

Android

Helymeghatározás, térkép kezelés

Dr. Ekler Péter

peter.ekler@aut.bme.hu



Department of
Automation and
Applied Informatics

Tartalom

- Helymeghatározás
- Helyfüggő szolgáltatások
 - > Geocoding/reverse geocoding
 - > ProximityAlert
 - > Geofences
 - > Activity recognition
- Google Maps képességek
 - > Térkép beállítások
 - > Markerek kezelése
 - > Maps Utility Library

Helymeghatározás

Helymeghatározás

- Android által támogatott módok:
 - > GPS
 - > Hálózat: Android's Network Location Provider
- GPS pontos, de csak kültéren működik és magas az energiaigénye
- Hálózat alapú:
 - > Cella alapú helymeghatározás
 - > WiFi alapú helymeghatározás
 - > Gyors, de gyakran pontatlan
- Egy időben akár mindkét módszert is alkalmazhatjuk

Nem triviális a pozíció meghatározása

- GPS, mobil hálózat és WiFi alapú metódus is adhat egyidőben információt, de figyelembe kell venni a következőket:
 - > Pontosság
 - > Sebesség
- A felhasználó folyamatos mozgásban lehet, sokszor szükség van becslésre
- Előfordulhat, hogy egy frissen kapott pozíció pontatlanabb, mint a 10 másodperccel ezelőtti!



Location engedély beállítása

- Manifest állományba új permission

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

- ACCESS_COARSE_LOCATION: csak hálózat alapú helymeghatározás engedély
- ACCESS_FINE_LOCATION: GPS és hálózat alapú helymeghatározás engedély

Android Location API 1/2

- Callback függvényekben érkezik az információ
- `LocationManager` rendszerszolgáltatás lekérése:

```
var locationManager: LocationManager =  
getSystemService(Context.LOCATION_SERVICE) as LocationManager
```

- Folyamatos frissítés kérése egy
`LocationListener` implementáció átadásával

```
// provider, mintime, mindistance, listener  
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0,  
locationListener)
```

Android Location API 2/2

- requestLocationUpdates() paraméterei:
 - > Provider típus (GPS vagy hálózat alapú)
 - Kétszer is meghívható más-más providerrel (GPS és hálózat)
 - > Minimum idő két frissítés között (0: lehető leggyakrabban)
 - > Minimum távolság két frissítés között (0: lehető leggyakrabban)
 - > LocationListener implementáció
- LocationListener callback függvényei:
 - > Új pozíció:
`onLocationChanged(Location location)`
 - > Provider állapotváltozás:
`onStatusChanged(String provider, int status, Bundle extras)`
 - > Provider használhatóvá vált:
`onProviderEnabled(String provider)`
 - > Provider nem használható
`onProviderDisabled(String provider)`

LocationListener példa

```
var locationListener: LocationListener = object : LocationListener {  
    override fun onLocationChanged(location: Location?) {  
    }  
  
    override fun onStatusChanged(provider: String?,  
        status: Int, extras: Bundle?) {  
    }  
  
    override fun onProviderEnabled(provider: String?) {  
    }  
  
    override fun onProviderDisabled(provider: String?) {  
    }  
}
```

Leiratkozás

- `removeUpdates (...)` függvény hívás minden létrehozott `LocationListener`-re
- Nagyon fontos, hogy **ne felejtsük el!**
- `locationManager.removeUpdates (locationListener)`

FusedLocationProvider

- Alkalmazások képesek megosztani a hely információkat egymással
- Google Play Services függőség:

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

- FusedLocationProviderClient

```
private val fusedLocationClient: FusedLocationProviderClient =  
    LocationServices.getFusedLocationProviderClient(context)
```

Indítás és leállítás

```
fun startLocationMonitoring() {  
    val locationRequest = LocationRequest()  
    locationRequest.interval = 1000  
    locationRequest.fastestInterval = 500  
    locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY  
  
    fusedLocationClient.requestLocationUpdates(locationRequest,  
        locationCallback, Looper.myLooper())  
  
}  
  
fun stopLocationMonitoring() {  
    fusedLocationClient.removeLocationUpdates(locationCallback)  
}  
  
private var locationCallback: LocationCallback = object : LocationCallback() {  
    override fun onLocationResult(locationResult: LocationResult) {  
        super.onLocationResult(locationResult)  
    }  
}
```

Fontos tippek & javaslatok

- Ellenőrizzük, hogy a kapott pozíció jelentősen újabb-e
- Ellenőrizzük a kapott pozíció pontosságát (accuracy)
- Az előző pozícióból és a hozzá tartozó sebességből ellenőrizhető az új pozíció realitása
- Ellenőrizzük melyik provider-től származik az új pozíció
- 60 másodpercnél gyakoribb pozíciókérés sokszor felesleges
- Nem mindig kell GPS és hálózati providert egyszerre használni



Geocoding

- GPS koordináta postacímből
- Internet engedély szükséges

```
val geocoder = Geocoder(this, Locale.ENGLISH)
val streetAddress = "Blaha Lujza tér 1, Budapest"
var locations: List<Address>? = null
geocoder.getFromLocationName(streetAddress, 3)
```

Reverse Geocoding

- Cím GPS koordinátából
- Internet engedély szükséges

```
val location = ...  
val latitude = location.getLatitude()  
val longitude = location.getLongitude()  
val gc = Geocoder(this, Locale.getDefault())  
var addrs: List<Address>? =  
    gc.getFromLocation(latitude, longitude, 3)
```

Helyfüggő szolgáltatások

Geocoding

- GPS koordináta postacímből
- Internet engedély szükséges

```
val geocoder = Geocoder(this, Locale.ENGLISH)
val streetAddress = "Blaha Lujza tér 1, Budapest"
var locations: List<Address>? = null
geocoder.getFromLocationName(streetAddress, 3)
```

