Mobile location-based apps

University of Tartu
Jaak Laineste, Oct. 2018
Location-based service overview

Part 1
Jaak Laineste

- GIS/LBS experience
  - 23 years in GIS/mapping, 18 years in LBS
  - Regio/Reach-U, Mobile operator LBS globally
  - Nutiteq: 2006-2016
  - CARTO 2016-18

- Nutiteq/CARTO and mobile development
  - CARTO Maps SDK - native maps API
  - CARTO Builder and Engine - Location Intelligence platform
Location-Based Services

1. Location-based: 80% of data
2. Service (or mobile application)
3. Mobile technologies (phones, networks)
4. Mobile positioning

- LBS is universal technology
  - Can be aspect of any app type: info, game, tool, social...
Android app with map

Part 2
What can be done

- **Display map**
  - MapView API - interactive map: Google Maps, 2D/2.5D, online
  - Add clickable points to map (annotations, clusters)
  - Add overlays, rasters - tiles or bitmaps
  - Show user GPS location

- **Geocoding**
  - android.location.Geocoder API, online

- **Reverse-geocoding**
  - android.location.Geocoder
  - getFromLocation()

- **Calculate distance** between objects
  - Location.distanceTo() method

- **Convert coordinates**
  - Projected to/from LatLong/WGS84/GPS

- **Point in Area** (and other georelations)
  - Spatial-enabled GIS engine: PostGIS in server or Spatialite in phone

- **Routing** - find optimal path in graph
  - Google online API, other on-line routing engines
  - Spatialite routing, in small area
  - Graphhopper offline - open source

- **Navigation**
  - Turn by turn - HERE, TomTom, Sygic

- **Offline**
  - Maps, Routing, Geocoding, Data - CARTO SDK
Practical coding on Android

1. Adding MapView to app
2. Add some objects to map
   - Single point
   - GeoJSON polygon
3. Get and show user location
4. Show own base map overlay
Add Google Maps SDK to your project

1. Create new project, with Google Maps SDK view
   - Default settings

2. Create Google API key as instructed in Studio/web

3. Consult
   https://developers.google.com/maps/documentation/android-sdk/intro
Data models

- **Raster**
  - PNG for maps, lossless
  - JPG for aerials
  - GeoTIFF, coverages
  - Key parameters: bitmap resolution, size, channels

- **Vector**
  - Base objects: **points, lines, polygons**
  - Collections: multi-polygon, multi-line, multi-point
  - **Attributes** (fields): text/boolean/numeric/binary etc
  - **Texts** on map - labels from attributes
  - **Layer** - same as „table“ in DB
  - Special cases: topological models, graphs, 3D vectors
GIS Layers

- Administrative areas
- Rights and interests
- Site addresses, uses, and restrictions
- Ownership and tax parcels
- Parcel framework
- Orthophoto
Other data models

• Elevation models
  – DEM - Digital Elevation Model
  – TIN - triangulated irregular network

• 3D worlds
  – Collada, X3D, KMZ, 3D vector tiles
  – CityGML, IndoorGML

• Point clouds
  – Lidar - laser measurements
Hillshade from DEM
TIN
Point cloud
Tallinn as Point Cloud

https://cafa3d.com/3dpoc/
3D model
Heatmap - density of points
Point Clusters
GIS layers in a map

- **One base layer**
  - Background map: Google, OpenStreetMap etc
  - Raster or vector-based. Or automatic.
- **Overlay layer(s)**
  - Points of Interest, markers
  - GPS location - dynamic info
  - Lines, Polygons, Points
  - Clustered points
  - 3D objects
Add pin to map

// Add a marker in Tartu and move the camera
 LatLng tartu = new LatLng(26.729038, 58.377983);
 mMap.addMarker(new MarkerOptions().position(tartu).title("Marker in Tartu");

mMap.moveCamera(CameraUpdateFactory.newLatLng(tartu));

Coordinate systems

- Geographical - spherical
  - Units: Latitude and Longitude
  - Based on an ellipsoid, e.g. WGS-84
  - Datums, also WGS84 for GPS
  - DMS for display, decimal degrees for programming

- Projected - cartesian
  - Units: usually meters
  - 1000+ named projections, mostly for local regions
  - Reduce distortions: keep angles, distances, areas equal
Geographical coordinate space

Longitude

Latitude
Cartesian (projected) coordinate space

Origin (0, 0)
Mercator
Lambert conformal conic projection

Different projections

• Extreme examples
  – http://www.jasondavies.com/maps/

• In real life
  – “GPS system”, Latitude/Longitude, WGS84 (EPSG:4326)
  – Web map, Spherical Mercator (EPSG:3857)
  – Mercator (real one)
  – UTM with zones
  – Robinson - equal-area world map
  – Estonia: Lambert Conformal Conic, L-EST (EPSG:3301)
From where to get my Lat/Lon?

