Multimédia, speciális kommunikációs technológiák

Ekler Péter

BME VIK AUT, AutSoft

peter.ekler@aut.bme.hu

hwsw

AutSoft

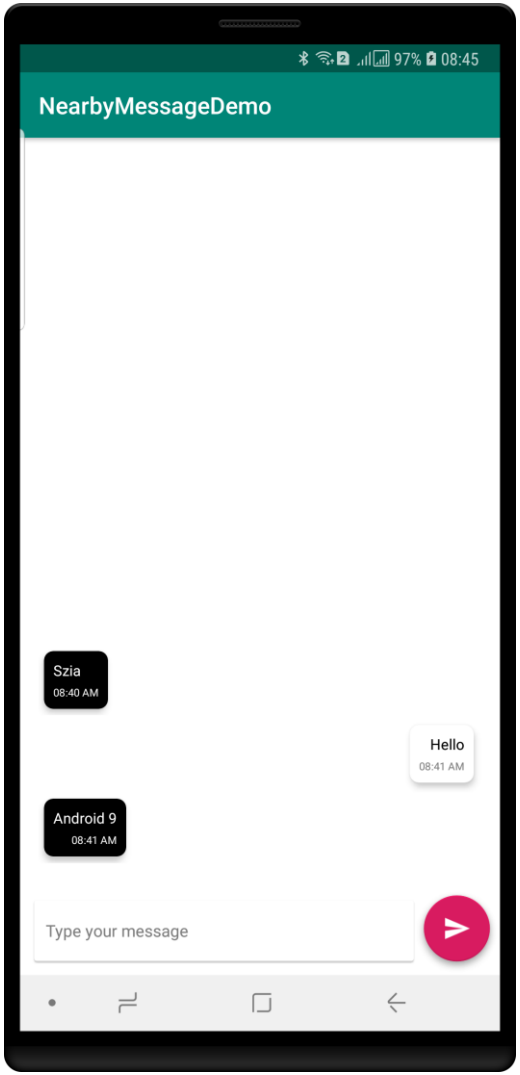
Tematika

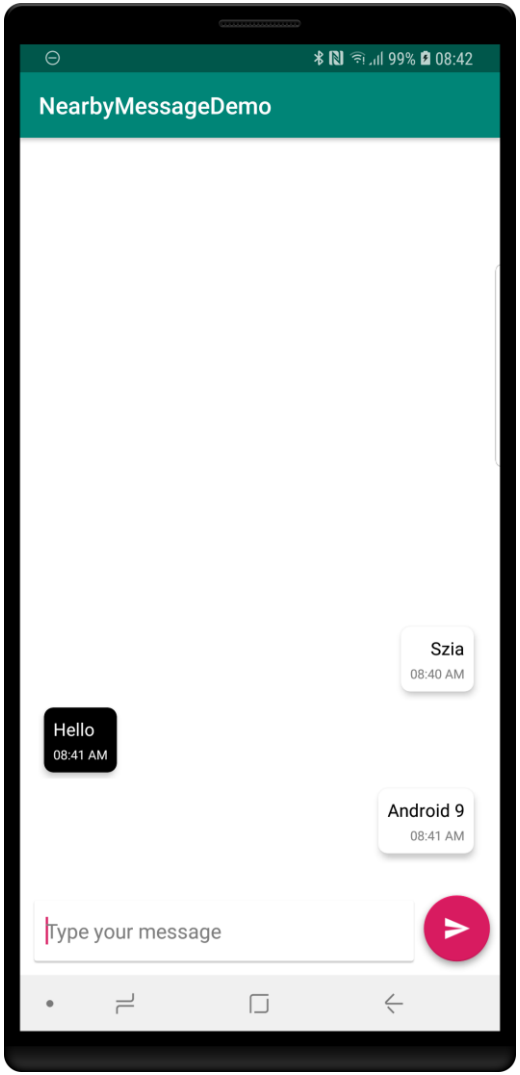
1. Service komponens
2. ContentProvider, Komplex felhasználói felületek
3. Játékfejlesztés
4. Grafikonok, Szenzorok, Külső osztálykönyvtárak, Multimédia alapok
5. Multimédia, további kommunikációs megoldások
6. Biztonságos alkalmazások
7. Android TV és Wear fejlesztés, Kotlin Coroutine
8. Android 9 újdonságok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

Tartalom

- Kamera kezelés
- Képfelismerés – gépi tanulás (MLKit)
  - QR kód olvasás, arcfelismerés, stb.
- Hanglejátszás és hangrögzítés
- Hangfelismerés, szöveg felolvasás
- Hálózati kommunikációs lehetőségek
- NFC
- Bluetooth Beacon
- TCP/IP
- Nearby API

Spoiler ☺


97% 08:45
NearbyMessageDemo
Szia
08:40 AM
Hello
08:41 AM
Android 9
08:41 AM
Type your message


99% 08:42
NearbyMessageDemo
Szia
08:40 AM
Hello
08:41 AM
Android 9
08:41 AM
Type your message

Multimédia képességek

Bevezetés

- Napjainkban a multimédia tartalmaknak meghatározó szerepük van mobil eszközökön
- Fontos a multimédia tartalom előállítása és kezelése is
- Használjuk kreatívan a készülék képességeit
- Mindig gondoljunk a lefoglalt erőforrások felszabadítására!
- A példák nagyon fontosak!

Kamerakezelés

- Széleskörű kamera támogatás
- Gazdag Android API
- Több kamera kezelése
- Különböző kamera típusok kezelése
- Kamera funkciók elérése API-ból (zoom, flash, stb.)
- Kép és video rögzítés
- Használjuk az a kamerát kreatív módon, például:
  - Mozgásérzékelés
  - Kiterjesztett valóság (Augmented Reality)

Kamerahasználat engedélyezése

- Meg kell adnunk a megfelelő *manifest* engedélyeket
- Kamera használata:
  - <uses-permission android:name="android.permission.CAMERA" />
- Kötelező-e a kamera megléte:
  - <uses-feature android:name="android.hardware.camera" android:required= " true/false" />
- Ha el kívánjuk tárolni a médiát a file rendszerben:
  - <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
- Audio felvétele a video tartalomhoz:
  - <uses-permission android:name="android.permission.RECORD_AUDIO" />

Beépített kamera alkalmazás használata

- Leggyorsabb módszer kép/video készítéséhez
- Egy megfelelő *Intent* összeállításával meghívhatjuk az alapértelmezett kamera alkalmazást
- Az *Intent* segítségével elindul a beépített kamera alkalmazás és készít egy képet vagy videót
- Végül a vezérlés visszakerül az alkalmazásunkhoz és az eredmény kiolvasható

Képkészítés lépései

- Megfelelő Intent összeállítása
  - Kép készítése: MediaStore.ACTION_IMAGE_CAPTURE
  - Video készítése: MediaStore.ACTION_VIDEO_CAPTURE
- Intent paraméterek megadása (*putExtra()*)
  - *MediaStore.EXTRA_OUTPUT*: kép mentési helye (Uri), ha nem adjuk meg, egy alapértelmezett könyvtárba menti
- Camera Intent indítása:
  - *startActivityForResult()*
- Visszatérés kezelése
  - Activity-ben *az onActivityResult()* felüldefiniálása
  - A rendszer ezt hívja meg, amikor visszatértünk a kamera alkalmazásból
  - A visszaadott *Intent getData()* függvényével lekérdezhető az alapértelmezett mentés helye