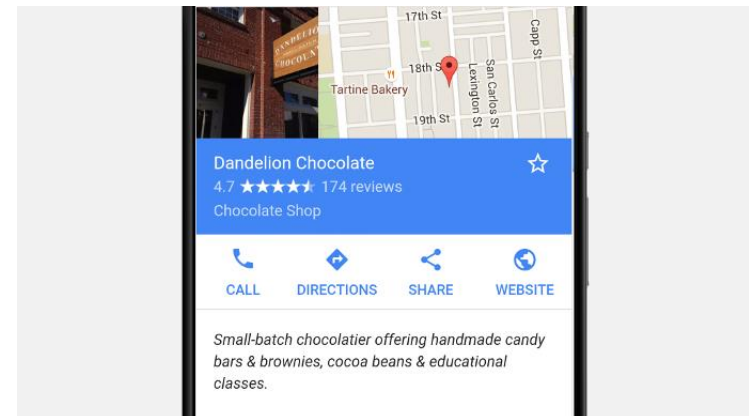
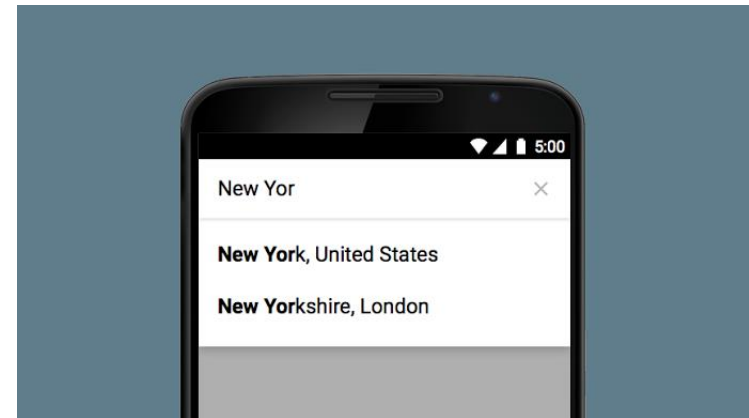
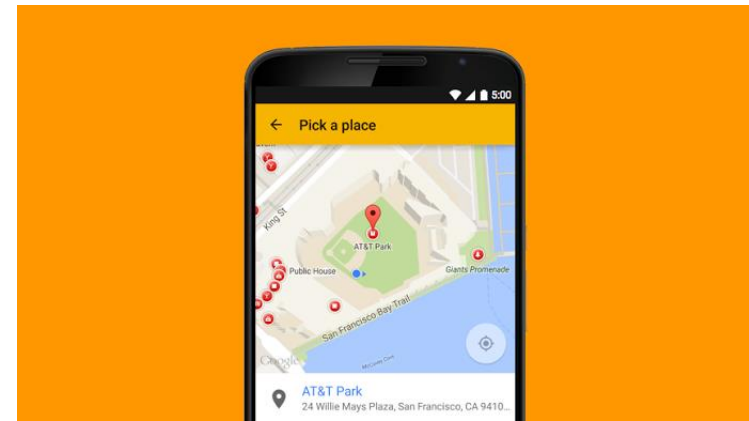
Reverse Geocoding

- Cím GPS koordinátából
- Internet engedély szükséges

```
val location = ...  
val latitude = location.getLatitude()  
val longitude = location.getLongitude()  
val gc = Geocoder(this, Locale.getDefault())  
var addrs: List<Address>? =  
    gc.getFromLocation(latitude, longitude, 3)
```

Places API

- PlacePicker
 - > Hely kiválasztó dialógus
- GeoDataApi
 - > Google hely adatbázisához hozzáférés
- PlaceDetectionApi
 - > Hozzáférés az az aktuális helyhez és jelentések készítése
- AutoComplete
 - > Hely kiegészítő – beviteli mező
- További részletek:
 - > <https://developers.google.com/places/android-api/>



ProximityAlert

- Értesítések egy adott környék megközelítésekor
 - > Koordináta és sugár

```
val intent = Intent(ACTION_PROXIMITY_ALERT)

val pendInt: PendingIntent = PendingIntent.getBroadcast(this, requestcode,
intent, flags)

locationManager.addProximityAlert(lat, long, radius, timeout, pendInt)

...

class ProximityIntentReceiver: BroadcastReceiver() {
    @Override
    override fun onReceive(context: Context, intent: Intent) {
        val key = LocationManager.KEY_PROXIMITY_ENTERING
        val entering = intent.getBooleanExtra(key, false)
        ...
    }
}
```

További GEO API-k

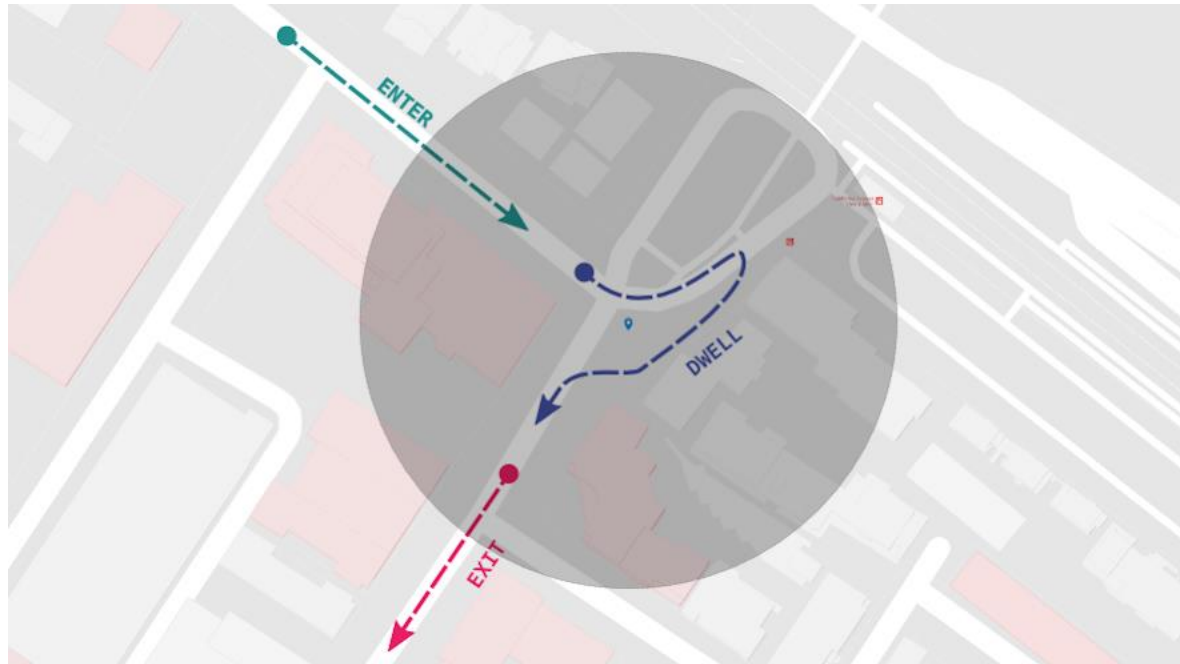
- Google Play Services része
- Új API-k:
 - > Fused location provider (erről már volt szó):
 - <https://developer.android.com/training/location/retrieve-current.html>
 - > Geofencing API
 - > Activity Recognition