- **Web tools**
  - [www.latlong.net](http://www.latlong.net)
  - [geojson.io](http://geojson.io)
- **Geocode APIs**
  - Google Geocoder
- **From GPS**
  - Location API
- **GIS databases**
  - LatLon coordinates
  - OpenStreetMap
- **From other coordinate systems**
  - **Proj.4** library (native, JS, Java etc versions)
  - Estonian Maaamet Open Data
Lon,Lat or Lat,Lon?

<table>
<thead>
<tr>
<th>lon, lat</th>
<th>lat, lon</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC: -74.0059, 40.7127</td>
<td>NYC: 40.7127, -74.0059</td>
</tr>
</tbody>
</table>

**formats**
- GeoJSON
- KML
- Shapefile
- WKT
- WKB
- geobuf

**javascript apis**
- OpenLayers
- d3
- ArcGIS API for JavaScript
- Mapbox GL JS

**mobile apis**
- Tangram ES
- Nutiteq SDK
- CARTO Mobile SDK

**misc**
- OSRM
- Redis

**GeoRSS**
- Encoded Polylines (Google)

**Leaflet**
- Google Maps API
  - CARTO API

**Google Maps iOS/Android**
- Apple MapKit
- Mapbox GL native

Sources: [http://www.macwright.org/lonlat/](http://www.macwright.org/lonlat/); jaak@carto.com
Lon,Lat or Lat,Lon?

Tartu: 58.378940, 26.729370

Tartu: 26.729370, 58.378940
Short and long projection description

**EPSG** codes: 4-5 digit numbers

- EPSG:4326 - WGS84
- EPSG:3301 - Estonian system
- EPSG:3857 - „Google web Mercator”  
  
  - A.k.a EPSG:900913

```xml
<EPSG:3301>
  PROJCS["Estonian Coordinate System of 1997", GEOGCS["EST97", DATUM["Estonia_1997", SPHEROID["GRS 1980",6378137,298.257222101, AUTHORITY["EPSG","7019"], TOWGS84[0,0,0,0,0,0,0], AUTHORITY["EPSG","6180"], PRIMEM["Greenwich",0, AUTHORITY["EPSG","8901"]], UNIT["degree",0.01745329251994328, AUTHORITY["EPSG","9122"]], AUTHORITY["EPSG","4180"]], UNIT["metre",1, AUTHORITY["EPSG","9001"]], PROJECTION["Lambert_Conformal_Conic_2SP"], PARAMETER["standard_parallel_1",59.33333333333334], PARAMETER["standard_parallel_2",58], PARAMETER["latitude_of_origin",57.51755393055556], PARAMETER["central_meridian",24], PARAMETER["false_easting",500000], PARAMETER["false_northing",6375000], AUTHORITY["EPSG","3301"], AXIS["Y",EAST], AXIS["X",NORTH]]
</EPSG:3301>
```
Map types and Camera

```java
mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);

// Set view
LatLng tartu = new LatLng(58.377983, 26.729038)
mMap.moveCamera(CameraUpdateFactory.zoomTo(12));
mMap.moveCamera(CameraUpdateFactory.newLatLng(tartu));
```
Web map tiling system (OSM)
Add Tile Overlay

- See sample code from Google Maps API guides
- URL format for `UrlTileProvider`
- [https://pastebin.com/6DkdcL4U](https://pastebin.com/6DkdcL4U)

```java
String s = String.format("http://tiles.maaamet.ee/tm/tms/1.0.0/foto@GMC/%d/%d/%d.png", zoom, x, (1 << zoom) - 1 - y);
```

- Note the “flipped” tile y coordinate here
User location on map

- Android Location API
  - GNSS (GPS, GLONASS, Galileo) + wifi + mobile network cell
- Alternative positioning methods
  - External GPS
  - Mobile Positioning
  - Indoor positioning
    - Beacons
    - Wifi
    - Bluetooth
    - Magnetic variations
Location API in Android

- Location Provider
  - Selected automatically based on set requirements: Cell-id, WiFi or GNSS
- Features
  - Listen for updates - most common
  - Last known location (cached) - not suggested
  - Proximity alerts - not guaranteed
- Best strategy depends on app
Using Android location API code

