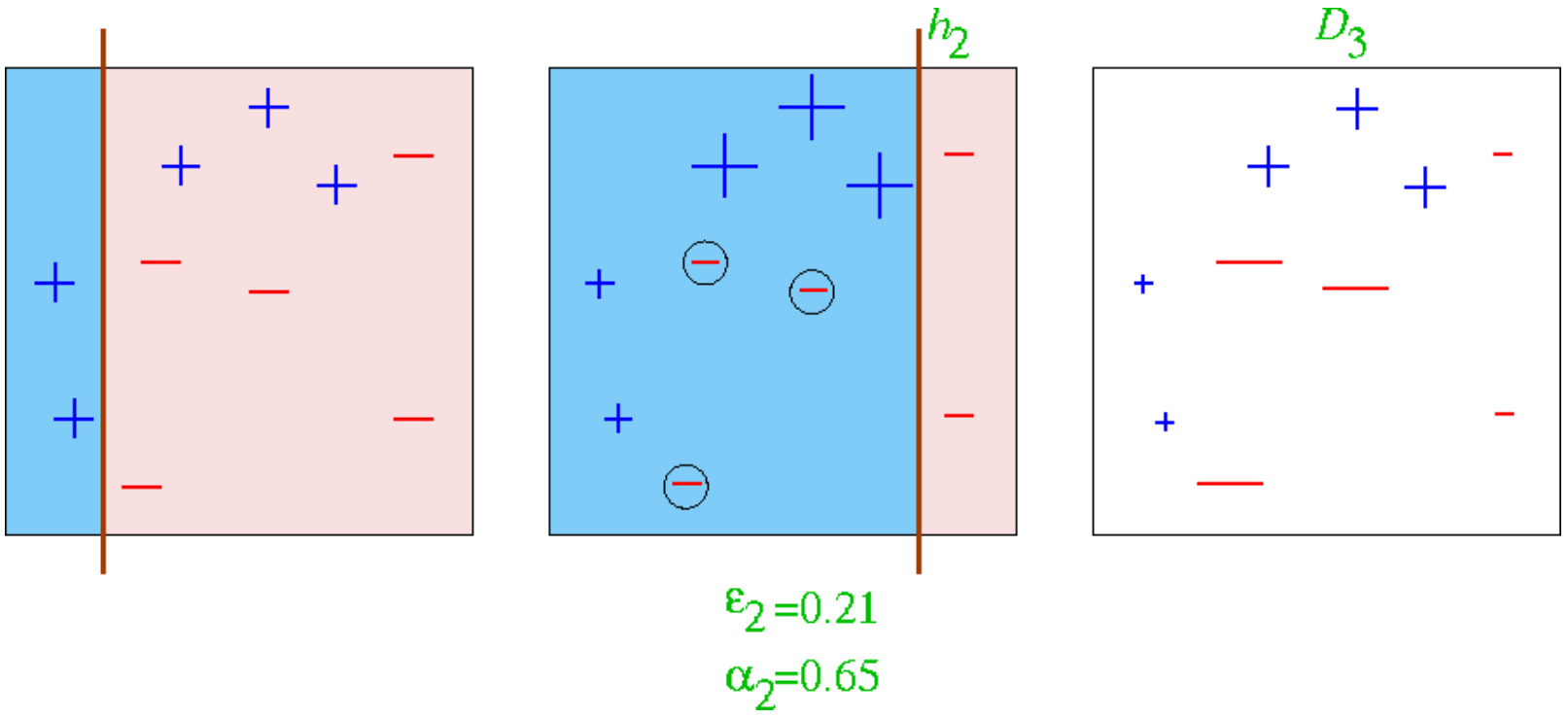
Forrás: Prof. Padhraic Smyth, University of California, Irvine

First classifier



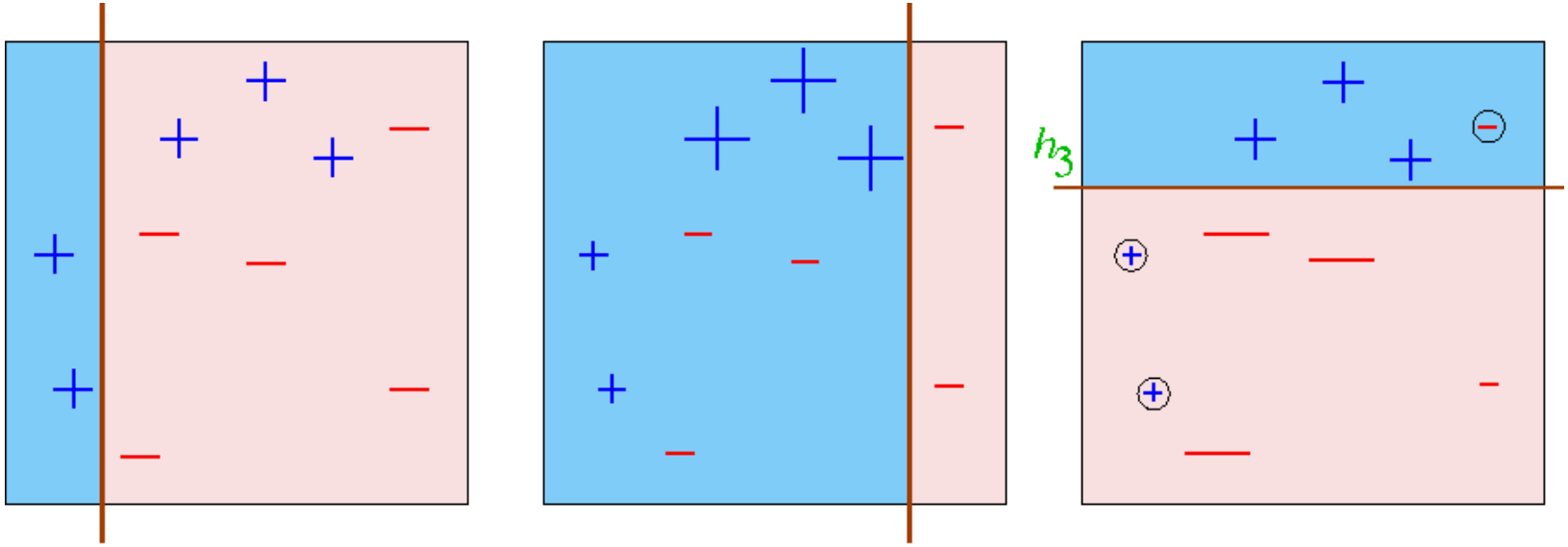
Forrás: Prof. Padhraic Smyth, University of California, Irvine

First 2 classifiers



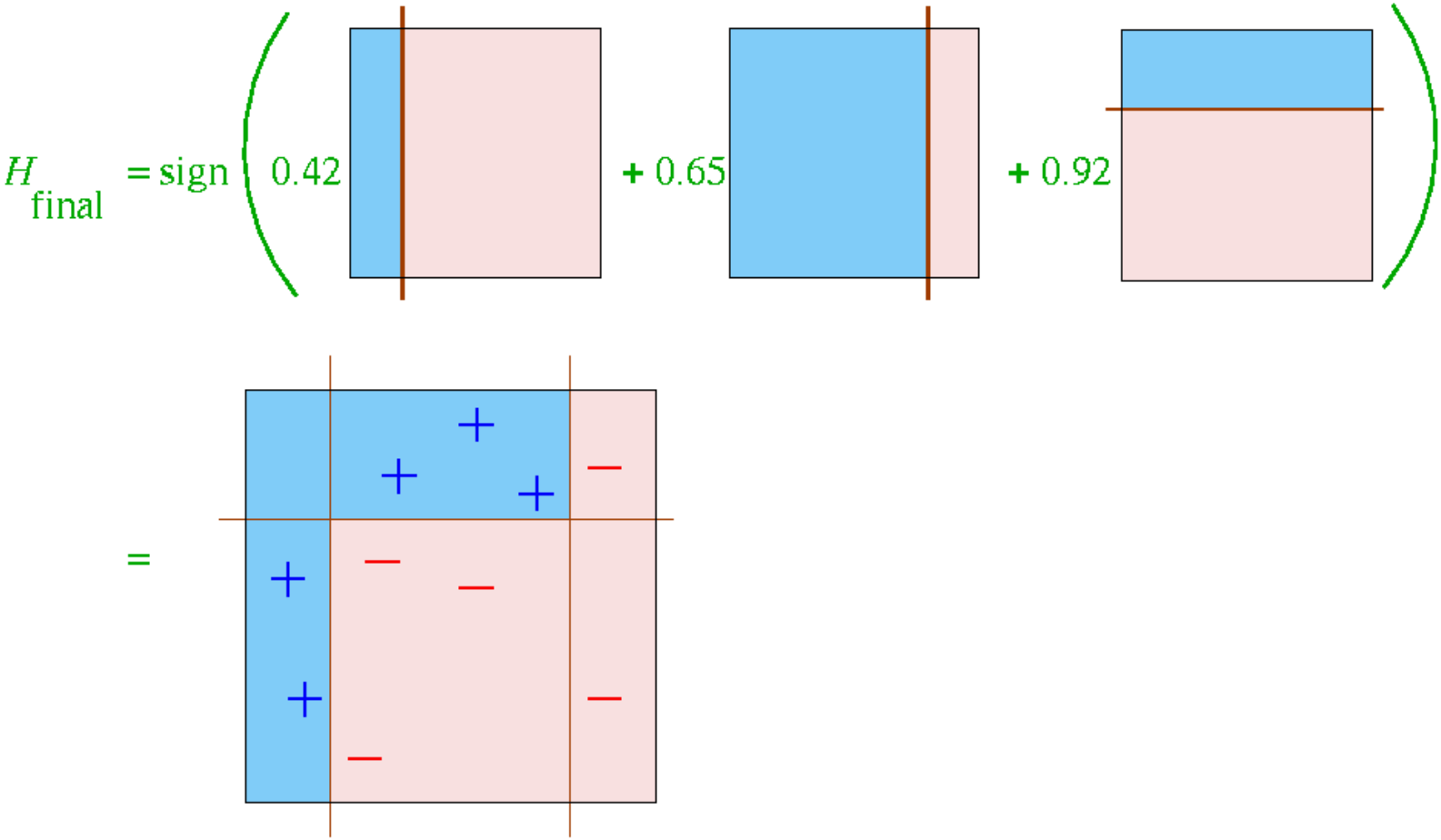
Forrás: Prof. Padhraic Smyth, University of California, Irvine