Geofencing API

Mostoha Roland (mostoha.roland@autsoft.hu)

Geofencing API

- Földrajzi határok beállítása és feliratkozás belépés vagy elhagyás események bekövetkezésekor



Geofencing API

- Google Play Services része
- Egyszerű, de gazdag API
 - > *Geofence* lista gyors és kötegelt megadása vagy eltávolítása
 - > Több *Geofence* egyidejű kezelése
 - > Riasztások szűrése
 - > Hatékony helymeghatározás *Fused Location Provider* használatával
 - > Alacsony energifogyasztás: a helymeghatározási technológiák dinamikus használata a *Geofence* határától függően

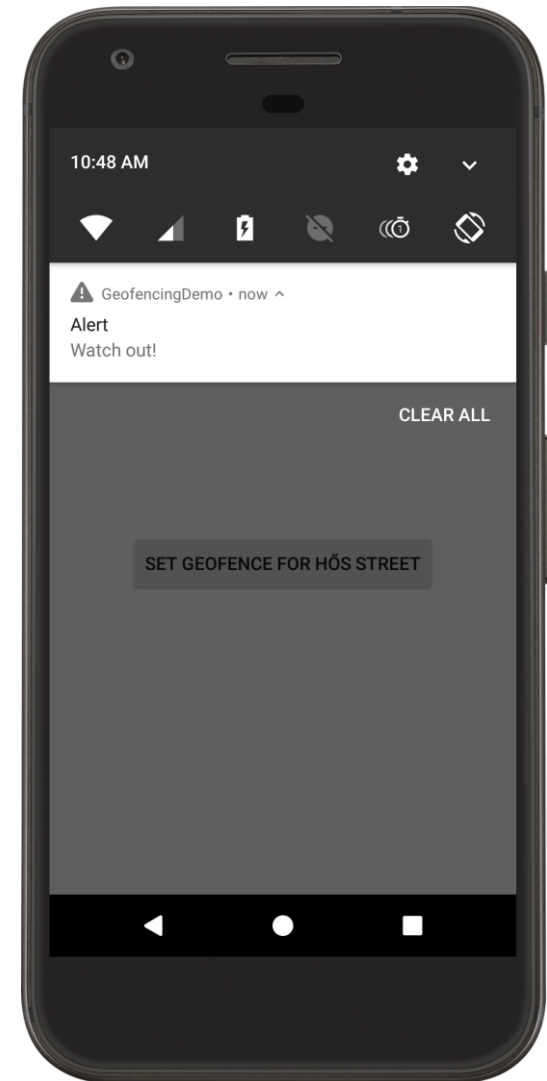
Geofencing API

- Szükséges engedély (dangerous, el kell kérni):
`android.permission.ACCESS_FINE_LOCATION`
- Szükséges függőség:
`com.google.android.gms:play-services-location:X`
- API használata
 - > Kliens inicializálása: *LocationServices.getGeofencingClient(this)*
 - > *IntentService* létrehozása, amely feldolgozza és reagál az *Geofence API* eseményeire
 - > *IntentService* és a számunkra érdekes események regisztrációja a kliensnél
 - > *GeofenceTransition* alapján lekezelni az eseményeket
 - > Szükség esetén kliens lecsatolása és *Geofence* lista törlése

Geofence API demo alkalmazás

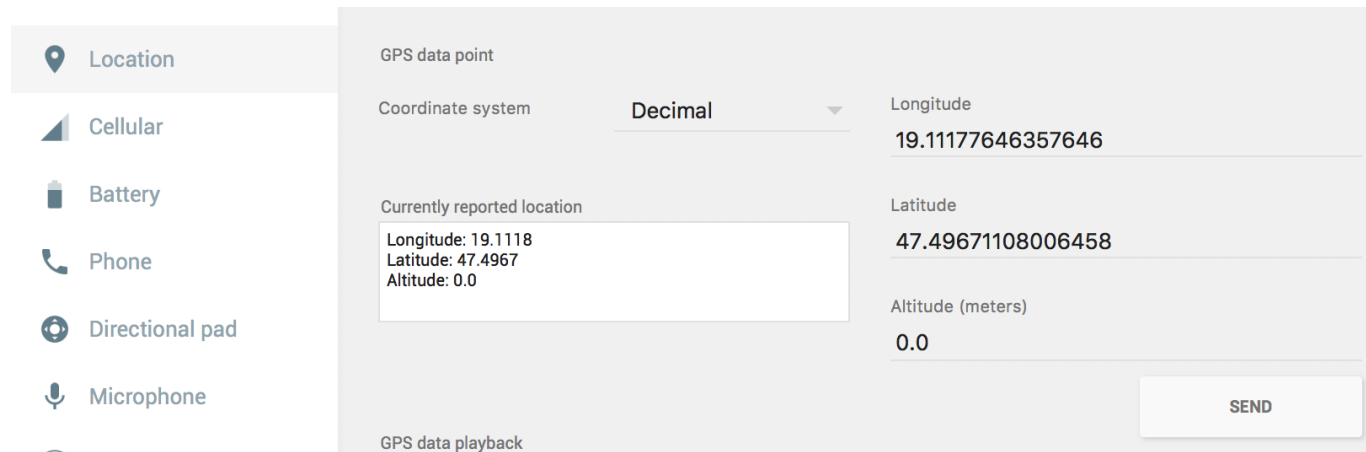
- A hős utcát megközelítve dobjunk egy figyelmeztető értesítést, elhagyva pedig egy megnyugtatót