1. Define permission in AndroidManifest.xml
   ```xml
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
   ```

2. Ask runtime permissions
   - Needed for Android 6.0+
   - Use Google sample code (PermissionUtils etc)

3. Different methods
   1. `MyLocationButton`
   2. Last known location
   3. `LocationCallback` with Fused Location API
public class MyLocationDemoActivity extends FragmentActivity
    implements OnMyLocationButtonClickListener,
    OnMyLocationClickListener,
    OnMapReadyCallback {

    ... 

    mMap.setMyLocationEnabled(true);
    mMap.setOnMyLocationButtonClickListener(this);
    mMap.setOnMyLocationClickListener(this);
    ...

    @Override
    public void onMyLocationClick(@NonNull Location location) {
        Toast.makeText(this, "Current location: \n" + location, Toast.LENGTH_LONG).show();
    }

    @Override
    public boolean onMyLocationButtonClick() {
        Toast.makeText(this, "MyLocation button clicked", Toast.LENGTH_SHORT).show();
        // Return false so that we don’t consume the event and the default behavior still occurs
        // (the camera animates to the user’s current position).
        return false;
    }
## Short glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Source</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find device location</td>
<td>GPS, GLONASS, Galileo, Wifi, GSM sensors</td>
<td>Lat, Lon, Accuracy (horiz., vertical), speed, direction</td>
</tr>
<tr>
<td>Geocoding</td>
<td>String: address, place or POI name</td>
<td>Point (Lat, Lon), hit rate estimate</td>
</tr>
<tr>
<td>Reverse-Geocoding</td>
<td>Point (Lat, Lon)</td>
<td>Place description: Address, POI etc</td>
</tr>
<tr>
<td>Point in Polygon</td>
<td>Point, Polygon[]</td>
<td>Polygon where point is</td>
</tr>
<tr>
<td>Routing, Directions</td>
<td>2 (or more) Points, Route mode (walk, car etc)</td>
<td>PolyLine, ETA, Array of turn instructions</td>
</tr>
<tr>
<td>Road matching</td>
<td>GPS Point, road network</td>
<td>Point projected to nearest road</td>
</tr>
<tr>
<td>Transform coordinates</td>
<td>Point, Line, Polygon or raster in Projection X</td>
<td>Same in Projection Y</td>
</tr>
<tr>
<td>Heatmap</td>
<td>Points</td>
<td>Raster with point density</td>
</tr>
<tr>
<td>Point Clustering</td>
<td>Points</td>
<td>Smaller # of points</td>
</tr>
<tr>
<td>Geofencing</td>
<td>Area(s), Dynamic Point(s)</td>
<td>Events for area enter/leave</td>
</tr>
</tbody>
</table>
Reverse-geocode my location

```java
Geocoder geocoder = new Geocoder(this, Locale.ENGLISH);

try {
    List<Address> addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(), 1);

    if (addresses.size() > 0) {
        Address fetchedAddress = addresses.get(0);
        StringBuilder strAddress = new StringBuilder();
        for (int i = 0; i <= fetchedAddress.getMaxAddressLineIndex(); i++) {
            strAddress.append(fetchedAddress.getAddressLine(i)).append(" ");
        }

        Toast.makeText(this, "Current location:
            n" + strAddress.toString(), Toast.LENGTH_LONG).show();
    } else {
        Toast.makeText(this, "Current location (No address):
            n" + location, Toast.LENGTH_LONG).show();
    }

} catch (IOException e) {
    e.printStackTrace();
    Toast.makeText(this, "Current location (Ex):
        n" + location, Toast.LENGTH_LONG).show();
}
```
Indoor LBS

- 90% time people are indoors
  - Retail, Safety
- Indoor maps
  - Google Maps - automatically in SDK
  - HERE, Micello
  - 3D technologies R&D Tartu: Wayfinder
- Indoor Positioning
  - IndoorAtlas, Indoor.rs etc
- Technologies
  - iBeacon positioning - BT LE
  - Standards: IndoorGML, OSM Simple 3D, i-locate.eu
Map data models
Adding online (or offline) data to map
Common GIS vector formats

- **GeoJSON**
  - Standard JSON, works with any JSON parser, easy to read
  - Nice web tool: [http://geojson.io](http://geojson.io), Github support
  - No style info, always LatLong coordinates

- **Shapefile (ESRI)** - most common in GIS data
  - 4+ files per layer: .shp, .dbf etc
  - One geometry type per layer allowed
  - No style information, pure geometry

- **KML (Google)**
  - Usually vector data, includes styles
  - Special data types: 3D data (Collada), linked data, visual coverages, raster tiles
  - Can be KMZ - zipped file

- **Other formats**
  - **GeoPackage**: SQLite binary format, vector and raster
  - TopoJSON - JSON with topology model, more compressed and complex
  - SpatiaLite: vector and raster data, SQLite. Any projection, no styles.
  - Text files (CSV) with coordinates or addresses
  - Every commercial GIS has own format(s)
Spatial databases: basics

- Special column data types(s):
  - Geometry, Point, Polygon ...