First 3 classifiers



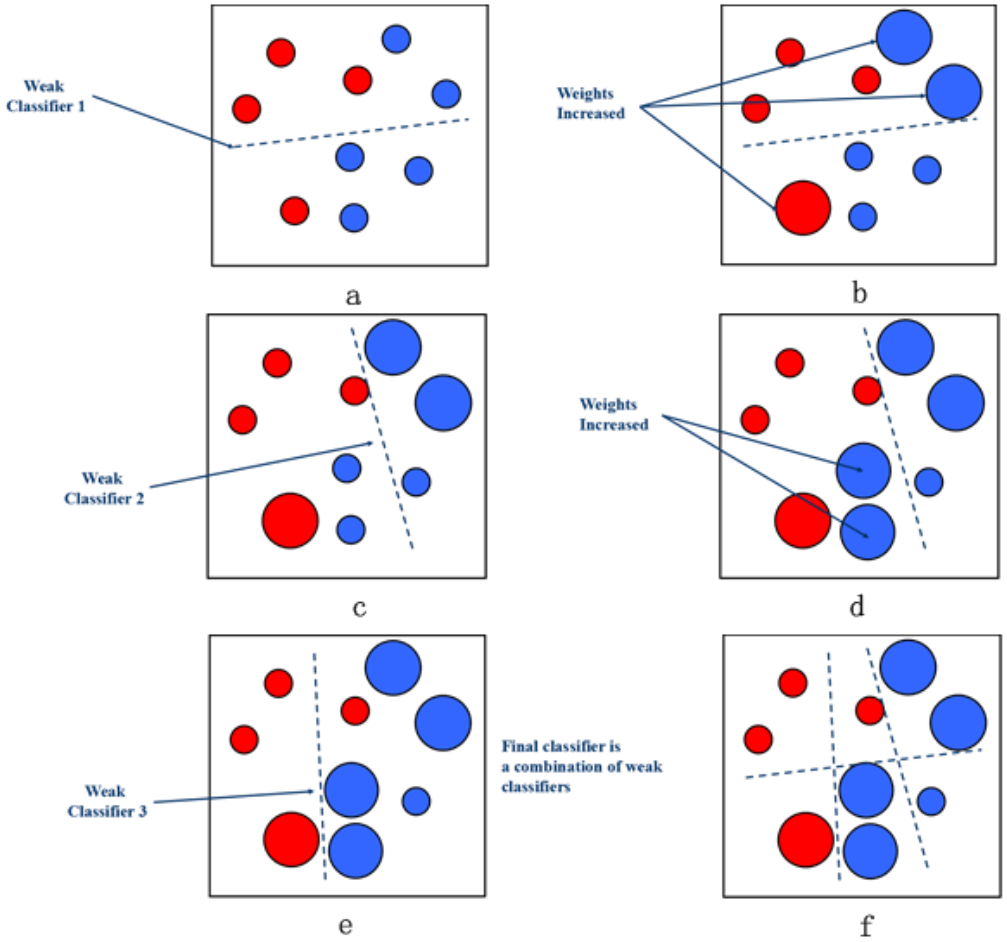
Forrás: Prof. Padhraic Smyth, University of California, Irvine

Final Classifier learned by Boosting



Forrás: Prof. Padhraic Smyth, University of California, Irvine

Boosting (másik példa)



Forrás: https://www.codeproject.com/Articles/4114375/Step-by-Step-Guide-to-Implement-Machine-Learning

Viola-Jones detektor

- Kiértékelés
  - Akár százezres nagyságrendű jellemző
  - Feladat: megfelelő súlyok meghatározása

$$o(x_1, x_2, \ldots, x_{d-1}, x_d) = 1 \quad (\text{ha} \quad w_1 x_1 + \ldots w_d x_d + w_{d+1} > 0)$$

$$= -1 \quad (\text{egyébként})$$

Viola-Jones detektor

- Néhány betanító példa

Forrás: Prof. Padhraic Smyth, University of California, Irvine

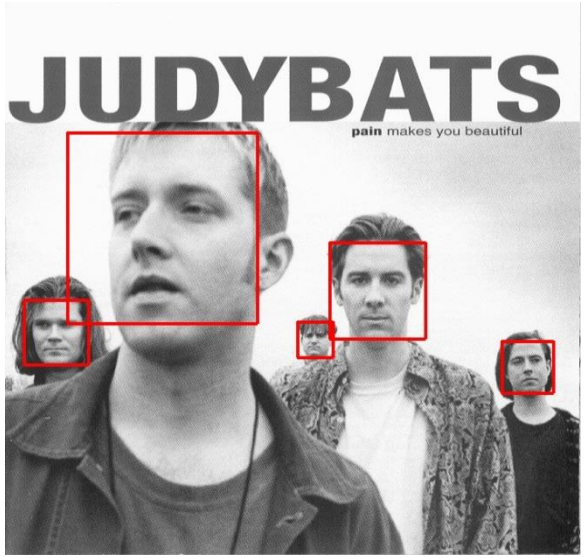
Forrás: University of Cambridge Digital Technology Group. Available online at http://www.cl.cam.ac.uk/research/dtg/attarchive/pub/data/att_faces.zip.

Viola-Jones detektor

- Detekció képeken
  - 24x24 méretű detektor; 38 rétegű kaszkádban, 6060 jellemző
  - Skálázás
    - Detektor méretének skálázása (a kép helyett) 1.25x
  - Pozíció
    - Detektor képpontonkénti mozgatása a képen
  - Végső detekció
    - Több átfedő detekció esetén (legalább pl. 3 darab)
    - Utófeldolgozással ezek összevonása 1 detekcióvá
  - Időigény
    - 384 x 288 méretű képen 0.067 másodperc (2003-ban; 700 MHz CPU)

Viola-Jones detektor

- Eredmények

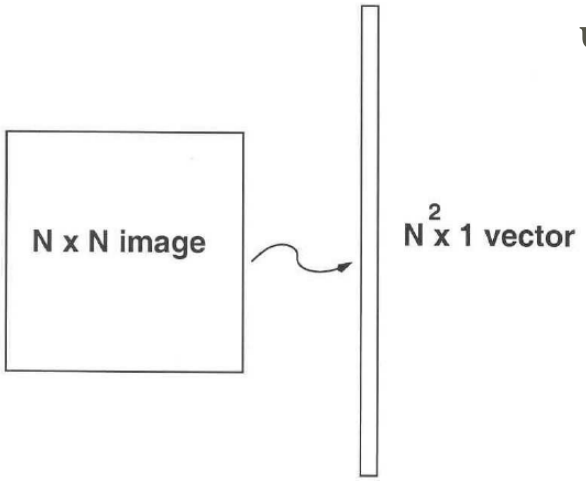


Forrás: Prof. Padhraic Smyth, University of California, Irvine

Arckép keresése

- Eigenface módszer
  - Sajátvektor reprezentáció
- Legyenek $I_1, I_2, I_3, \ldots, I_M$ a tanító adatbázis elemei
  - **FONTOS:** Azonos középpont és méret!
- Képvektorok: $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$
- Átlagkép:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$



Eigenface módszer

- Átlagkép kivonása:

$$\Phi_i = \Gamma_i - \Psi$$

- Kovariancia mátrix:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T$$

- Sajátérték λ_i, sajátvektorok u_i (*eigenfaces*) kiszámítása.
- Ezek a sajátvektorok fejezik ki az arcok közötti eltéréseket. Ez az úgynevezett *arctér* .
- Normalizálás: $\|u_i\| = 1$

Eigenface módszer

- A vizsgált kép: Γ
- Rávetítjük az arctérre, $k = (1, 2, \dots, M)$

$$w_k = u_k^T (\Gamma - \Psi),$$
$$\Omega^T = (w_1, \dots, w_k)$$

- A k. osztály: Ω_k (mindegyik képet rá kell vetíteni az arctérre).
- *Distance from face space* (DFFS) Euklideszi távolság:

$$\varepsilon_k = \|\Omega - \Omega_k\|$$

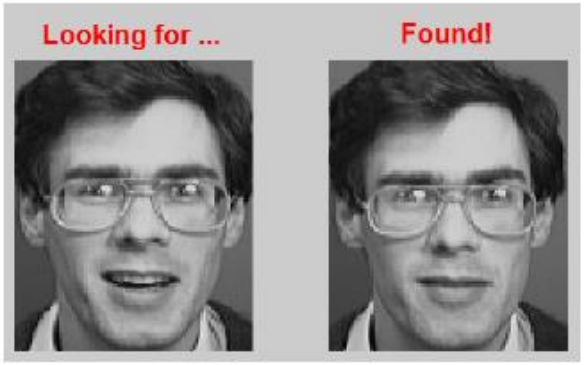
- Küszöb paraméter: θ
- Ha $\varepsilon_k < \theta$, akkor az input kép egy arc