47.49671, 19.11177



Tippek a teszteléshez

- Google Play Services legújabb verziója szükséges a teszteléshez
- Emulátoron teszteljünk, használjuk a Location tool-t a pozíciók beküldéséhez



Tippek a teszteléshez

- Ne használjunk túl pontos pozíciókat vagy túl kis távolságot
- *Settings- Location mode – High accuracy* módot használjunk, különben az API 1000-es error code-al tér vissza
- Wi-Fi legyen bekapcsolva
- Az esemény kiváltáshoz akár 2-3 perc szükséges lehet

Kotlin kiegészítések

Pair, Triple beépített osztályok Kotlinban:

```
// Hős street (latitude, longitude, circular distance in meters)  
private val region = Triple(47.49671, 19.11177, 5000f)
```

by lazy delegate. Első híváskor inicializálódik, utána ugyanazt a példányt használja:

```
private val geofencePendingIntent: PendingIntent by lazy {  
    val intent = Intent(this, GeofenceIntentService::class.java)  
    PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)  
}
```

Számok tagolása kódban. Növeli az olvashatóságot:

```
.setExpirationDuration(120_000L)
```

Kotlin kiegészítések

Java `||` helyett `or`, javítja az olvashatóságot

(Geofence.*GEOFENCE_TRANSITION_ENTER* or Geofence.*GEOFENCE_TRANSITION_EXIT*)

Függvényhívások ugyanazon az objektumon: `apply`

```
return GeofencingRequest.Builder().apply {  
    setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)  
    addGeofences(geofenceList)  
}.build()
```

Activity Recognition

Mostoha Roland (mostoha.roland@autsoft.hu)

Activity Recognition API

- Google Play Services része
- A felhasználók cselekvéseit detektálhatjuk és reagálhatunk rá
 - > IN_VEHICLE
 - > ON_BICYCLE
 - > ON_FOOT
 - > STILL
 - > WALKING
 - > RUNNING

Activity Recognition API

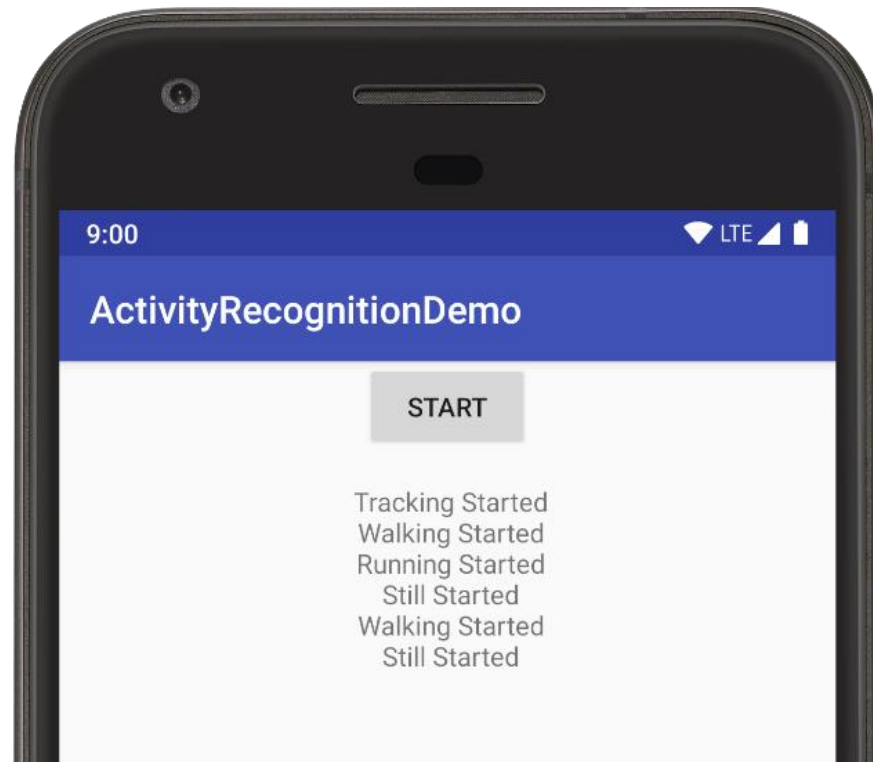
- Feliratkozhatunk a cselekvés megváltozását jelző ENTER és EXIT eseményekre
- Pl. Kalória számláló alkalmazás, amely automatikusan változtatja az elégetett kalória mennyiséget aszerint hogy gyaloglunk, futunk vagy állunk
- Pl. Egy chat alkalmazás, amely vezetés közben letiltja az értesítéseket

Activity Recognition API

- Szükséges engedély:
`com.google.android.gms.permission.ACTIVITY_RECOGNITION`
- Szükséges függőség:
`com.google.android.gms:play-services-location:X`
- API használata
 - > Kliens inicializálása: *ActivityRecognition.getClient(this@MyActivity)*
 - > *IntentService* létrehozása, amely feldolgozza és reagál az *ActivityRecognition* eseményeire
 - > *IntentService* és a számunkra érdekes események regisztrációja a kliensnél
 - > *ActivityType* és *TransitionType* alapján lekezelni az eseményeket
 - > Kliens lecsatolása