- Geographical indexing
  - Usually R-Tree, based on object bounds

- Geographical functions:
  - Manipulations, relations, queries etc

- Metadata table:
  - Defines coordinate system, data type for every Geometry column
Geometry primitives in WKT (2D)

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>POINT (30 10)</td>
</tr>
<tr>
<td>LineString</td>
<td>LINESTRING (30 10, 10 30, 40 40)</td>
</tr>
<tr>
<td>Polygon</td>
<td>POLYGON ((30 10, 10 20, 20 40, 40 40, 30 10))</td>
</tr>
<tr>
<td></td>
<td>POLYGON ((35 10, 10 20, 15 40, 45 45, 35 10), (20 30, 35 35, 30 20, 20 30))</td>
</tr>
</tbody>
</table>
## Multipart geometries in WKT (2D)

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultiPoint</td>
<td>MULTIPOLYGON (((10 40), (40 30), (20 20), (30 10)))</td>
</tr>
<tr>
<td></td>
<td>MULTIPOLYGON (10 40, 40 30, 20 20, 30 10)</td>
</tr>
<tr>
<td>MultiLineString</td>
<td>MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))</td>
</tr>
<tr>
<td>MultiPolygon</td>
<td>MULTIPOLYGON (((30 20, 10 40, 45 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))</td>
</tr>
<tr>
<td></td>
<td>MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 45 20, 30 5, 10 10, 10 30, 20 35), (30 20, 20 25, 20 15, 30 20)))</td>
</tr>
</tbody>
</table>
PostGIS sample: spatial query

• Coordinate systems
  – Distances in meters vs degrees
  – Different projections do not fit well
• “Big data” challenge
  – use two-phase filters to ensure Spatial Indexes
• Sample: http://bit.ly/utlbs2018

```sql
select a.* from osm_amenities_eur a, osm_roads r
where r.name like 'Tallinn&Tartu'
  and a.type='fuel'
  and st_dwithin(r.the_geom,a.the_geom,0.05)
  and st_distance_sphere(r.the_geom,a.the_geom)<1000
```
PostGIS: spatial query
OpenStreetmap
As key data source
OpenStreetMap (OSM)

- Free and open data
  - Vector data in 2D
  - Streets, roads, buildings, amenities etc
- A lot of services
  - Map images (tiles), geocoders, routers etc
  - Special views: opencyclemap, openpistemap etc
- Everyone can improve the map
  - www.openstreetmap.org
  - www.maakaart.ee - in Estonian
ID – web-based OSM editor
OSM Latest Changes
Explore latest changes on OpenStreetMap

→ CharlieHotelRomeo  Oct 9th 2018, 7:06:27 pm
  Connect paths near Tiigi street (for pathfinding). Add some pedestrian

→ CharlieHotelRomeo  Oct 9th 2018, 6:58:39 pm
  Update area around and Naerumaa kindergarten itself.

→ Pikse  Oct 8th 2018, 6:57:17 pm
  Nimeparandusi

→ jaakl  Oct 8th 2018, 10:24:28 am
  Lisatud restoran, kohapealne teave

→ JeanFred  Oct 5th 2018, 10:44:34 am
  Add bicycle parking capacities

→ JeanFred  Oct 5th 2018, 10:44:32 am
  Add road surfaces

→ CharlieHotelRomeo  Sep 20th 2018, 10:06:51 pm
  Add a parking aisle. Reclassify Samelini kohvik so it would show up on the

→ mueschel  Sep 9th 2018, 2:16:09 pm
  One wikipedia is sufficient, articles are linked

→ mueschel  Sep 9th 2018, 1:50:16 pm
  8 website links for one restaurant are just too much. One is sufficient.

→ CharlieHotelRomeo  Sep 6th 2018, 11:15:20 am
  Add Wikipedia links to Tartu town hall square.