Példa: Fotó készítése beépített kamera alkalmazással

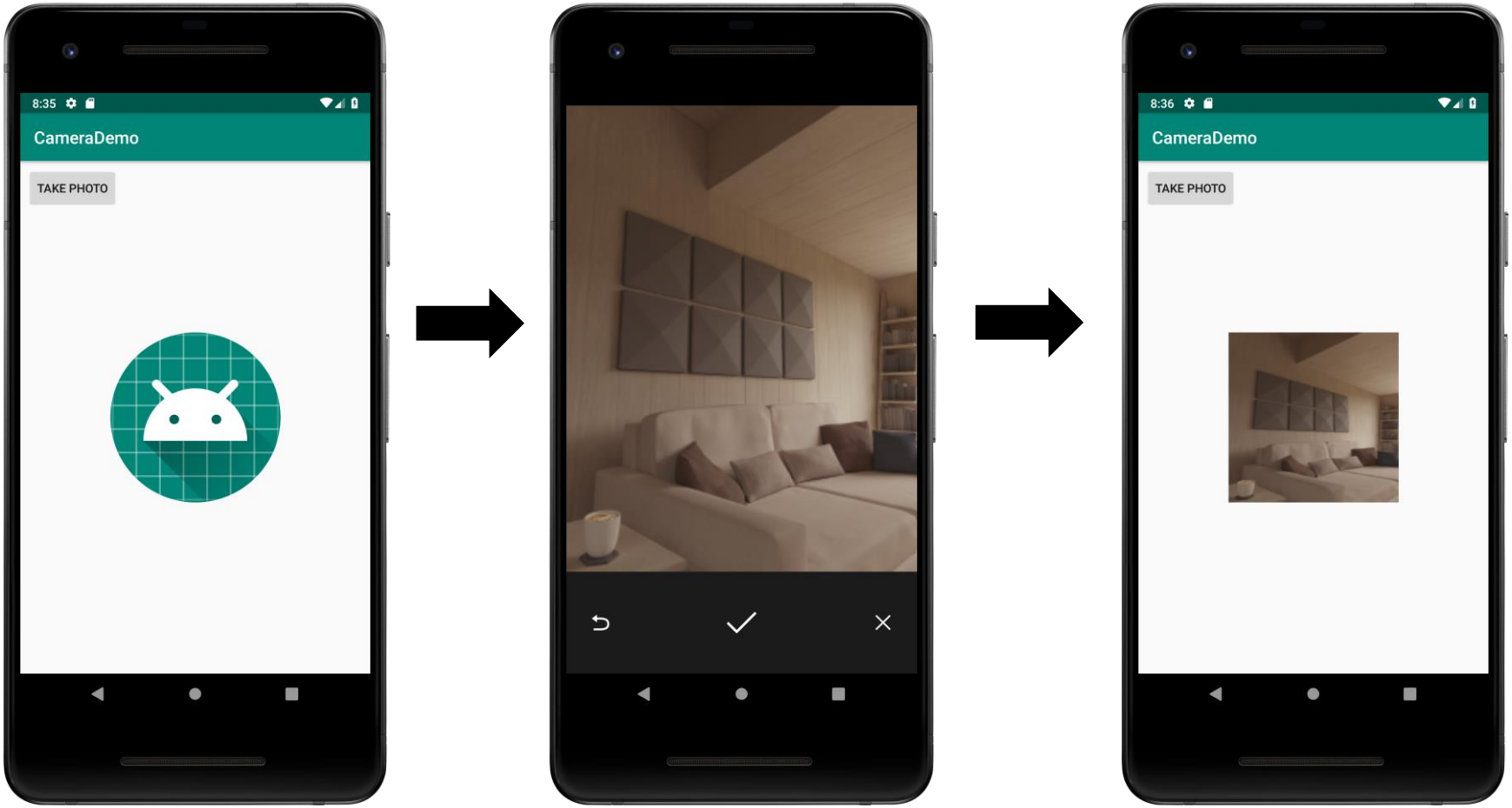
- Fotó készítése

```
btnPhoto.setOnClickListener {
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
        takePictureIntent.resolveActivity(packageManager)?.also {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
        }
    }
}
```

- Válasz kezelése és kép megjelenítése

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        data?.also {
            val imageBitmap = it.extras.get("data") as Bitmap
            ivPhoto.setImageBitmap(imageBitmap)
        }
    }
}
```

Példa: Fotó készítése beépített kamera alkalmazással



Video készítés lépései

- Hasonló, mint a kép készítés
- Intent paraméterek:
  - *MediaStore.EXTRA_OUTPUT*: felvett video állomány helye (Uri)
  - *MediaStore.EXTRA_VIDEO_QUALITY*: 0 és 1 közötti float, 0: legrosszabb minőség és legkisebb file méret
  - *MediaStore.EXTRA_DURATION_LIMIT*: Video hossz korlát másodpercben
  - *MediaStore.EXTRA_SIZE_LIMIT*: Méret korlát a felvett videóra
- Visszatéréskor az Activity onActivityResult(...) függvényében visszaadott Intent getData() függvényével lekérdezhető az alapértelmezett mentés helye

Példa: Visszatérés video felvételből

```
val CAPTURE_VIDEO_REQUEST_CODE = 200

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    if (requestCode == CAPTURE_VIDEO_REQUEST_CODE) {
        if (resultCode == Activity.RESULT_OK) {
            // Video felvéve és elmentve a megfelelő könyvtárba
            Toast.makeText(this,
          "Video mentés helye:\n" + data?.data?, Toast.LENGTH_LONG).show()
        } else if (resultCode == Activity.RESULT_CANCELED) {
            // A felhasználó mégsem mentett videót
        } else {
            // Video felvétel nem sikerült, figyelmeztessük a felhasználót
        }
    }
}
```

Video felvétel programozottan

- *Camera open()* és *realease()* mellett a *lock()* és *unlock()*-ot is kezelnünk kell (4.0-tól már nem! ☺)
- Felvételhez több művelet szükséges, mint a kép készítéshez
- Ügyeljünk arra, hogy ne hagyjunk felesleges video állományt a készüléken, mert sok helyet foglalhat
- **Mindig szabadítsuk fel a Camera eszközt!**

Video lejátszása

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="200dp"
    android:layout_height="200dp"
/>
```

```
VideoView videoView =
(VideoView)this.findViewById(
  R.id.videoView);
MediaController mc =
  new MediaController(this);
videoView.setMediaController(mc);
videoView.setVideoURI(
Uri.parse("https://www.sample-
videos.com/video/mp4/480/big_buck
_bunny_480p_1mb.mp4"));
// videoView.setVideoPath(
//  "/sdcard/movie.mp4");
videoView.requestFocus();
videoView.start();
```

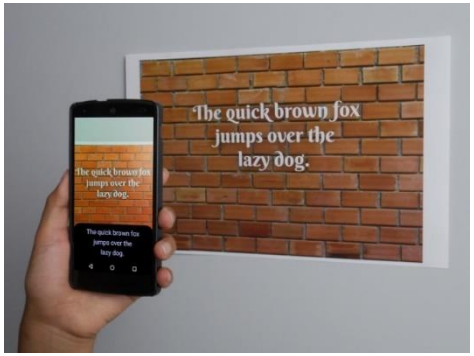
Egyedi kamera nézet készítse

- Camera API V2
  - > https://developer.android.com/reference/android/hardware/camera2/package-summary
- GoldenEye osztálykönyvtár:
  - > https://github.com/infinum/Android-GoldenEye

```
val goldenEye = GoldenEye.Builder(activity).build()

val backCamera = goldenEye.availableCameras.find { it.facing == Facing.BACK }

goldenEye.open(textureView, backCamera, initCallback)

goldenEye.takePicture(pictureCallback)
```