Activity Recognition demo alkalmazás

- Activity Log gyűjtése az események alapján



Tippek a teszteléshez

- Valós eszközön teszteljük (az emulátor szenzorai csak ritkán triggerelik a szolgáltatást)
- Tiltsuk le az elforgatást
- Az újabb verziókban már automatikusan történik a mintavételezés, nem tudjuk befolyásolni a gyakoriságot
- Legyünk türelmesek és kitartóak 😊

Kotlin kiegészítések

Előtte:

```
LocalBroadcastManager.getInstance(this).registerReceiver(  
    object : BroadcastReceiver() {  
        override fun onReceive(context: Context, intent: Intent) {  
            ...  
        }  
    },  
    statusIntentFilter)
```

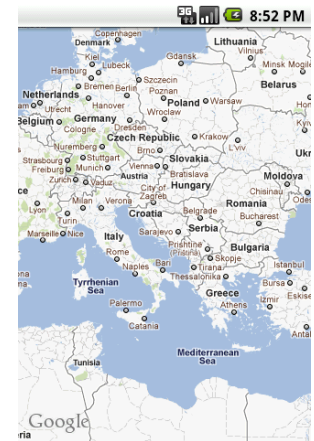
Utána:

```
LocalBroadcastManager.getInstance(this).registerReceiver(statusIntentFilter) {  
    ...  
}
```

Térkép kezelése

Térkép nézet

- Földrajzi pozíciók megjelenítése térkép-szerűen
- Teljes kontroll a megjelenítés felett
 - > Helyszín, nagyítási szint
 - > Térkép, műhold, traffic
- Ráhelyezhetőek overlay-ek
- Megjeleníthetők rajta tetszőleges POI-k

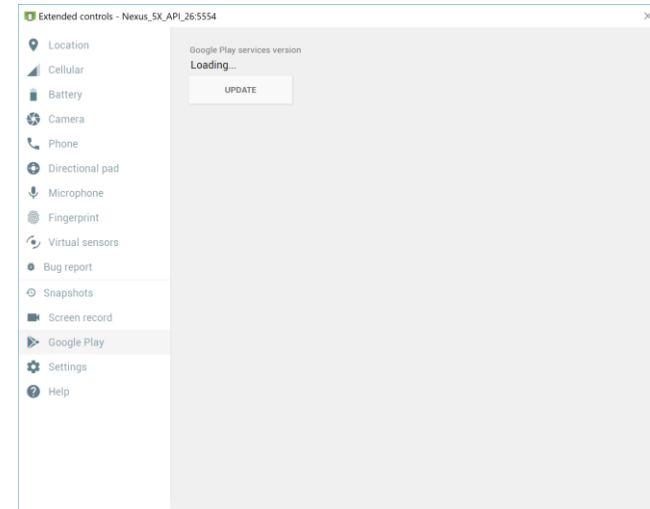


Google Maps API V2

- Google Play Services SDK része
- MapFragment-be helyezett map nézet
 - > Kis kijelzőkön jól használható
 - > Nagy kijelzőkön könnyebben lehet komplex nézeteket létrehozni
- Nincs szükség MapActivity-re, mint V1-ben
- Vektoros térkép csempék:
 - > Gyorsabb megjelenítés
 - > Kevesebb adatforgalom
- Fejlettebb térkép cache
- 3D-s nézet támogatás, perspektívikus megjelenítése

Térkép nézet készítése

- MapFragment alapú térkép megjelenítés
- A térkép letöltése darabonként, on-demand történik, tehát Internet permission-re szükség van
- Manifest engedélyek beállítása
- OpenGL ES jelzése
- Projekt regisztrálása és API key igénylés
 - > Google APIs console
 - > <https://code.google.com/apis/console/>
- Ne importoljuk a teljes play servicest, csak ami kell:
 - > <https://developers.google.com/android/guides/setup>
 - > *Emulátoron érdemes frissíteni a Play Services-t*



MapFragment paraméterei

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    class="com.google.android.gms.maps.SupportMapFragment"
    map:cameraBearing="112.5"
    map:cameraTargetLat="-33.796923"
    map:cameraTargetLng="150.922433"
    map:cameraTilt="30"
    map:cameraZoom="13"
    map:mapType="normal"
    map:uiCompass="false"
    map:uiRotateGestures="true"
    map:uiScrollGestures="false"
    map:uiTiltGestures="true"
    map:uiZoomControls="false"
    map:uiZoomGestures="true"/>
```

Demo – MapFragment felület



MapFragment megjelenítés 1/3

- Google APIs Target kiválasztása projekt létrehozáskor
- Szükséges OpenGL ES osztálykönyvtár használat megjelölése a manifest állomány <application> tag-jében:

```
> <uses-feature android:glEsVersion="0x00020000"  
    android:required="true" />
```

- Szükséges engedélyek:

```
> android.permission.INTERNET  
> android.permission.ACCESS_NETWORK_STATE  
> android.permission.WRITE_EXTERNAL_STORAGE  
> com.google.android.providers.gsf.permission.READ_GSERVICES  
> <permission      android:name=  
    "[package].permission.MAPS_RECEIVE"  
    android:protectionLevel="signature" />
```

- Opcionális: címsor elrejtése nagyobb terület érdekében

```
> <activity android:name=  
    ".HelloGoogleMaps" android:label=  
    "@string/app_name" android:theme=  
    "@android:style/Theme.NoTitleBar">
```

MapFragment megjelenítés 2/3

- Map API kulcs Manifest *<application>* tagjén belül

```
<meta-data android:name=  
    "com.google.android.maps.v2.API_KEY"  
    android:value="API kulcs"/>
```

- MapFragment XML-ben:

```
<?xml version="1.0" encoding="utf-8"?>  
<fragment xmlns:android=  
    "http://schemas.android.com/apk/res/android"  
    android:id="@+id/map"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:name=  
        "com.google.android.gms.maps.SupportMapFragment"/>
```

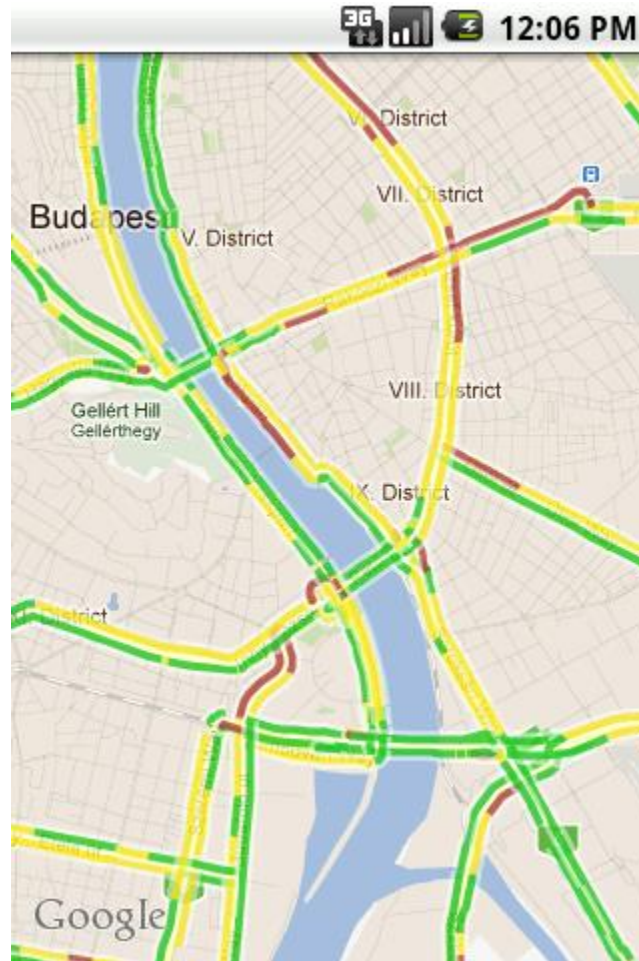
MapFragment megjelenítés 3/3

- `Activity` leszámaztatása elegendő
- `SupportMapFragment` használata, ha szükséges a visszafele kompatibilitás
- `MapFragment` egy `MapView`-ban
- `GoogleMap` elkérése és vezérlése
- Google Maps API AVD létrehozása, Play Services verzió ellenőrzése

Map kezelése - példa

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {  
    private lateinit var myMap: GoogleMap  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_maps)  
        val mapFragment = supportFragmentManager  
            .findFragmentById(R.id.map) as SupportMapFragment  
        mapFragment.getMapAsync(this)  
    }  
  
    override fun onMapReady(googleMap: GoogleMap) {  
        myMap = googleMap  
  
        myMap.isTrafficEnabled = true  
        myMap.mapType = GoogleMap.MAP_TYPE_SATELLITE  
  
        val budapest = LatLng(47.0, 19.0)  
        myMap.addMarker(MarkerOptions()  
            .position(budapest)  
            .title("Marker in Hungary"))  
        myMap.moveCamera(CameraUpdateFactory.newLatLng(budapest))  
    }  
}
```

Demo - Forgalmi nézet példa



Térkép nézet vezérlése

- Érintés esemény kezelése
 - > `GoogleMap.setOnMapClickListener (OnMapClickListener)`
- `UiSettings` objektum:

```
myMap.uiSettings.isRotateGesturesEnabled = true  
myMap.uiSettings.isCompassEnabled = true  
myMap.uiSettings.isZoomControlsEnabled = true
```

MapView

- `View` osztály leszármazottja
- Térkép megjelenítése
- Konténer szerep `GoogleMap` objektumon keresztül
- `Activity` életciklus függvényeit továbbítani kell a `MapView` fele
- Egy *Activity* egyszerre jelenleg leginkább csak egy *MapView*-t támogat

Marker megjelenítése 1/2

```
val hungary = LatLng(47.0, 19.0)
myMap.addMarker(MarkerOptions().
    position(hungary).
    title("Marker in Hungary"))
myMap.moveCamera(CameraUpdateFactory.newLatLng(hungary))
```

Marker megjelenítése 2/2

```
val markerHU = myMap.addMarker(  
    MarkerOptions()  
        .position(hungary)  
        .title("Magyarország")  
        .snippet("Lakosság: 9.700.000")  
        .icon(BitmapDescriptorFactory.fromResource(  
            R.mipmap.ic_launcher_round)) )  
markerHU.isDraggable = true
```

Map esemény kezelés

```
mMap.setOnMapClickListener {  
    val markerHU = mMap.addMarker(  
        MarkerOptions()  
            .position(it)  
            .title("Hello")  
            .snippet("Lakosság: 9.700.000"))  
    markerHU.isDraggable = true  
  
    mMap.animateCamera(CameraUpdateFactory.newLatLng(it))  
}
```