(a) Hasonlóság mértéke: 2.1191×10^{-7}

(b) Hasonlóság mértéke: 1.3257×10^{-7}

(c) Hasonlóság mértéke: 8.3059×10^{-8}

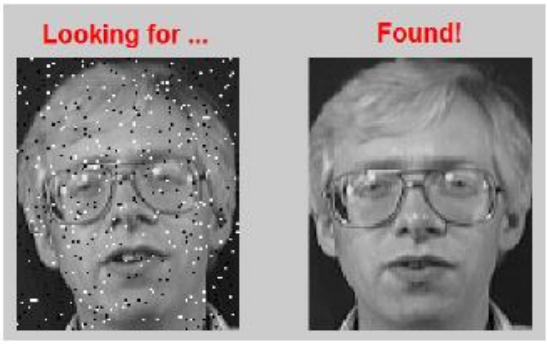
(d) Hasonlóság mértéke: 5.8996×10^{-7}



(e) Hasonlóság mértéke: 1.1971×10^{-7}

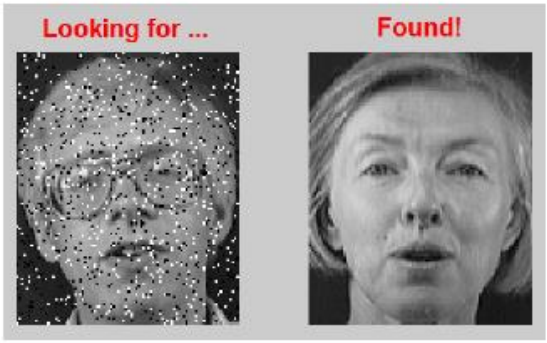
2.4. ábra. A program futásának eredménye az 5 kiragadott arcképre.

(a) 0.02 só-bors zaj
Hasonlóság mértéke: 1.8223×10^{-7}

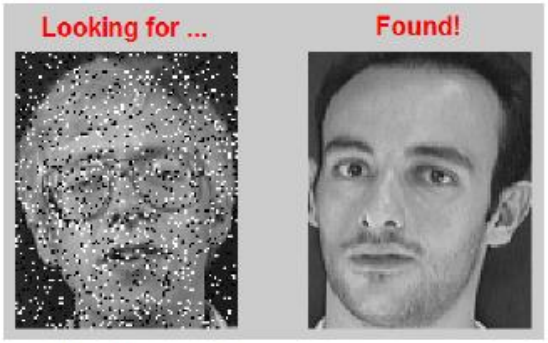


(b) 0.05 só-bors zaj
Hasonlóság mértéke: 1.3174×10^{-7}

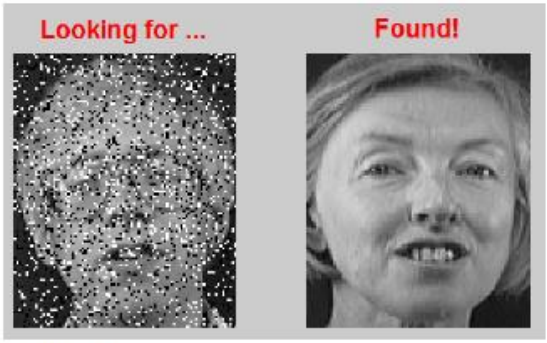
(c) 0.08 só-bors zaj
Hasonlóság mértéke: 9.2335×10^{-8}



(d) 0.1 só-bors zaj
Hasonlóság mértéke: 8.0006×10^{-8}



(e) 0.14 só-bors zaj
Hasonlóság mértéke: 7.5277×10^{-8}



(f) 0.2 só-bors zaj
Hasonlóság mértéke: 6.7974×10^{-8}

2.5. ábra. A program futásának eredménye ugyanarra, de 6 különböző zajjal terhelt arcképre.

Összegzés

- **Mindenképp szolgáltat egy választ** az algoritmus, még akkor is, ha a két kép közötti hasonlóság nagyon kicsi
- ***θ* küszöbparaméter** használatával eldobható lenne a 2.5d, 2.5e, 2.5f ábrán szereplő találat.

További irányok

- Arc változások modellezése
  - http://hvg.hu/tudomany/20140415_Ilyen_a_gyerekkori_fotod_Akkor_ilyen_lesz
  - 1500 személy, különböző életkorokban készült képek
- Facebook DeepFace
  - Kilenc réteg mélységű neuronháló
  - 97% feletti felismerési arány (összemérhető az emberrel)
  - 3D arcmodell készítése
  - http://www.engadget.com/2014/03/18/facebook-deepface/
  - http://vigyazo.blog.hu/2015/02/12/facebook_tetszik_az_arcod_eltarolhatom
- 3D szkennerek által készített képek felhasználása
  - Térbeli információ
  - Egyre gyakoribb állami hivatalokban is

OpenCV lehetőségek

- Gépi tanulás erőteljes támogatása
  - `opencv_haartraining` (régebbi OpenCV verziókban)
  - `opencv_traincascade` (újabb OpenCV verziókban)
    - Haar jellemzők
    - Local binary patterns (LBP) jellemzők
  - Negatív és pozitív példák megadhatók
- Felismeréshez
  - Eigenface reprezentáció mellett Fisherfaces és LBP hisztogram módszerek is támogatottak
- Forrás
  - Fotó vagy mozgókép

Android lehetőségek

- Arcdetektálás fotón / Bitmap-en (API 1 óta)
  - `android.media.FaceDetector` osztály
    - `findFaces` függvény hívása
    - Eredmény `FaceDetector.Faces[]` tömbben
    - Csak RGB_565 formátumú képre!
  - `FaceDetector.Face`
    - Konfidencia (`0` és `1` közötti szám a megbízhatóság becslésére)
    - Szemek közötti távolság (pixelben)
    - Szemek közötti szakasz felezőpontja
    - Arc orientációja (X, Y, Z térbeli tengelyek körül)
  - Példaprogram
    - https://software.intel.com/en-us/blogs/2013/10/28/implementing-face-detection-in-android

Android lehetőségek

- Arcdetektálás kamera élőképen (API 14 felett)
  - `android.hardware.Camera.FaceDetectionListener` interfész
    - Nem minden eszköz implementálja!
    - `getMaxNumDetectedFaces()` kamera paraméter függvény lekérdezése
      - Ha `0`, akkor nem támogatott
  - `Camera.startFaceDetection()` függvény hívására
  - `onFaceDetection(Face[] faces, Camera camera)` függvény
  - `android.hardware.Camera.Face`
    - `id`: arc azonosítója követés esetére
    - `leftEye`, `rightEye`: szem középpont koordináták (Point)
    - `mouth`: száj középpontja (`Point`)
    - `score`: megbízhatósági szint (`int`)
    - `rect`: arc határa (`Rect`)
      - `-1000` és `+1000` közötti érték! (bal-jobb; fent-lent)
      - Szenzor orientációhoz viszonyítva!
      - View koordináta-rendszerbe konvertálás
        - http://developer.android.com/reference/android/hardware/Camera.Face.html#rect

Az új, Camera2 interfész megjelenésével elavultnak lett minősítve!

Android lehetőségek

- Firebase ML Kit
  - https://firebase.google.com/docs/ml-kit
  - https://github.com/firebase/quickstart-android/tree/master/mlkit
- Google Android Vision csomag
  - https://developers.google.com/android/reference/com/google/android/gms/vision/package-summary
  - Vizuális kód, arc, és szöveg kezelésére kamera élőképed
- Arc API
  - https://developers.google.com/vision/android/face-tracker-tutorial
  - Forrás GitHub-on elérhető

További anyagok

- Qualcomm Snapdragon for Android SDK
  - Arckép feldolgozás
    - Pislogás, mosoly detektálása
    - Arc orientáció meghatározása
    - Arcfelismerés (2.1 verziótól)
      - Bemutató
- facereclib
  - https://pypi.python.org/pypi/facereclib
  - Arcfelismerő algoritmusok összehasonlítása standard adatbázisok felhasználásával
- Számos alkalmazás a Google Play áruházban
  - OpenCV demók, „face detection” kulcsszó

Emberalak detekció

- Aktív kutatási terület
  - Elsősorban telepített megfigyelő rendszerek esetében
    - Repülőtereken utasok mozgások követése
    - Gyalogos forgalomfigyelő és –irányító rendszerek
    - Gyanús mozgások felismerése
    - Biometriai azonosítók, gesztusok kinyerése
  - Vezetést segítő alkalmazások
    - Gyalogosok detektálása

Emberalak detekció

- Algoritmus típusok
  - Háttér típusa: rögzített vagy változó
    - A konstans hátteret kivonva megkapjuk a mozgó alakzatokat
  - Holisztikus vagy részeken alapuló
    - Holisztikus: Kép vizsgálata adott méretű ablakokban; detekció, ha az ablakban található részlet valamilyen módon kinyert tulajdonságai egy megadott küszöböt meghaladó mértékben hasonlítanak a keresett objektumot ábrázoló képek jellemző tulajdonságaira
    - Részeken alapuló: egyes testrészekre (fej,törzs,lábak) specializált felismerőket tanítunk be, és ezeket kombináljuk,csak akkor adunk pozitív választ, ha a részfelismerők által visszaadott testrész találatok olyan relatív pozícióban vannak, ami elképzelhető egy ember által felvett póznál.
  - Top-down vagy bottom-up
    - Top-down: pl. csúszóablakos; kijelölt részen belüli vizsgálat
    - Bottom-up: objektumon előforduló részek keresése, majd ezek összevonása