Karakter/szöveg felismerés

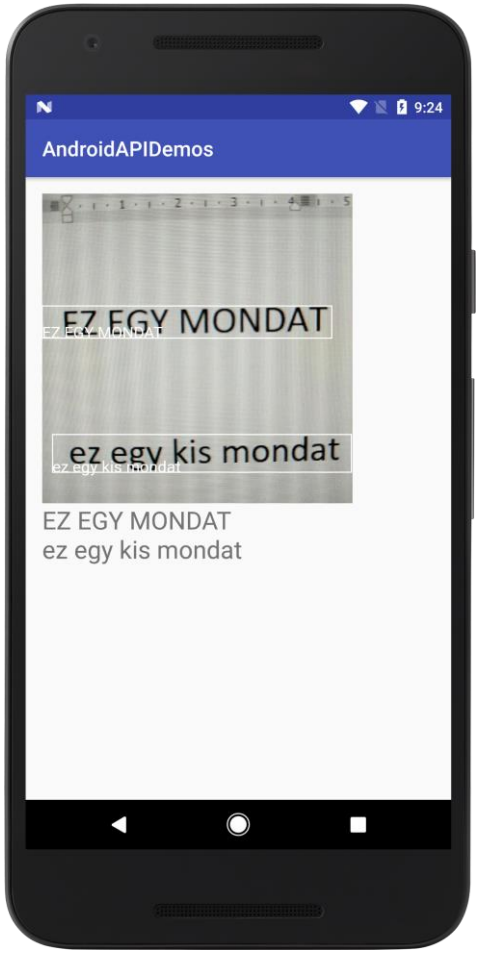
- Play Services -> Vision API



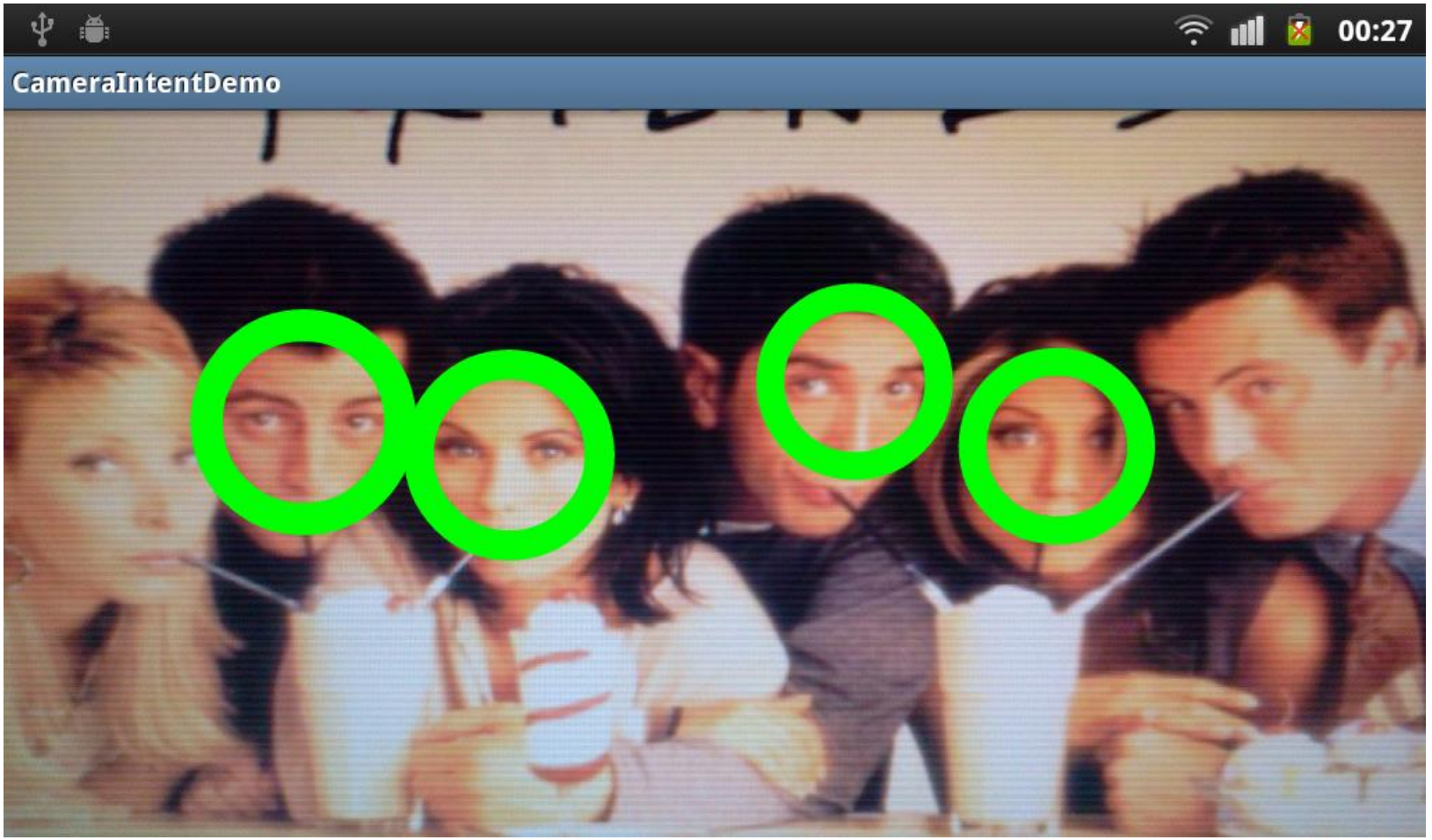
- Arcfelismerés
- QR/Bar/stb kód detektálás
- Karakter/szöveg felismerés
- Gradle:
  - com.google.android.gms:play-services-vision:16.2.1

Szöveg felismerés

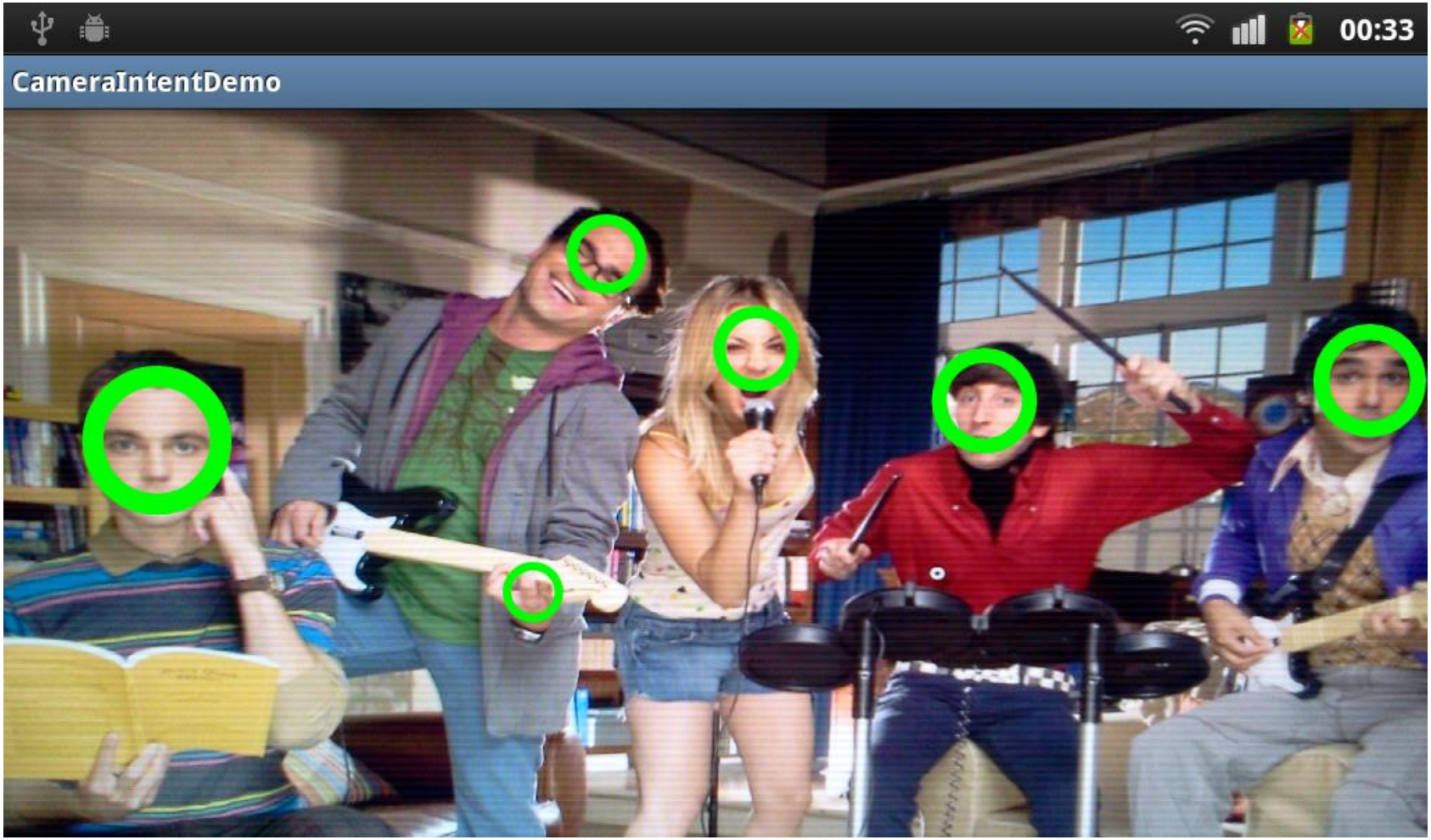
- Szöveg és mondat felismerés
- Szöveg blokkok azonosítása
- Javasolt beállítás:
  - android:keepScreenOn="true"