Marker kezelés

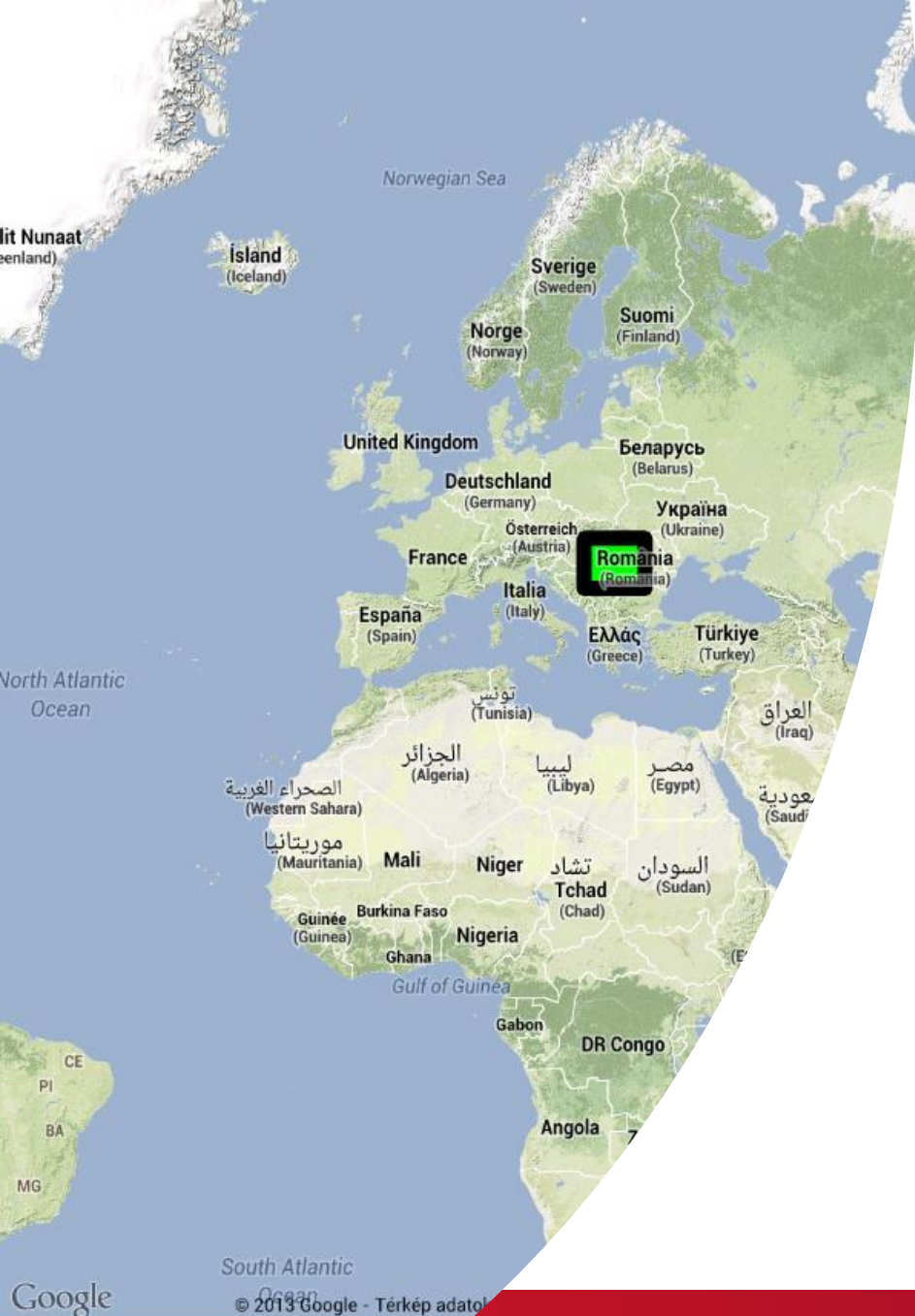
- InfoWindow:
 - > Testreszabható InfoWindow felület
 - > InfoWindowAdapter
 - > Megjelenítés/eltüntetés programozottan
 - > Eseménykezelés: OnInfoWindowClickListener
- Marker eseménykezelők:
 - > OnMarkerClickListener
 - > OnMarkerDragListener
 - > Stb.

Rajzolás térképre

- Támogatott elemek:
 - > Polygon
 - > Polyline
 - > Circle
- Testre szabható megjelenítés
 - > Vonal szín
 - > Kitöltés szín
 - > Z-index
 - > Láthatóság
 - > Stb.

Téglalap rajzolása térképre

```
val polyRect: PolygonOptions = PolygonOptions().add(  
    LatLng(44.0, 19.0),  
    LatLng(44.0, 26.0),  
    LatLng(48.0, 26.0),  
    LatLng(48.0, 19.0))  
val polygon: Polygon = myMap.addPolygon(polyRect)  
polygon.fillColor = Color.GREEN
```