Csúszóablakos technikák

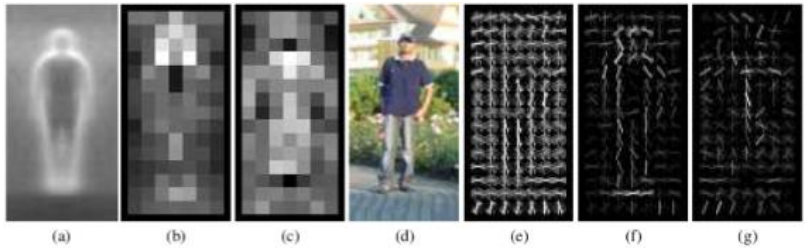
- Csúszóablak
  - A képen egy rögzített méretű ablakban jellemzők kinyerése
  - Ez jellemzően 1:2 oldalarányú, kis méretű téglalap (pl. 64x128 vagy 48x96 pixel), 12-16 pixeles kerettel
  - Jellemzően 4-8 pixeles csúszóablak eltolás
  - Rengeteg ablakot kell kiértékelni → gyors legyen
- Objektum méret
  - Kép kicsinyítése pl. 5%-kal, míg el nem fér 1 ablakban
- Nullhipotézis
  - Nem emberalak látható a csúszóablakban
- Hibafajták
  - Elsőfajú hiba: nem embert ábrázoló részlet emberként felismerve
  - Másodfajú hiba: embert ábrázoló részt nem ismer fel

HOG képjellemző

- Histogram of Oriented Gradients (HOG) képjellemző
  - Dalal, N., Triggs, B.: **Histograms of oriented gradients for human detection**. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Volume:1), pp. 886-893.
  - A Pascal Visual Object Challenge győztese 2005-ben
- Alapja
  - Egy alakzat jól leírható az élek vagy színintezitás-átmenetek irányának eloszlásának felhasználásával, az élek vagy át- menetek pontos helye, iránya nélkül is.
  - Az ablakot cellákra osztjuk, és a cellákon belül kiszámoljuk a színátmenetek irányának hisztogramját, a szögtartomány egy előre meghatározott felosztása felett. Az ablakon belül található cellák hisztogramját összefűzve kapjuk az ablakot leíró vektort.

HOG jellemző

- Osztályozás
  - Jellemző: *k*x*n* elemű vektor (*n* darab cella az ablakon belül, cellánként *k* darab rekesz)
  - Súlyvektor: *k*x*n* elemű vektor
    - Értékeit betanítással kapjuk meg
  - Ezek komponensenkénti szorzatának összegét hasonlítjuk össze egy küszöbértékkel
- Betanítás
  - SVM
  - Latent SVM
    - Rejtett változókkal
    - Deformálható modell
    - Testrészek elhelyezkedése
  - Adaboost + kaszkád



3.2. ábra. Egy emberfelismerő HOGDescriptor és felhasználásának megjelenítése
a) a tanítópéldák kontúrjainak átlaga
b) a súlyvektor értékeinek vizualizálása: egy pixel fényessége az általa lefedett területen található legnagyobb pozitív súlyértékkel arányos
c) mint b), negatív súlyértékekkel
d) a felismerendő kép
e) a képből kinyert HOGDescriptor
f) a kinyert HOGDescriptor és a pozitív súlyok szorzata
g) a kinyert HOGDescriptor és a negatív súlyok szorzata

Emberalak felismerés

- OpenCV lehetőségek
  - HOG, Haar és LBP jellemző vektor kinyerésére és osztályozásra a beépített SVM alapú osztályozók, vagy Cascade classifier-en keresztül

- Képi adatbázisok
  - INRIA Rhône-Alpes Annotated Person Database
  - Penn-Fudan Pedestrian Dataset
  - Pascal Visual Object Classes
  - Daimler Chrysler Pedestrian Dataset

Eredmények

- Android alkalmazás tesztelésre
  - Lőricz Attila diplomamunkája

	C++ osztály	Modell forrása
HOGDescriptor	cv::HOGDescriptor	getDefaultPeopleDetector()
Cascade of rejectors (HOG)	cv::CascadeClassifier	hogcascade.xml
Cascade of rejectors (Haar)	cv::CascadeClassifier	haarcascade_fullbody.xml
PartBased	cv::LatentSvmDetector	Person.xml

5.1. táblázat. Tesztelt felismerők

Hibaszám (FPPI)	Cascade (Haar)	Cascade (HOG)	HOGDescriptor	PartBased
INRIA Neg	313 (0,69)	772 (1,7)	177 (0,39)	148 (0,33)
INRIA Neg szürkeárnyalatos	317 (0,7)	765 (1,7)	59 (0,13)	
Penn-Fudan	180 (1,06)	777 (4,57)	159 (0,94)	149 (0,88)

5.4. táblázat. Elsőfajú hiba összehasonlítása

	Cascade (Haar)	Cascade (HOG)	HOGDescriptor	PartBased
INRIA Pos	534(47,4%)	1014 (90,5%)	841 (74,7%)	1021 (90,4%)
INRIA Pos+tolerancia	360 (32%)	1011 (89,8%)	775 (68,8%)	1018 (90,4%)
INRIA Pos szürkeárnyalatos	543 (48,2%)	1016 (90,2%)	822 (73%)	
Penn-Fudan	80 (18,9%)	257 (60,8%)	207 (48,9%)	303 (71,6%)

5.5. táblázat. Találati arány (Recall rate)

Eredmények

- Sebesség
  - A Latent-SVM alapú módszer a legpontosabb, de nagyságrendileg lassabb

Sebesség	Cascade (Haar)	Cascade (HOG)	HOGDescriptor	PartBased
INRIA Negatív teszt	984 mp	1064 mp	5193 mp	12813 mp
Penn-Fudan teszt	382 mp	441 mp	608 mp	4885 mp
Kamera FPS (320x240) Negatív	1.45 Hz	1.43 Hz	0.9 Hz	0.07 Hz
Kamera FPS (320x240) Pozitív	0.97 Hz	0.93 Hz	0.8 Hz	0.06 Hz

5.6. táblázat. Teljesítmény

Kép visszakeresés feladata

- **Óriási mértékű adatnövekedés**
  - Digitális eszközök tömeges elérhetővé válása
    - Digitális kamerák, kamerás mobil eszközök
  - Internet
    - Képi tartalmak egyszerű megosztási lehetősége

  → Nagy igény automatizált kép visszakeresésre

- **Több évtizedes aktív kutatási terület**
  - CBIR (Content-based image retrieval)
  - QBE (Query by Example)
  - Összefoglaló cikk
    - Ying Liua, Dengsheng Zhanga, Guojun Lua,Wei-Ying Mab. A survey of content-based image retrieval with high-level semantics. Pattern Recognition 40 (2007) pp. 262–282.

Általános célú kép-visszakereső rendszerek

- Szöveg alapú
  - Manuálisan annotált képi adatbázis; keresés az annotáció alapján
    - Sok emberi plusz munka
    - Emberi szubjektivitás a megfogalmazásban
    - Magasabb szintű, absztrakt leírás
  - Internetes keresés: a képek körülvevő szöveg alapján
    - Emberi plusz munkát nem igényel
    - Esetenként nem teljesen releváns a szöveg

- Képi tartalmon alapuló
  - 1980-as évek elejétől kutatott terület
  - Képek indexelése
    - Alacsony szintű képi információk: szín, textúra, alak
    - Magasabb szintű absztrakció bevezetése

Szemantikus rés

- **Szemantikus rés**
  - Általában nincs közvetlen kapcsolat az *alacsony szintű jellemzők*, és az *absztrakt, magasabb szintű leírások* között

- **Képi adatbázis típusok**
  - Specializált
    - Jól definiált, szűk területről származó objektumokról képek
    - Pl. növényi levelek, virágok, autótípusok, …
  - Széles tartalmú
    - Heterogén képállomány
    - Pl. fotómegosztóra feltöltött képek