Arcfelismerés példa 1/2



Arcfelismerés példa 2/2



Arcfelismerés

- Beépített *android.media.FaceDetector* osztály az arcfelismeréshez
- Kép alapján automatikusan detektálja az arcokat (Face osztály)
- Megadható a maximálisan detektálandó arc egy képen
- Nem real time a detektálás!
- Használat:
  - Kamera focus automatikus beállítása
  - Kép effekt alkalmazása az arcokon

Face osztály képességei

- *Confidence* (megbízhatóság): helyes detektálás valószínűsége
- *EyesDistance*: szemek közti távolság
- *MidPoint*: két szem közötti középpont PointF-ben
- *Pose*: A felismert arc Euler szöge a megadott tengelyhez képest (elfordulás a kiválasztott X, Y vagy Z tengelyekhez képest)

Vision API változások

- „Offline” Vision API Deprecated lesz hamarosan
- Áttérés ML Kit-re (Firebase)

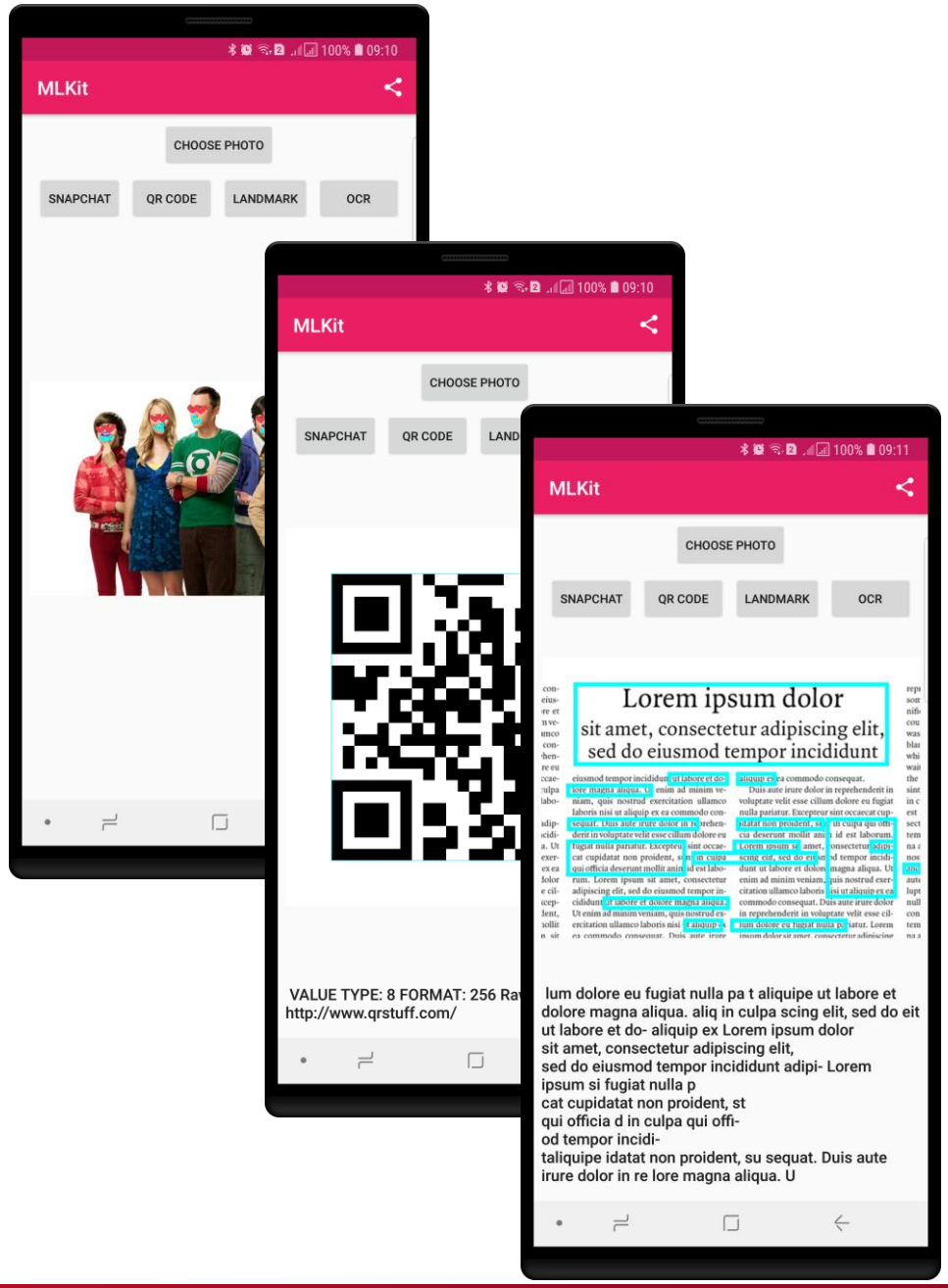
„The Mobile Vision API is now a part of ML Kit. We strongly encourage you to try it out, as it comes with new capabilities like on-device image labeling! Also, note that we ultimately plan to wind down the Mobile Vision API, with all new on-device ML capabilities released via ML Kit.”

ML Kit – Machine Learning

- Gépi tanulás könyvtárak felhőbe kiszervezve
- Alap API-k:
  - Kép azonosítás (objektum, tevékenység, termék, állat, stb.)
  - Szöveg felismerés
  - Arc felismerés
  - Vonalkód olvasás
  - Épületek/ismert helyek azonosítása
  - Smart reply (hamarosan…)
- Firebase integráció szükséges hozzá

ML Kit demo

- Eredeti forrás:
  - https://github.com/riggaroo/android-demo-mlkit
- Teendők:
  - Projekt importálása
  - Firebase projekt létrehozása és google-services.json másolása a projekt *app* modulja alá
  - Firbase consolse-on ML Kit bekapcsolása
- Fizetős funkciók (pl. Landmark)



QR és Bar kód olvasás külső library-val 1/2

- Zxing libary
- Régóta népszerű a fejlesztők körében
- Teljes képernyős és saját nézetes scanner felület
- Valós idejű olvasás
- További információk:
  - https://github.com/zxing/zxing
  - https://github.com/dm77/barcodescanner

QR kód olvasó példa

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity">

    <me.dm7.barcodescanner.zxing.ZXingScannerView
            android:id="@+id/zxingView"
            android:layout_width="match_parent"
            android:layout_height="300dp"/>


    <TextView
            android:id="@+id/tvScan"
            android:textSize="24sp"
            android:autoLink="all"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World!"/>

</LinearLayout>
```

QR kód olvasó példa

```
class MainActivity : AppCompatActivity(), ZXingScannerView.ResultHandler {
    public override fun onCreate(state: Bundle?) {
        super.onCreate(state)
        setContentView(R.layout.activity_main)
    }
    public override fun onResume() {
        super.onResume()
        startCamera()
    }
    @WithPermissions(
        permissions = [Manifest.permission.CAMERA]
    )
    private fun startCamera() {
        zxingView.setResultHandler(this)
        zxingView.startCamera()
    }
    public override fun onPause() {
        super.onPause()
        zxingView.stopCamera(
    }
    override fun handleResult(rawResult: Result) {
        tvScan.text = rawResult.text
        zxingView.resumeCameraPreview(this)
    }
}
```

QR kód olvasó példa

- Modern QR kód generáló szolgáltatás:
  - > https://www.qrcode-monkey.com/

HANGOK LEJÁTSZÁSA ÉS FELVÉTELE

Egyszerű hangok lejátszása

- Az Android platform lehetőséget biztosít arra, hogy egyszerű figyelmeztető hangokat gyorsan és egyszerűen le tudjunk játszani
- Lejátszáshoz szükség van egy *MediaPlayer* objektumra
- *RingtoneManager*: figyelmeztető hangok elérése (Alarm, Notification, RingTone)
- ToneGenerator: bonyolultabb hang szekvenciák előállítása

Példa: figyelmeztető hang lejátszása