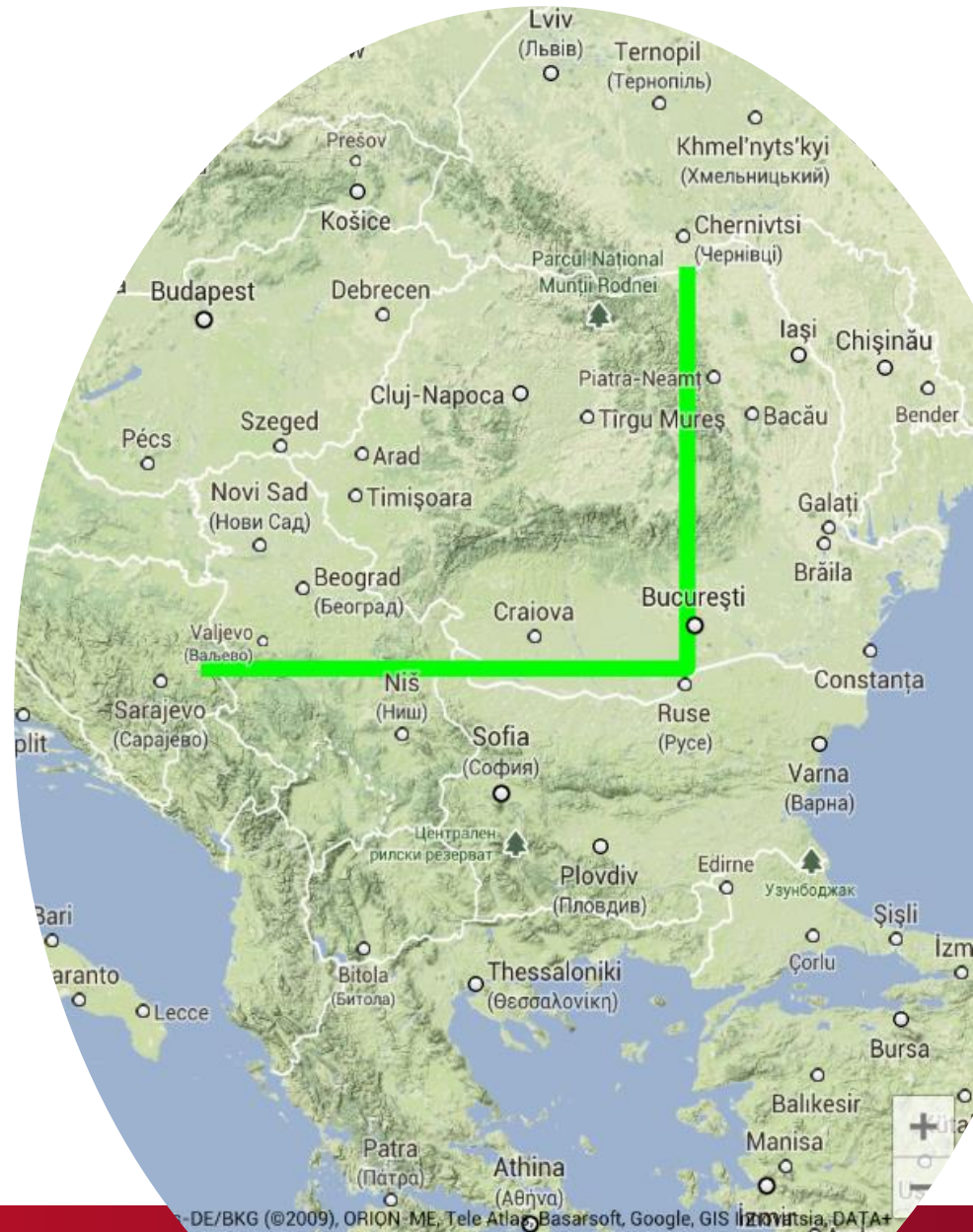



Demo – Téglalap rajzolás

Vonal rajzolás példa

```
val polylineOpts = PolylineOptions().add(  
    LatLng(44.0, 19.0),  
    LatLng(44.0, 26.0),  
    LatLng(48.0, 26.0))  
val polyline = myMap.addPolyline(polylineOpts)  
  
polyline.color = Color.GREEN
```

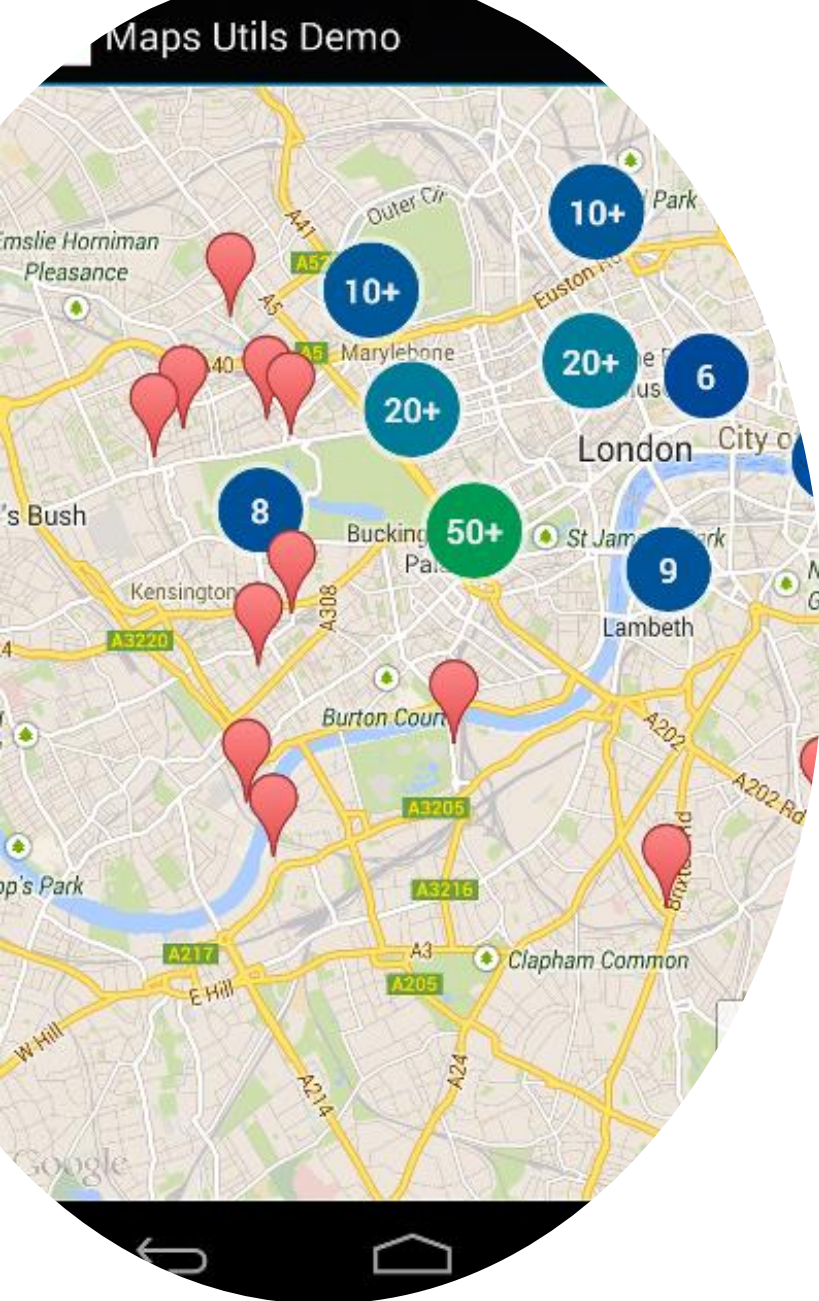
Demo – Vonal rajzolás



További térkép funkciók

- `moveCamera ()`: kamera mozgatása
- `animateCamera ()`: animált mozgatás
- `CameraPosition`: kamera állítása factory-val
 - > Target
 - > Zoom
 - > Bearing: orientáció
 - > Tilt: döntés

```
val cameraPosition = CameraPosition.Builder()  
    .target(LatLng(47.0, 19.0))  
    .zoom(17f)  
    .bearing(90f)  
    .tilt(30f)  
    .build()  
myMap.animateCamera(CameraUpdateFactory.newCameraPosition(  
    cameraPosition))
```

<https://developers.google.com/maps/documentation/android-api/utility/?hl=en>

Map Utility Library

Kérdések