→ CharlieHotelRomeo  Sep 6th 2018, 11:08:40 am
  Add some tree rows in city center.

→ CharlieHotelRomeo  Sep 6th 2018, 11:02:09 am
  Specify and add bicycle parkings in the city center.
OSM advantages

- Free and open to use
  - No advertising, restrictions
- Vector data access
  - Custom styles for mapping
  - Own filters of data on map (layers)
  - Interactive data overlays (POI layers)
  - Advanced services: routing, search, analysis, ...
- Fast and easy updates
  - Find error - go fix it yourself!
  - Note: follow community guidelines

- Maps as a Service
  - Cleaned data, easy to use
  - Mapbox - SDKs, tiles, geocoding etc
  - OpenMapTiles - tiles

- Data for advanced users
  - download.geofabrik.de - per country
  - Overpass API - complex queries
  - planet.openstreetmap.org - raw data, load to own Postgres GIS
  - https://registry.opendata.aws/osm/- SQL via AWS Athena, raw data
Other sources for map data

- **Global vendors**
  - Vector: HERE, TomTom, AND
  - Aerial/Satellite: DigitalGlobe, Planet Labs

- **Local vendors**
  - Maa-amet - from 2018 free open data
  - Estonia: Regio, Latvia: Jana Seta etc
  - In almost every country, usually detailed maps

- **Specifics**
  - Technically quite flexible
  - Often quite expensive
Useful geo tools
Free GIS tools

- Desktop tools - mapping, analysis, processing
  - QGIS - freeware
  - Google Earth Pro - view/create KML
  - Trimble SketchUp - create 3D models, paid plans

- Databases
  - PostGIS - Postgres add-on
  - GeoPackage and Spatialite - SQLite geo extension (not enabled on Android/iOS)
  - Big data Geo data extensions (no free): AWS Athena, BigQuery,
Free GIS tools for developers

- **Mobile SDKs**
  - Mapbox, Google Maps SDK, CARTO

- **Web SDKs**
  - Google Maps, Mapbox, LeafletJS, OpenLayers, CesiumJS (3D maps)

- **PostGIS**
  - Postgre SQL server extension
  - Raster, Topology features

- **Libraries**
  - JTS - Java Topology Suite, ports in C (GEOS) etc. Different geo operations
  - Proj.4 - C-based, has few ports. Proj4J for Java - projection conversions
  - GDAL/OGR - raster and vector library, command-line tools
  - Mapbox-GL, Mapbox Vector Tiles tools

- **Server platforms/frameworks**
  - GeoDjango (Python), GeoNode (Node.js), GeoTools (Java)
Some GeoWebServices SaaS

- **Hosted GIS**
  - CARTO - Location Intelligence
  - Mapbox - Google Maps API clone
  - ESRI ArcGIS - the most popular GIS vendor
  - GISCloud.com (ESRI clone)
- **OpenStreetMap.org**
  - Only for community-created data
  - Check content requirements: verifyable objects etc
- **Github**
  - GeoJSON and TopoJSON support
  - geojson.io - GeoJSON editor
- **Google Firebase**
  - Backend-as-service for mobile and web apps
  - GeoFire - location tracking, geofencing samples
Some cool future geo things

- Cesium JS - 3D in web browser
- D3 JS - cool rendering
- OmniSci (MapD) - GPU-based superfast database
- Augmented Reality maps?
  - https://drive.google.com/open?id=0B3Rm1qU6KnySN0IEXzAtOFZQeU0
Summary: what you can do now

- **Display map**
  - MapView API - interactive map: Google Maps, 2D/2.5D, online
  - Add clickable points to map (annotations, clusters)
  - Add overlays, rasters - tiles or bitmaps
  - Show user GPS location

- **Geocoding**
  - android.location.Geocoder API, online

- **Reverse-geocoding**
  - android.location.Geocoder getFromLocation(), online

- **Calculate distance** between objects
  - Location.distanceTo() method

- **Convert coordinates**
  - Projected to/from LatLong/WGS84/GPS

- **Point in Area** (and other georelations)
  - Spatial-enabled GIS engine: PostGIS in server or Spatialite in phone

- **Routing** - find optimal path in graph
  - Google online API,
  - Other on-line routing engines

- **Navigation**
  - Turn by turn - HERE, TomTom, Sygic

- **Offline**
  - Maps, Routing, Geocoding, Data - CARTO SDK

- **Map Data**
  - Formats - vector and raster
  - OpenStreetMap - using and editing
Thank you!

Jaak Laineste

jaak@nutiteq.com