```
private fun playNotificationTone() {
    val uriNotif = RingtoneManager.getDefaultUri(
        RingtoneManager.TYPE_NOTIFICATION
    )
    val r = RingtoneManager.getRingtone(
        applicationContext, uriNotif
    )
    r.play()
}
```

Android média lejátszás

- Az Andriod Multimédia API lehetővé teszi, hogy egyszerű módon lejátszunk média tartalmakat különböző helyről:
  - Alkalmazás erőforrás (*res/raw* könyvtár)
  - File
  - Hálózaton keresztül
- A jelenlegi média API hangot csak a standard hangkimeneten v. Bluetooth Headset-en tud lejátszani, hívásba például nem tud hangot bekeverni
- Legfontosabb osztályok:
  - *MediaPlayer*: elsődleges osztály hang és videó lejátszásához
  - *AudioManager*: audio forrás (felvételhez) és kimenet megadása

Média lejátszás erőforrásból

- Nincs szükség *prepare()* hívásra, ebben az esetben a *create()* elvégzi ezt

```
mediaPlayer = MediaPlayer.create(this@MainActivity,
    R.raw.mysound)
mediaPlayer.start()
```

Média lejátszás HTTP URL-ről

```
class MainActivity : AppCompatActivity(), MediaPlayer.OnPreparedListener {

    private var mediaPlayer: MediaPlayer? = null

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

        btnStart.setOnClickListener {
            mediaPlayer = MediaPlayer.create(this@MainActivity,
               Uri.parse("http://babcomaut.aut.bme.hu/tmp/demo.mp3"))
            mediaPlayer?.setOnPreparedListener(this@MainActivity);
        }

        btnStop.setOnClickListener {
            mediaPlayer?.stop();
        }
    }

    override fun onPrepared(player: MediaPlayer) {
        mediaPlayer?.start();
    }


    override fun onStop() {
        mediaPlayer?.stop()
        super.onStop()
    }
}
```

Hangfelvétel folyamata 1/2

1. *android.media.MediaRecorder* példány létrehozása
2. Audio forrás beállítása:
   MediaRecorder.setAudioSource(…), pl.:
   MediaRecorder.AudioSource.MIC
3. Kimeneti formátum beállítása
4. Kimeneti file beállítása
5. Audio encoding beállítása
6. Felvétel előkészítése: *MediaRecorder.prepare()*

Hangfelvétel folyamata 2/2

- Felvétel indítása: *MediaRecorder.start()*
- Felvétel leállítása: *MediaRecorder.stop()*
- Felszabadítás: *MediaRecorder.release()* (**nagyon fontos!**)
- Szükséges engedélyek:
  - <uses-permission android:name="android.permission.RECORD_AUDIO"/>
  - <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

Hangfelvétel példa előkészítés

```
private val fileName = Environment.getExternalStorageDirectory()
   .getAbsolutePath() + "/audiorecordtest.3gp"
private var myPlayer: MediaPlayer? = null
private var myRecorder: MediaRecorder? = null
```

Hangfelvétel indítása

```
private fun startRecording() {
    try {
        myRecorder = MediaRecorder()
        myRecorder?.setAudioSource(
            MediaRecorder.AudioSource.MIC
        )
        myRecorder?.setOutputFormat(
            MediaRecorder.OutputFormat.THREE_GPP
        )
        val outputFile = File(fileName)
        if (outputFile.exists())
            outputFile.delete()
        outputFile.createNewFile()
        myRecorder?.setOutputFile(fileName)

        myRecorder?.setAudioEncoder(
            MediaRecorder.AudioEncoder.AMR_NB
        )
        myRecorder?.prepare()
        myRecorder?.start()
    } catch (e: IOException) {
        Log.e(LOG_TAG, "prepare() failed")
    }
}
```

Hangfelvétel leállítása

```
private fun stopRecording() {
    myRecorder?.stop()
    myRecorder?.release()
}
```

Hanglejátszás indítása

```
private fun startPlaying() {
    myPlayer = MediaPlayer()
    try {
        myPlayer?.setDataSource(fileName)
        myPlayer?.prepare()
        myPlayer?.start()
    } catch (e: IOException) {
        Log.e(LOG_TAG, "prepare() failed")
    }
}
```

Hanglejátszás befejezése

```
private fun stopPlaying() {
    myPlayer?.release()
}
```

Hangfelismerés

- SpeachRecognition API
- Folyamat:
  - Hangfelismerés indítása
  - Értesítések a felismerés állapotáról: indítás, vége, hiba, stb.
  - Eredmény egy „String tömb”

Hangfelismerés példa - indítás

```
val sr = android.speech.SpeechRecognizer
    .createSpeechRecognizer(this)
sr.setRecognitionListener(SpeechRecognizer())
…
val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
intent.putExtra(
    RecognizerIntent.EXTRA_LANGUAGE_MODEL,
    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
)
//intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_PREFERENCE,
//        "hu-HU");
intent.putExtra(
    RecognizerIntent.EXTRA_CALLING_PACKAGE,
    "hu.aut.android.ttsvoicerecogkotlin"
)

intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5)
sr.startListening(intent)
```

Hangfelismerés példa - eredmények

```
override fun onResults(results: Bundle) {
    val str = String()
    Log.d(TAG, "onResults $results")
    val data = results
        .getStringArrayList(
            android.speech.SpeechRecognizer.RESULTS_RECOGNITION
        )
    tvDetectedText.text = ""

    for (text in data) {
        tvDetectedText.append(text + "\n")

    }
}
```

TextToSpeach

- Szövegfelolvasó API
- Google Translate-n ismert megoldás
- Nyelveket külön kell beszerezni
- Egyszerű használat
- Indítás előtt meg kell várni az inicializálás befejeződését

Példa: TexToSpeach 1/3

- onCreate(...)-be:

```
tts = TextToSpeech(this, this)

btnRead.setOnClickListener { speak(etData.text.toString()) }
```

```
private fun speak(text: String) {
    tts.speak(text, TextToSpeech.QUEUE_FLUSH, null)
}
```

Példa: TexToSpeach 2/3

```
override fun onInit(status: Int) {
    if (status == TextToSpeech.SUCCESS) {

        //int result = tts.setLanguage(Locale.forLanguageTag("hu-HU"));
        val result = tts.setLanguage(Locale.forLanguageTag("en-EN"))

        // tts.setSpeechRate((float) 0.8);
        // tts.setPitch(1.0f); tts.setPitch(1.1f);

        if (result == TextToSpeech.LANG_MISSING_DATA || result ==
          TextToSpeech.LANG_NOT_SUPPORTED) {
            val installIntent = Intent()
            installIntent.action = TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA
            startActivity(installIntent)
        } else {
            speak("Speech system works perfectly!")
        }

    } else {
        val installIntent = Intent()
        installIntent.action = TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA
        startActivity(installIntent)
    }
}
```

Példa: TexToSpeach 3/3

```
override fun onDestroy() {
    super.onDestroy()

    try {
        tts.stop()
        tts.shutdown()
    } catch (e: Exception) {
        e.printStackTrace()
    }

    try {
        sr.destroy()
    } catch (e: Exception) {
        e.printStackTrace()
    }