Szemantikus rés

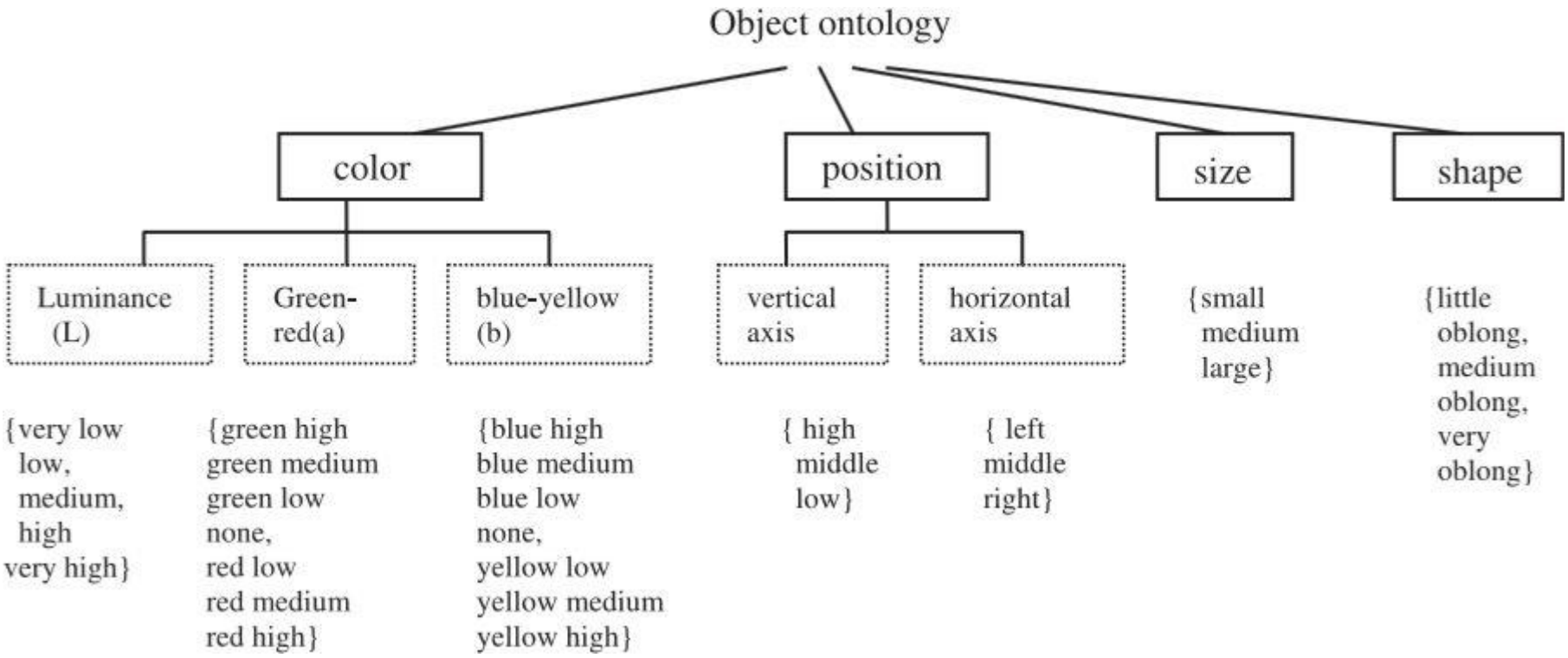
- Három féle keresési szint
  - 1. szint
    - Keresés alacsony szintű képi jellemzők alapján
      - „Keress ehhez hasonló képet!”
      - Keresendő kép vagy skicc megadásával
  - 2. szint (szemantikus keresés)
    - Adott típusú objektumok visszakeresése olyan kinyert jellemzők alapján, amelyekből logikus következtetés végezhető
      - „Keress virágot ábrázoló képet!”
  - 3. szint (szemantikus keresés)
    - Visszakeresés absztrakt attribútumok alapján; széleskörű magas szintű elemzés használatával (események, érzelmek, …).
      - „Keress boldog embereket ábrázoló képeket!”
- A 2. és 3. szinten fontos a szemantikus rés minél jobb áthidalása!

Szemantikus rés csökkentése

- Szemantikus rés csökkentése
  - Magas szintű koncepciók definiálása objektum ontológiával
  - Alacsony szintű jellemzők összekapcsolása keresési koncepcióval gépi tanulási eljárásokkal
  - Relevancia visszajelzés biztosítása a folyamatos tanulás eléréséhez
  - Szemantikus sablon generálása
  - Képi tartalom és azt körülvevő szöveg együttes felhasználása WWW keresésnél
- 3. szintű keresés
  - Kevésbé gyakori, bonyolult
  - Jellemzően specializált adatbázisok esetén
- 2. szintű keresés
  - Gyakoribb
  - Fő lépései: alacsony szintű jellemzők kinyerése; hasonlósági mérték; szemantikus rés csökkentése

Szemantikus rés csökkentése

- Objektum ontológiák
  - Alacsony szintű képi jellemzők egy-egy intervallumához közbülső-szintű leíró
  - Ezen leírók „szótára” az objektum ontológia



Szemantikus rés csökkentése

- Gépi tanulási módszerek
  - Felügyelt tanulás
    - SVM; Bayes osztályozás; neuronhálók
    - Probléma: nagy mennyiségű tanító adathalmaz szükséges, amit nehéz előállítani; a betanítás egy adott adathalmazra vonatkozik
    - Döntési fák (ID3, CART)
  - Felügyelet nélküli tanulás
    - Nincs címke az osztályozás tanulásához; feladat a bementi jellemzők klaszterezésének meghatározása
    - K-means osztályozók
  - Mélytanulás *(„Deep Learning”)*
    - Mai népszerű „buzzword”
    - Nemlineáris feldolgozó egységek többrétegű összefűzése
    - Lehet felügyelt vagy felügyelet nélküli
    - Többszintű képjellemzőkön vagy képi reprezentáción alapul
    - Hierarchikus reprezentáció: a magasabb szintek az alacsonyabbakból kerülnek leszármaztatásra
    - Többszintű absztrakció tanulása (koncepció hierarchia)

Szemantikus rés csökkentése

- Relevancia visszacsatolás
  - Online folyamat: felhasználóval való interakció alapján „tanul”
  - Lépések
    1) Kezdeti találati lista visszaadása
    2) A felhasználó megítéli a találatok helyességét
    3) Gépi tanulási algoritmus feldolgozza a felhasználó visszacsatolást
       - Pl. újrasúlyozás
    - 2) és 3) ismétlése, amíg a felhasználó nem elégedett az eredménnyel
  - QPM (Query Point Movement)

Szemantikus rés csökkentése

- Szemantikus sablon
  - Egy koncepció reprezentatív jellemzője
  - Összegyűjtött mintaképek alapján meghatározva
  - Szemantikus vizuális sablon (SVT)
    - Vizuális sablon: ikongyűjtemény vagy koncepciókat bemutató képek (gyűlés, naplemente, …)
    - Jellemzővektorok kinyerése
    - SVT generálás: Sablon kiválasztása egy koncepcióhoz; objektumok, helyi és időbeli korlátok; minden objektumhoz tartozó jellemző súlyának megadása; felhasználóval való interakció segítségével konvergálás példakeresések egy kis halmazához, ami a koncepcióhoz legjobban tartozik
    - SVT-alapú keresés: ST = {C, F, W}

Vizuális szavak modellje

- Képek
  - Vizuális szavak gyűjteménye
- Szótár készítése
  - Minden betanító halmazbeli képre J_i koordináták meghatározása
    - Pl. kulcspont detektorral, vagy egységes mintavételezéssel
  - Minden pozícióra leíró számítása
    - Pl. SIFT
  - Leíró vektorok klaszterezése K csoportba
    - Pl. K-means algoritmussal
  - Prototípus a szótárban (vizuális szó)
    - Minden klaszterre a vektorok súlypontja
  - Jellemzően több százezer leíróból százas nagyságrendű prototípus készül
    - Jó tömörítés

Vizuális szavak modellje

- Keresés
  - A keresendő képre pozíciók keresése, leírók kiszámítása
  - Legközelebbi vizuális szó választása
- „Bag of words” modell
  - A leírók koordinátáit nem vesszük figyelembe a keresésnél
  - Csak a leíró számít
- Jellemzői
  - Jól működik, de
  - Nem vesz figyelembe pozíció információt (nem mondja meg, hol van a képen a keresett objektum)
  - Feltételezi, hogy a vizuális szavak generálása független egymástól (nem így van)
  - Alkalmatlan többféle objektum leírására 1 képen

„Bag of Words”

- Anyagok, tutorialok
  - http://people.csail.mit.edu/torralba/shortCourseRLOC/
    - Recognizing and Learning Object Categories
    - Awarded the Best Short Course Prize at ICCV 2005
    - ICCV 2005, CVPR 2007, ICCV 2009 konferenciák tutorial anyaga

Kép alapú keresők

- Mobil platformon
  - Rendszerint kliens-szerver alapú megközelítés
  - A mobil eszköz feltölti a képet vagy a kinyert jellemzőket, és a szerver biztosítja az eredményeket
- Android alkalmazások
  - PlinkArt: festmények felismerése (Google felvásárolta)
  - Google Goggles: Híres épületek, festmények azonosítása; vizuális kódok felismerése; névjegykártya adatok felismerése; könyvek, más áruk kép-alapú felismerése; Sudoku megoldó (lefotózott feladat)
  - Google Lens: Google fotók és kamera integráció
- Desktop alkalmazás
  - ImgSeek
    - Windows/Linux rendszerre, webes környezetbe integrálható
    - Képi adatbázist kell készíteni hozzá

Képtartalom szöveges leírása

- Google's AI is getting really good at captioning photos
  - https://www.engadget.com/2016/09/23/googles-ai-is-getting-really-good-at-captioning-photos/

Google Cloud Vision API

- Google Cloud Vision API
  - 2016. tavaszán megnyitásra került
    - Google opens its Vision API in public beta to give any app the power of Google Photos
  - CLOUD VISION API BETA
  - Quickstart
- Sample Applications
  - Label Tagging Using Kubernetes; Landmark Detection Using Google Cloud Storage; Text Detection Using the Vision API
  - Image Detection Using Android Device Photos
- Főbb jellemzők
  - Használata internetkapcsolatot igényel
  - Az ingyenesen használható keret erősen korlátozott

TensorFlow

- TensorFlow
  - A Google nyílt forráskódúvá tett MI-tanítási csomagja
    - GitHub-on elérhető
  - C++, Python API + más nyelvek is támogatottak
  - Desktop PC + mobil környezet
    - https://www.tensorflow.org/mobile/
    - Android, iOS, Raspberry
- Jellemzők
  - Betanítás után lokálisan a készüléken használható
  - Át kell alakítani TensorFlow Lite modellé (nem minden vihető át!)
- Példaprogramok
  - https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android/
  - `TF Classify`, `TF Detect`, `TF Stylize` alkalmazások