}
```

Média használati javaslatok

- A multimédia kezelő mobil alkalmazások száma rohamosan növekszik
- Média eszközök kreatív használata
- Mindig pontosan valósítsuk meg a média lejátszás teljes életciklusát (pl. *prepare()*)
- Beépített kamera alkalmazás használata
- Multimedia framework
- Ne feledkezzünk el az erőforrások felszabadításáról!

HALADÓ HÁLÓZATI KOMMUNIKÁCIÓS MÓDSZEREK

Hálózati kommunikáció

- NFC
- Bluetooth
- Nearby API
- TCP/IP
- HTTP – Kotlin Cooroutine

NFC

Near Field Communication

Near Field Communication (NFC)

- Rövidtávú vezeték nélküli kommunikációs technológia
- <4cm távolságon belül működik
- NFC tag és mobil telefon közti kis méretű adatátvitel (payload)
- Mobil telefon és mobil telefon közti kis méretű adatátvitel
- NFC Forum által meghatározott formátum: NDEF (NFC Data Exchange Format)

NFC Tag-ek jellemzői

- Írható/olvasható/egyszer írható
- Komplex Tag-ek tartalmazhatnak matematikai műveleteket is és lehet külön kriptográfia hardver egységük authentikáció céljából
- Még bonyolultabb Tag-ek akár saját működési környezettel is rendelkezhetnek és egyszerűbb kód végrehajtására is alkalmasak

NFC olvasás lépései

- Rendszerbeállításokban engedélyezni kell
- Feloldott képernyőzár esetén működik
- Tag Dispatch System a felelős
- Beolvasott NFC Tag automatikus felderítése és a megfelelő alkalamzás indítása (Intent)
- Az alkalmazások *IntentFilter* formájában jelezhetik az őket érdeklő NFC formátumokat
- Érdemes “egyedi” Tag-eket definiálni, amiket csak a saját alkalmazásunk kezel (ne szakadjon meg a kapcsolat mozgatás miatt)

Tag Dispatcher System működése

- NFC Tag parse-olása és a MIME type v. URI felderítése
- MIME type / URI azonosítja a payload (adat) típusát
- MIME type / URI és payload becsomagolása egy Intent-be
- Activity indítása az Intent alapján

NFC írás lépései

- TAG felderítése
- NDEF üzenet megfelelő összeállítása
- NDEF üzenet küldése a felderített TAG-re
- Írás sikerességének ellenőrzése

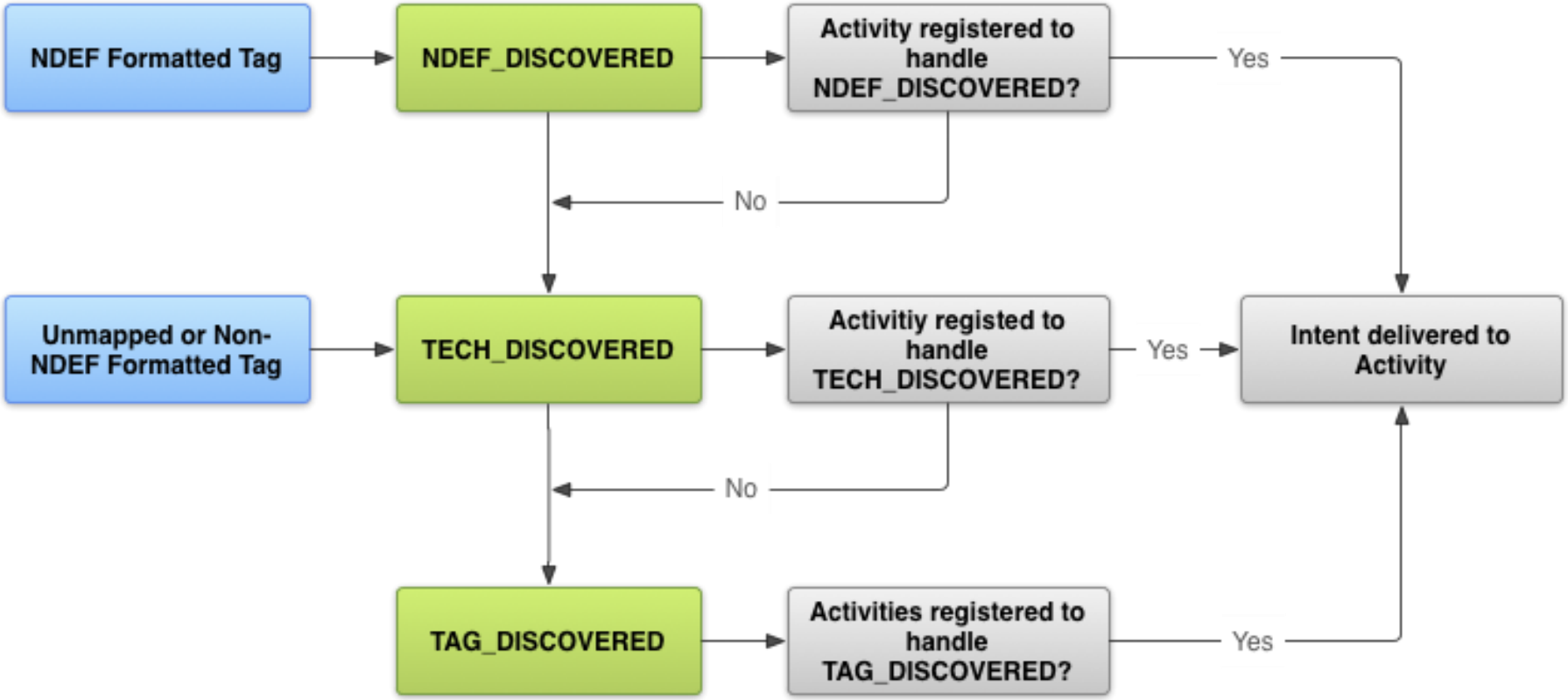
NFC Beam

- NDEF üzenet küldése két Android készülék között
- Nincs szükség készülék felderítésre
- A kapcsolat automatikusan felépül, ha két eszköz hatótávon belülre kerül
- Beépített alkalmazások is támogatják adatcserére (pl.: Browser, Névjegyzék, YouTube stb.)

NDEF adat szerkezete

- *NdefMessage*: NDEF adatot tartalmazza
- *NdefRecord*: egy *NdefMessage* egy vagy több *NdefRecord*-ot tartalmaz
- Minden NdefRecord egy előre meghatározott struktúrát követ
- Az *NdefMessage* első *NdefRecord*-ja határozza meg tipikusan a továbbiak feldolgozását
- Nem NDEF formátum is támogatott Android-on, de nem ajánlott, kezelésük:
  - *android.nfc.tech* csomag osztályai

Tag Dispatch System működése



NFC használat lépései

- Permission
  - > <uses-permission android:name="android.permission.NFC" />
- Minimum Android verzió
  - > <uses-sdk android:minSdkVersion="10"/>
- NFC hardware ellenőrzése (így nem jelenik meg azon eszközök számára, amik nem támogatják az NFC-t)
  - > <uses-feature android:name="android.hardware.nfc" android:required="true" />

Szöveges tartalom detektálása

- Manifest-en belül a megfelelő *<activity>*-elembe:

```
<intent-filter>
<action android:name="android.nfc.action.NDEF_DISCOVERED"/>
<category android:name="android.intent.category.DEFAULT"/>
<data android:mimeType="text/plain" />
</intent-filter>
```

URI detektálása

- Szűrés a http://www.aut.bme.hu/ URI-ra

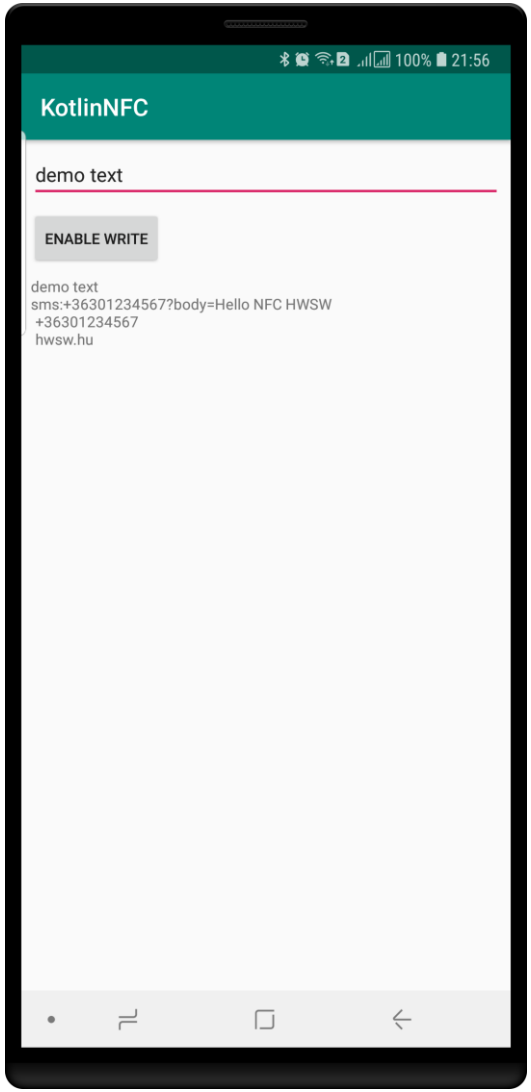
```
<intent-filter>
<action android:name="android.nfc.action.NDEF_DISCOVERED"/>
<category android:name="android.intent.category.DEFAULT"/>
<data android:scheme="http" android:host="www.aut.bme.hu"
android:pathPrefix="/" />
</intent-filter>
```

Beam detektálása

```
<intent-filter>
<action android:name=
  "android.nfc.action.NDEF_DISCOVERED"/>
<category android:name=
  "android.intent.category.DEFAULT"/>
<data android:mimeType=
  "application/vnd.com.example.android.beam"/>
</intent-filter>
```

Első NdefRecord payload-jának kiolvasása

```
override fun onResume() {
    super.onResume()
    if (intent.action == NfcAdapter.ACTION_NDEF_DISCOVERED) {
        val parcelableArray = intent.getParcelableArrayExtra(
          NfcAdapter.EXTRA_NDEF_MESSAGES)
        parcelableArray?.forEach {
            val ndefMsg = it as NdefMessage
            ndefMsg?.records.forEach {
                tvStatus.append("${String(it.payload)}\n")
            }
        }
    }
}
```



Gyakoroljunk!

- Készítsünk NFC író/olvasó alkalmazást
- Szöveges tartalom esetén induljon el automatikusan az Activity

BLUETOOTH KOMMUNIKÁCIÓ

Bluetooth

- Bluetooth készülékek felderítése
- Helyi Bluetooth adapter-ek lekérdezése a párosított eszközökhöz
- RFCOMM csatorna kiépítése
- Kapcsolódás service discovery-n keresztül
- Adattovábbítás készülékek közt
- Több egyidejű kapcsolat kezelése
- Fontosabb osztályok:
  - android.bluetooth csomag

Legfontosabb Bluetooth osztályok

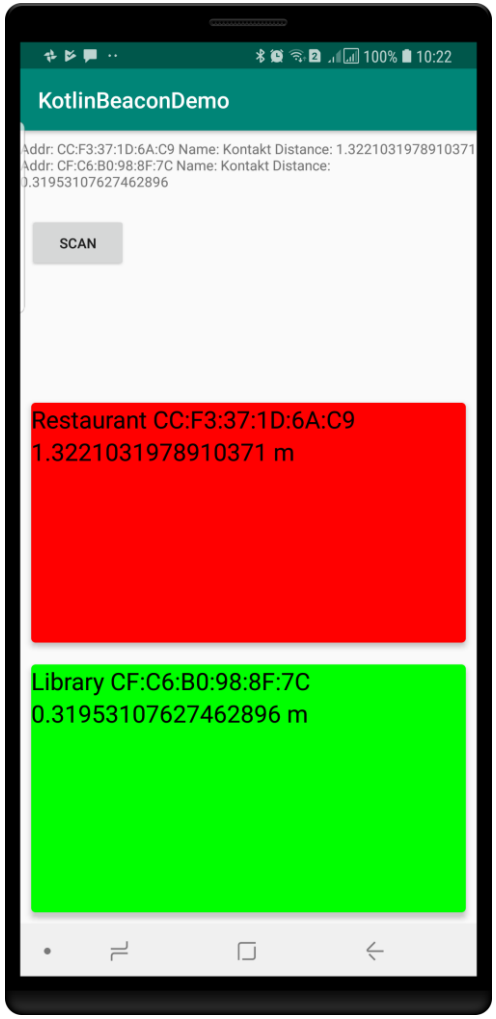
- *BluetoothAdapter*: felderítés, párosítások lekérdezése, BluetoothDevice példányosítása, server socket létrehozás
- *BluetoothDevice*: távoli Bluetooth eszközt jelképezi
- *BluetoothSocket/BluetoothServerSocket*
- *BluetoothClass*: eszköz tulajdonságai (read-only)
- *BluetoothProfile*: profil jellemzői

Bluetooth low energy

- Android 4.3 (API Level 18)
- Bluetooth Low Energy; *central role* és *perhiperarl* APIk
- Felderítés és kommunikáció támogatása (karakterisztikák írása)
- Lényegesen kevesebb energiahasználat (telefon és eszköz oldalon).
- Tipikus eszközök:
  - Közelség érzékelők, szívritmus szenzorok, eü. Kiegészítők, stb.

Hagyományos Bluetooth példák

- Chat:
  - http://mrbrownsandroid.blogspot.com/2017/03/blue-tooth-chat-application-android.html
  - https://github.com/googlesamples/android-BluetoothChat
- Bluetooth osztálykönyvtárak:
  - https://android-arsenal.com/details/1/5821



Gyakoroljunk!

Készítsünk egy alkalmazást, mely Bluetooth Beacon-oket derít fel!

Beacon felderítés 1/3

- Android Beacon Library
  - https://altbeacon.github.io/android-beacon-library/
- Gradle:
  - implementation 'org.altbeacon:android-beacon-library:2+'
- Szükséges engedélyek:
  - ACCESS_COARSE_LOCATION, BLUETOOTH, BLUETOOTH_ADMIN
- BeaconManager:

```
beaconManager = BeaconManager.getInstanceForApplication(this)
```

Beacon felderítés 2/3

- Felderítendő beacon típusok regisztrálása

```
companion object {
    const val IBEACON_LAYOUT = "m:0-3=4c000215,i:4-19,i:20-21,i:22-23,p:24-24"
    const val EDDYSTONE_UID_LAYOUT = BeaconParser.EDDYSTONE_UID_LAYOUT
    const val EDDYSTONE_URL_LAYOUT = BeaconParser.EDDYSTONE_URL_LAYOUT
    const val EDDYSTONE_TLM_LAYOUT = BeaconParser.EDDYSTONE_TLM_LAYOUT
}
```

```
beaconManager.beaconParsers.add(BeaconParser().setBeaconLayout(IBEACON_LAYOUT))
beaconManager.beaconParsers.add(BeaconParser().setBeaconLayout(EDDYSTONE_UID_LAYOUT))
beaconManager.beaconParsers.add(BeaconParser().setBeaconLayout(EDDYSTONE_URL_LAYOUT))
beaconManager.beaconParsers.add(BeaconParser().setBeaconLayout(EDDYSTONE_TLM_LAYOUT))

beaconManager.bind(this)
```

Beacon felderítés 3/3

```
override fun onBeaconServiceConnect() {
    beaconManager.addRangeNotifier { beacons, region ->
        for (beacon in beacons) {
        }
    }
    beaconManager.addMonitorNotifier(object : MonitorNotifier {
        override fun didEnterRegion(region: Region) {
            tvStatus.text = "didEnterRegion ${region.bluetoothAddress} ${region.uniqueId}"
        }
        override fun didExitRegion(region: Region) {
            tvStatus.text = "didExitRegion ${region.bluetoothAddress} ${region.uniqueId}"
        }
        override fun didDetermineStateForRegion(state: Int, region: Region) {
            tvStatus.text =
                    "state switch: $state ${region.bluetoothAddress} ${region.uniqueId}"
        }
    })
    try {
        beaconManager.startMonitoringBeaconsInRegion(Region(
          "myRangingUniqueId", null, null, null))
        beaconManager.startRangingBeaconsInRegion(Region(
          "myRangingUniqueId", null, null, null))
    } catch (e: Exception) {
        tvStatus.text = e.message
        e.printStackTrace()
    }
}
```

TCP/IP KOMMUNIKÁCIÓ

TCP/IP Socket

- Szabványos *Socket* implementáció
- Jól ismert *java.net.Socke*t osztály a kapcsolatok megnyitására
- *java.net.ServerSocket* osztály a bejövő kapcsolatok fogadására
  - Localhoston az alkalmazások egymás közti kommunikációja is megoldható
- InputStream és OutputStream támogatás az adatok olvasására és írására

Socket példa

```
val socket = Socket("192.168.2.112", 8787)
val inputStream = socket.getInputStream()
val isr = InputStreamReader(
    inputStream,
    "UTF-8"
)
val resultBuffer = StringBuilder()
var inChar: Int
while ((inChar = isr.read()) != -1) {
    resultBuffer.append(inChar.toChar())
}
val result = resultBuffer.toString()
// result kezelése
// ...
inputStream.close()
socket.close()
```

UDP communication

- User Datagram Protocol (UDP) for supporting Datagram messages
- Short, quick message handling
- UDP does not ensures that the packet will arrive
- Used typically when packet loss is not as important
- Example usages:
  - Real time multimedia transfer
  - Computer games
  - Etc.

UDP on Androidon

- Use the standard Java classes
- *java.net.DatagramSocket*: socket
- Use it in case of broadcast:

```
DatagramSocket s = DatagramSocket()
s.setBroadcast(true)
```

- *java.net.DatagramPacket*: UDP package object

UDP üzenetek küldése