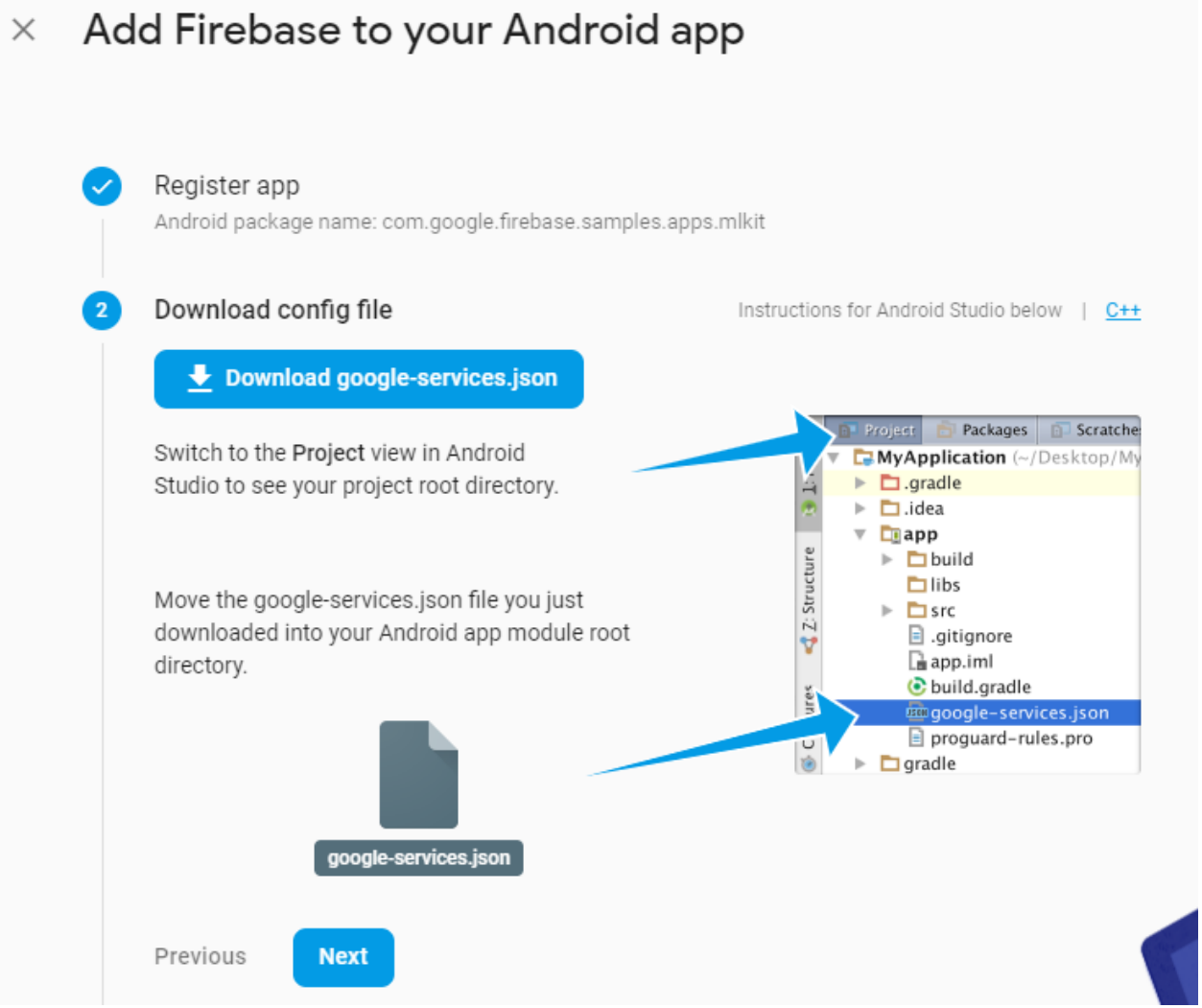
ML Kit for Firebase

- Főbb jellemzők
  - https://firebase.google.com/docs/ml-kit
  - OCR, arcdetekció, helyszínek felismerése, vizuális kódok beolvasása, képek címkézése, szöveg nyelvének felismerése
  - Eszközön vagy felhőben
  - Feltölthető saját TensorFlow Lite modell a Firebase-re
- Lépések
  - Firebase projekt készítése
  - Összekötése az Android alkalmazással
    - Firebase konzolon app azonosító megadása
    - google-services.json fájl letöltése és Android projekthez adása
- Árazás
  - A felhő funkcionalitás használatát mérik
  - Többféle díjcsomag elérhető, köztük ingyenes is
  - Nagymértékű használat esetén nem ingyenes!

Feladat: ML Kit tesztelése

- Töltsük le a Firebase quickstart Android forráskód csomagját
  - https://github.com/firebase/quickstart-android
- Csomagoljuk ki és nyissuk meg az Android Studio-ban
- Böngészőben lépjünk be a Firebase konzolra
  - https://console.firebase.google.com
  - Indítsunk új projektet
  - Adjunk meg tetszőleges nevet, pl. `MLKit-test`
  - A rendszer generál hozzá egy egyedi azonosítót
  - Adjuk meg az Android alkalmazásunk csomagnevét
    - Pl. `com.google.firebase.samples.apps.mlkit`
  - Töltsük le a generált `google-services.json` fájlt
- Másoljuk be a `json` fájlt az `mlkit` Android alkalmazásunk `app` mappájába
- Futtassuk az `mlkit` alkalmazást

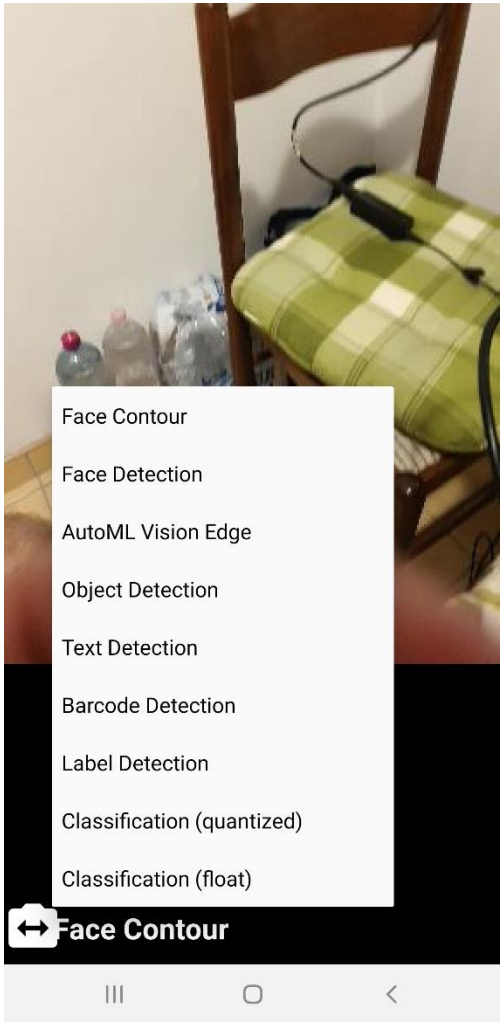
Firebase – Android összekötés

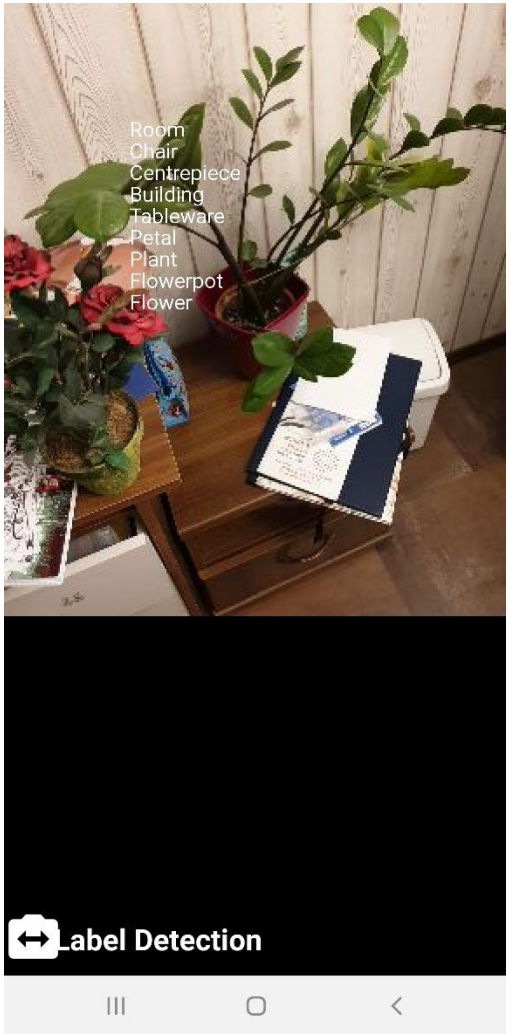


mlkit app futtatás

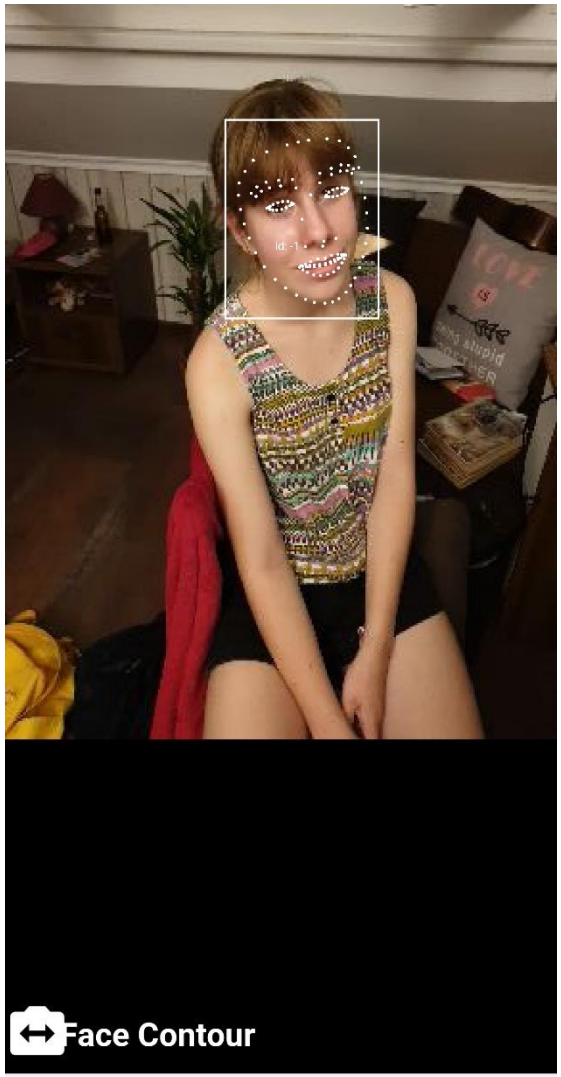
- `LivePreviewActivity`
  - Képelemzés kamera élőképen
- `StillImageActivity`
  - Képelemzés készülékkel készített fotón vagy galériából kiválasztotton

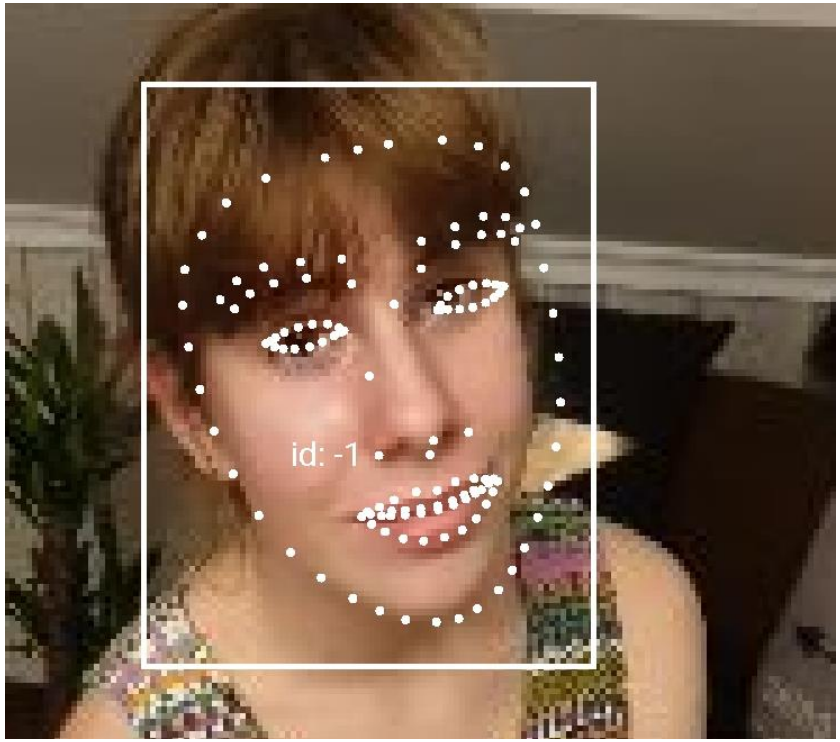
LivePreviewActivity példák





Arc kontúr detekció példa





Importált definíciók

```
import com.google.firebase.ml.vision.objects.FirebaseVisionObjectDetectorOptions;
import com.google.firebase.samples.apps.mlkit.R;
import com.google.firebase.samples.apps.mlkit.common.CameraSource;
import com.google.firebase.samples.apps.mlkit.common.CameraSourcePreview;
import com.google.firebase.samples.apps.mlkit.common.GraphicOverlay;
import com.google.firebase.samples.apps.mlkit.java.automl.AutoMLImageLabelerProcessor;
import com.google.firebase.samples.apps.mlkit.java.barcodescanning.BarcodeScanningProcessor;
import com.google.firebase.samples.apps.mlkit.java.custommodel.CustomImageClassifierProcessor;
import com.google.firebase.samples.apps.mlkit.java.facedetection.FaceContourDetectorProcessor;
import com.google.firebase.samples.apps.mlkit.java.facedetection.FaceDetectionProcessor;
import com.google.firebase.samples.apps.mlkit.java.imagelabeling.ImageLabelingProcessor;
import com.google.firebase.samples.apps.mlkit.java.objectdetection.ObjectDetectorProcessor;
import com.google.firebase.samples.apps.mlkit.java.textrecognition.TextRecognitionProcessor;
```

Használat

- Például arc kontúr detektor objektum beállítása

```
cameraSource.setMachineLearningFrameProcessor(
      new FaceContourDetectorProcessor()
);
```

Face contour inicializálás

```
public class FaceContourDetectorProcessor extends VisionProcessorBase<List<FirebaseVisionFace>> {

    private static final String TAG = "FaceContourDetectorProc";

    private final FirebaseVisionFaceDetector detector;

    public FaceContourDetectorProcessor() {
        FirebaseVisionFaceDetectorOptions options =
                new FirebaseVisionFaceDetectorOptions.Builder()
                        .setPerformanceMode(FirebaseVisionFaceDetectorOptions.FAST)
                        .setContourMode(FirebaseVisionFaceDetectorOptions.ALL_CONTOURS)
                        .build();

        detector = FirebaseVision.getInstance().getVisionFaceDetector(options);
    }

    @Override
    public void stop() {
        try {
            detector.close();
        } catch (IOException e) {
            Log.e(TAG, "Exception thrown while trying to close Face Contour Detector: " + e);
        }
    }

    @Override
    protected Task<List<FirebaseVisionFace>> detectInImage(FirebaseVisionImage image) {
        return detector.detectInImage(image);
    }
```

Élőkép feldolgozása

```
@Override
    protected void onSuccess(
            @Nullable Bitmap originalCameraImage,
            @NonNull List<FirebaseVisionFace> faces,
            @NonNull FrameMetadata frameMetadata,
            @NonNull GraphicOverlay graphicOverlay) {
        graphicOverlay.clear();
        if (originalCameraImage != null) {
            CameraImageGraphic imageGraphic = new CameraImageGraphic(graphicOverlay, originalCameraImage);
            graphicOverlay.add(imageGraphic);
        }
        for (int i = 0; i < faces.size(); ++i) {
            FirebaseVisionFace face = faces.get(i);
            FaceContourGraphic faceGraphic = new FaceContourGraphic(graphicOverlay, face);
            graphicOverlay.add(faceGraphic);
        }
        graphicOverlay.postInvalidate();
    }

    @Override
    protected void onFailure(@NonNull Exception e) {
        Log.e(TAG, "Face detection failed " + e);
    }
}
```

Arc annotáció rajzolása vászonra

```
package com.google.firebase.samples.apps.mlkit.java.facedetection;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import com.google.firebase.ml.vision.face.FirebaseVisionFace;
import com.google.firebase.ml.vision.face.FirebaseVisionFaceContour;
import com.google.firebase.ml.vision.face.FirebaseVisionFaceLandmark;
import com.google.firebase.samples.apps.mlkit.common.GraphicOverlay;
import com.google.firebase.samples.apps.mlkit.common.GraphicOverlay.Graphic;

/** Graphic instance for rendering face contours graphic overlay view. */
public class FaceContourGraphic extends Graphic {

  private static final float FACE_POSITION_RADIUS = 4.0f;
  private static final float ID_TEXT_SIZE = 30.0f;
  private static final float ID_Y_OFFSET = 80.0f;
  private static final float ID_X_OFFSET = -70.0f;
  private static final float BOX_STROKE_WIDTH = 5.0f;

  private final Paint facePositionPaint;
  private final Paint idPaint;
  private final Paint boxPaint;

  private volatile FirebaseVisionFace firebaseVisionFace;
```

Arc annotáció rajzolása vászonra

```
public FaceContourGraphic(GraphicOverlay overlay, FirebaseVisionFace face) {
  super(overlay);

  this.firebaseVisionFace = face;
  final int selectedColor = Color.WHITE;

  facePositionPaint = new Paint();
  facePositionPaint.setColor(selectedColor);

  idPaint = new Paint();
  idPaint.setColor(selectedColor);
  idPaint.setTextSize(ID_TEXT_SIZE);

  boxPaint = new Paint();
  boxPaint.setColor(selectedColor);
  boxPaint.setStyle(Paint.Style.STROKE);
  boxPaint.setStrokeWidth(BOX_STROKE_WIDTH);
}
```