```
val msg = "UDP Test"
val socket = DatagramSocket()
// In case of broadcast
socket.setBroadcast(true)
//InetAddress localAddr =
//  InetAddress.getByName("192.168.0.110");
val message = msg.toByteArray()
val p = DatagramPacket(
    message,
    msg.length, localAddr, 10100
)
socket.send(p)
```

UDP üzenet fogadása

```
val message = ByteArray(1500)
val packet = DatagramPacket(message, message.size)
val socket = DatagramSocket(10100)
socket.receive(packet)
val msg = String(
    message, 0, packet.getLength()
)
Log.d("MYTAG", "received: $msg")
socket.close()
```

Népszerű osztálykönyvtárak

- KryoNet
  - https://github.com/EsotericSoftware/kryonet
- Near
  - https://github.com/adroitandroid/Near

KryoNet példa

- Adat osztály regisztráció (szerializációhoz)

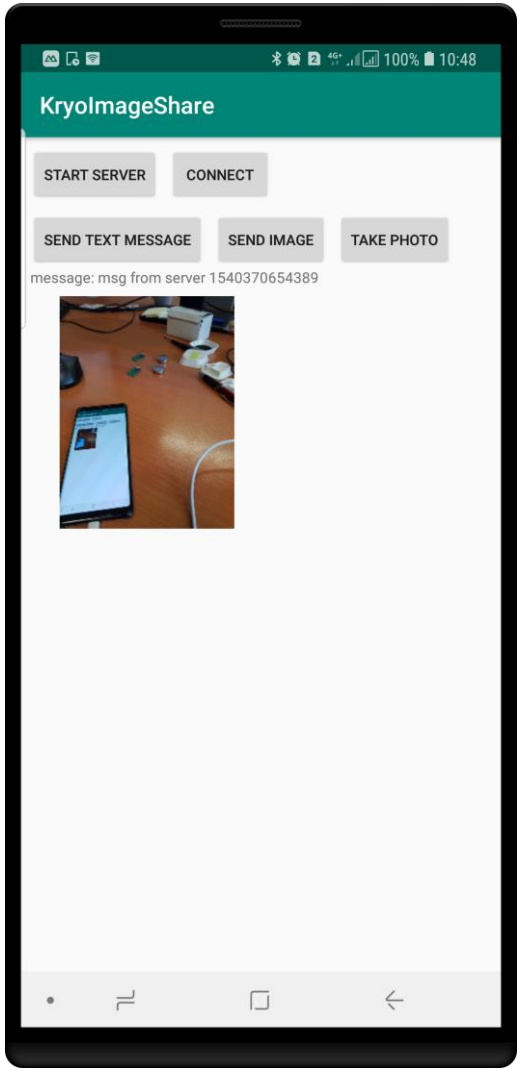
```
class TextMessage {
    var text: String? = null
}

class ImageMessage {
    var image: ByteArray? = null
}


Log.set(Log.LEVEL_DEBUG) // sokat segít ☺

server = Server(131072, 131072) // write és object buffer méretek
val kryo = server.kryo
kryo.register(TextMessage::class.java)
kryo.register(ImageMessage::class.java)
kryo.register(ByteArray::class.java)
```

KryoNet – szerver inicializálás

```
server.addListener(object : Listener() {
    override fun connected(connection: Connection?) {
        super.connected(connection)
    }

    override fun received(connection: Connection, obj: Any) {
        …
    }

    override fun disconnected(connection: Connection?) {
        super.disconnected(connection)
        …
    }
})

server.start()
server.bind(54555, 54777) // tcp és udp portok
```

KryoNet – kliens felderítés és kapcsolódás

```
var client = Client()
client.start()
val kryo = client.kryo
kryo.register(TextMessage::class.java)
kryo.register(ImageMessage::class.java)
kryo.register(ByteArray::class.java)
val address = client.discoverHost(54777, 20000)

client.addListener(object : Listener() {
    override fun connected(connection: Connection?) {
        super.connected(connection)

    }

    override fun received(connection: Connection, obj: Any) {
        handleNetworkMessage(obj)
    }

    override fun disconnected(connection: Connection?) {
        super.disconnected(connection)
        runOnUiThread {
            Toast.makeText(this@MainActivity, "DISCONNECTED", Toast.LENGTH_LONG).show()
        }
    }
})
client.connect(20000, address, 54555, 54777)
```



Gyakoroljunk

Készítsünk egy kép megosztó alkalmazást KryoNet-el

NEARBY API

Nearby API

- Három fő építő elem
- Nearby Messages
  - Egyszerű üzenet küldés és fogadás
  - byte[] küldhető
- Nearby Connections
  - Egyszerű peer-to-peer kapcsolat kiépítés
- Nearby Notifications
  - Weboldal vagy alkalmazás *beacon*-hoz csatolása, értesítések megjelenítése
- További információk:
  - https://developers.google.com/nearby/

Gyakoroljunk!

- Készítsünk egyszerű chat alkalmazást Nearby Messages API-val!

```
implementation 'com.google.android.gms:play-services-nearby:11.6.0'
```

- Fő lépések:
  - ACCESS_FINE_LOCATION engedély szükséges
- Feliratkozás:

```
Nearby.getMessagesClient(this).subscribe(messageListener)
```

- Leiratkozás:

```
Nearby.getMessagesClient(this).unsubscribe(messageListener)
```

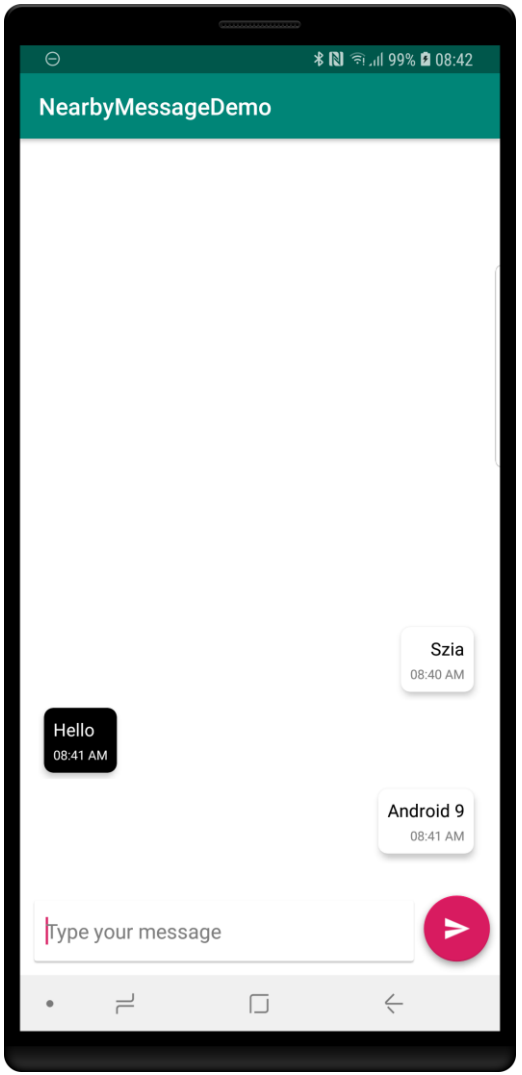
- Üzenet küldése:

```
val nearByMessage = Message(message.toByteArray())
Nearby.getMessagesClient(this).publish(nearByMessage)
```

MessageListener megvalósítása

```
private var messageListener = object : MessageListener() {
    override fun onFound(
      message: com.google.android.gms.nearby.messages.Message) {
        // üzenet mint String: String(message.content)
    }

    override fun onLost(
      message: com.google.android.gms.nearby.messages.Message) {
        Toast.makeText(this@MainActivity,
          "LOST ${String(message.content)}",
          Toast.LENGTH_LONG).show()
    }
}
```

- Próbáljuk ki az Android Chat UI library-t
- https://github.com/timigod/android-chat-ui



Összefoglalás

- Kamera kezelés
- Hanglejátszás és hangrögzítés
- Hangfelismerés, szöveg felolvasás
- Képfelismerés – gépi tanulás (MLKit)
  - QR kód olvasás, arcfelismerés, stb.
- Hálózati kommunikációs lehetőségek
- NFC
- Bluetooth Beacon
- TCP/IP
- Nearby API

Köszönöm a figyelmet!

peter.ekler@aut.bme.hu

AutSoft