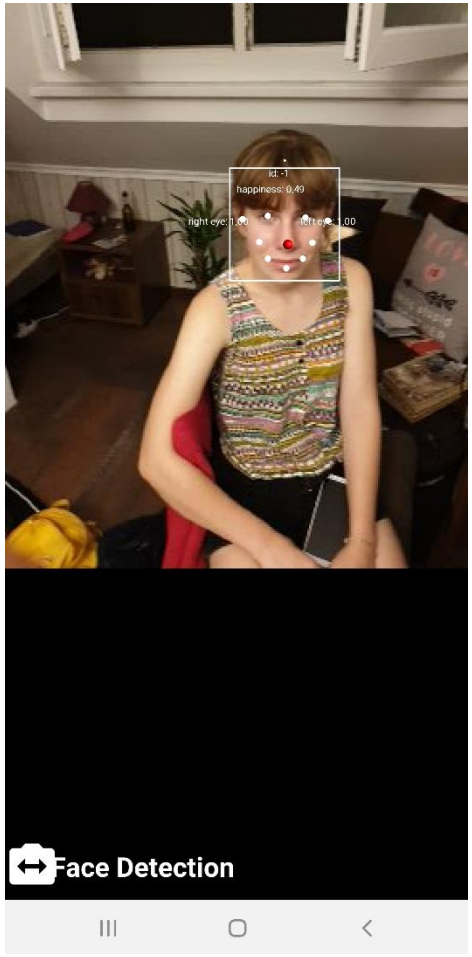
Rajzolás

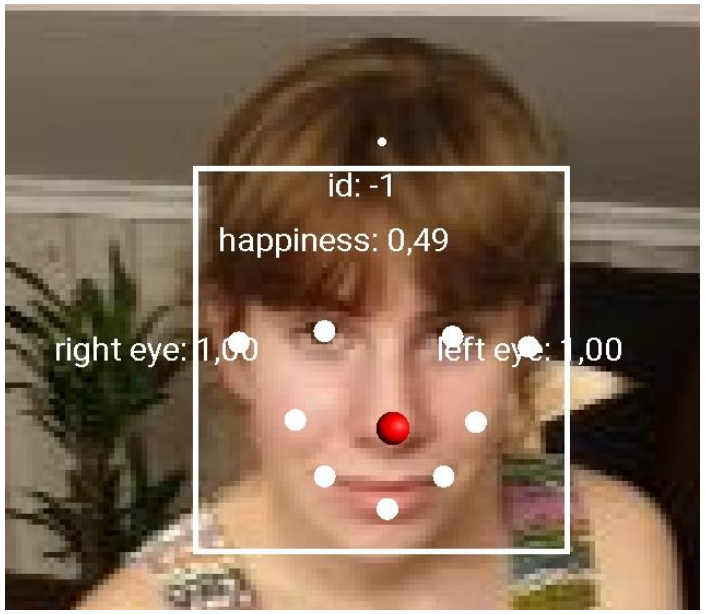
```
/** Draws the face annotations for position on the supplied canvas. */
  @Override
  public void draw(Canvas canvas) {
    FirebaseVisionFace face = firebaseVisionFace;
    if (face == null) {
      return;
    }

    // Draws a circle at the position of the detected face, with the face's track id below.
    float x = translateX(face.getBoundingBox().centerX());
    float y = translateY(face.getBoundingBox().centerY());
    canvas.drawCircle(x, y, FACE_POSITION_RADIUS, facePositionPaint);
    canvas.drawText("id: " + face.getTrackingId(), x + ID_X_OFFSET, y + ID_Y_OFFSET, idPaint);

    // Draws a bounding box around the face.
    float xOffset = scaleX(face.getBoundingBox().width() / 2.0f);
    float yOffset = scaleY(face.getBoundingBox().height() / 2.0f);
    float left = x - xOffset;
    float top = y - yOffset;
    float right = x + xOffset;
    float bottom = y + yOffset;
    canvas.drawRect(left, top, right, bottom, boxPaint);

    FirebaseVisionFaceContour contour = face.getContour(FirebaseVisionFaceContour.ALL_POINTS);
    for (com.google.firebase.ml.vision.common.FirebaseVisionPoint point : contour.getPoints()) {
      float px = translateX(point.getX());
      float py = translateY(point.getY());
      canvas.drawCircle(px, py, FACE_POSITION_RADIUS, facePositionPaint);
    }
```

Arc kontúr vizualizáció

```
if (face.getSmilingProbability() >= 0) {
  canvas.drawText(
      "happiness: " + String.format("%.2f", face.getSmilingProbability()),
      x + ID_X_OFFSET * 3, y - ID_Y_OFFSET, idPaint);
}
if (face.getRightEyeOpenProbability() >= 0) {
  canvas.drawText(
      "right eye: " + String.format("%.2f", face.getRightEyeOpenProbability()),
      x - ID_X_OFFSET, y, idPaint);
}
if (face.getLeftEyeOpenProbability() >= 0) {
  canvas.drawText(
      "left eye: " + String.format("%.2f", face.getLeftEyeOpenProbability()),
      x + ID_X_OFFSET * 6, y, idPaint);
}
FirebaseVisionFaceLandmark leftEye = face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EYE);
if (leftEye != null && leftEye.getPosition() != null) {
  canvas.drawCircle(
      translateX(leftEye.getPosition().getX()),
      translateY(leftEye.getPosition().getY()),
      FACE_POSITION_RADIUS,
      facePositionPaint);
}
FirebaseVisionFaceLandmark rightEye = face.getLandmark(FirebaseVisionFaceLandmark.RIGHT_EYE);
if (rightEye != null && rightEye.getPosition() != null) {
  canvas.drawCircle(
      translateX(rightEye.getPosition().getX()),
      translateY(rightEye.getPosition().getY()),
      FACE_POSITION_RADIUS,
      facePositionPaint);
}
```

Arc detekció

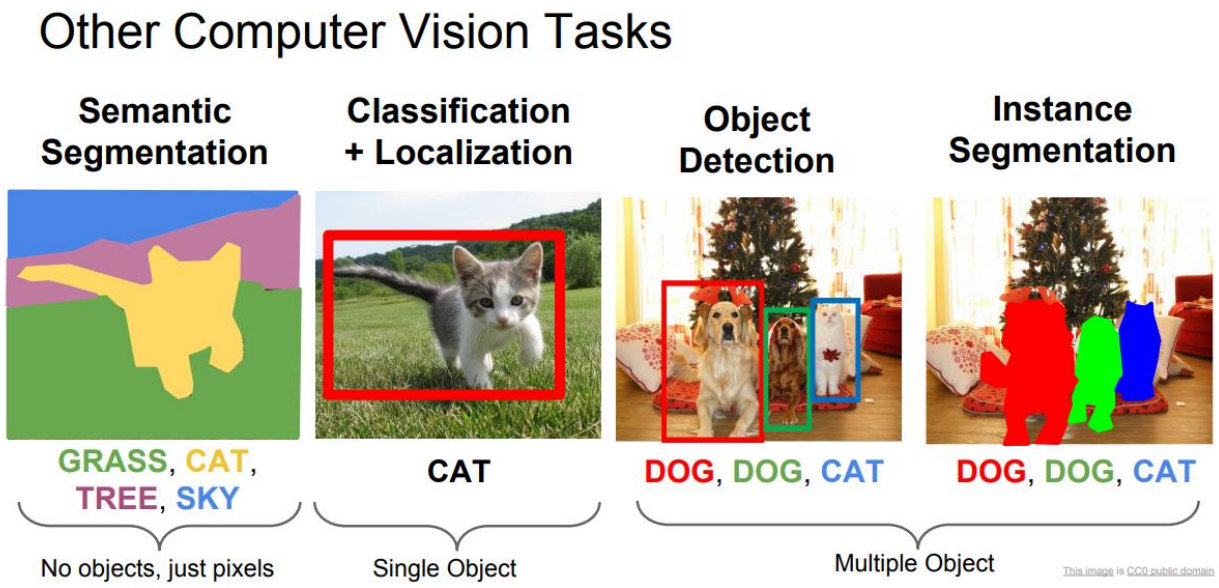




Képfeldolgozó felhő API-k

- Összefoglaló cikk a lehetőségekről
  - Image Recognition APIs: Google, Amazon, IBM, Microsoft, and more
    - https://www.altexsoft.com/blog/image-recognition-apis/

- Feladatok



Source: *Stanford Lecture Slides. Lecture 11: Detection and Segmentation*

Más cégek MI felhő szolgáltatásai

- Amazon Rekognition
  - https://aws.amazon.com/rekognition/
  - Címkézés, tartalom moderálás, szöveg detekció, arc detekció és elemzés, arc-alapú ellenőrzés, híres ember felismerő, mozgáskövetés videón,
- Microsoft Cognitive Services
  - https://azure.microsoft.com/en-us/services/cognitive-services/directory/vision/
- IBM Watson Visual Recognition
  - https://www.ibm.com/watson/services/visual-recognition/demo/#demo
- Huawei ML Kit
  - https://developer.huawei.com/consumer/en/hms/huawei-mlkit
  - Kép osztályozás, objektum detekció és követés, nevezetes hely felismerés, képszegmentálás, vizuális termékkereső
  - Példaprogram
    - https://www.codeproject.com/Articles/5262937/Android-How-to-Develop-an-ID-Photo-DIY-Applet-in-3

Könyvek

- Anubhav Singh, Rimjhim Bhadani: **Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter.** Packt Publishing, 2020.
  - http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2432341
  - Egyetemi hálózatból ingyenesen elérhető
- Benjamin Planche, Eliot Andres: **Hands-On Computer Vision with TensorFlow 2.** Packt Publishing, 2019.
- Antonio Gulli, Amita Kapoor, Sujit Pal: **Deep Learning with TensorFlow 2 and Keras - Second Edition.** Packt Publishing, 2019.