



JÖNKÖPING UNIVERSITY

School of Engineering

ANDROID GPS

Peter Larsson-Green

Jönköping University

Spring 2020

GLOBAL POSITIONING SYSTEM

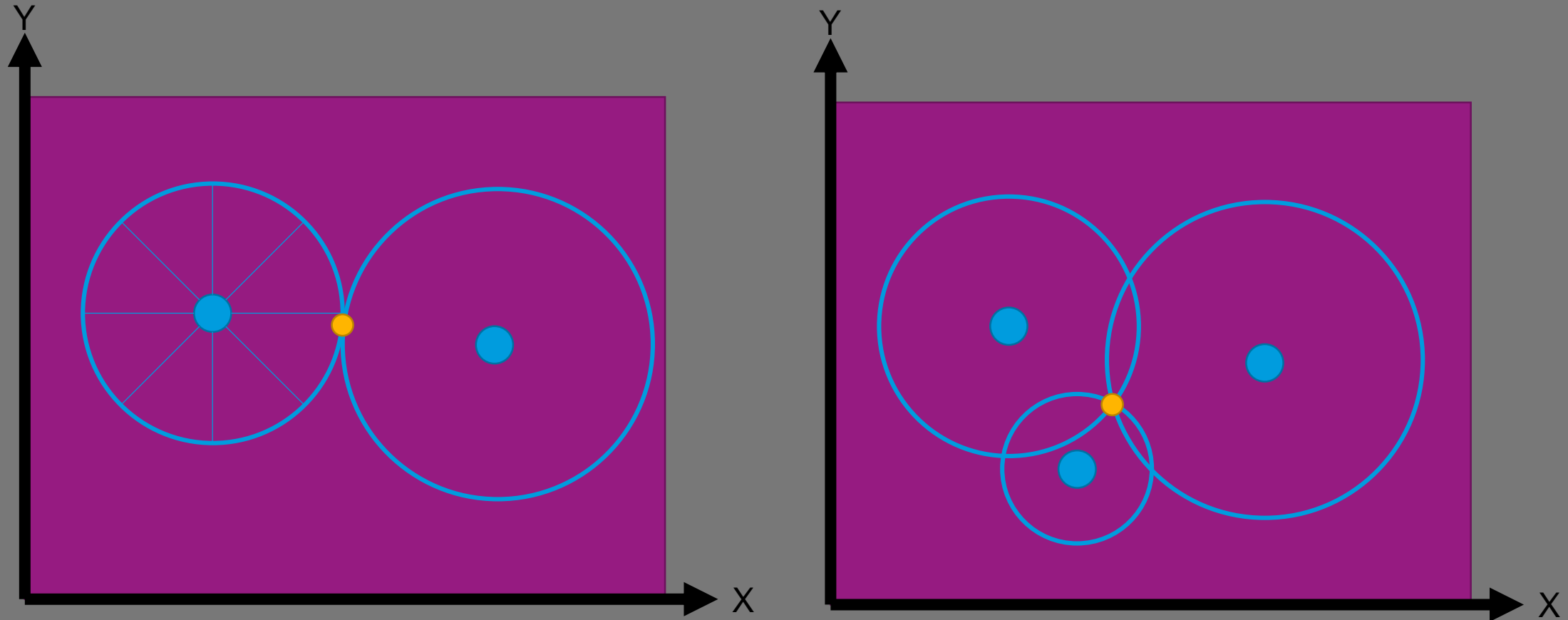
Not only GPS, could be:

- GLONASS (Soviet/Russia)
 - "[...] in October 2011 the full orbital constellation of 24 satellites was restored, enabling full global coverage."
- Galileo (European Union)
 - "As of July 2018, 26 of the planned 30 active satellites are in orbit."
- BeiDou-3 (China):
 - "As of October 2018, fifteen BDS-3 satellites have been launched. BeiDou-3 will eventually consist of 35 satellites and is expected to provide global services upon completion in 2020."

Relies on trilateration:

- More connected satellites → better precision.
- Each connected satellite consumes battery power.

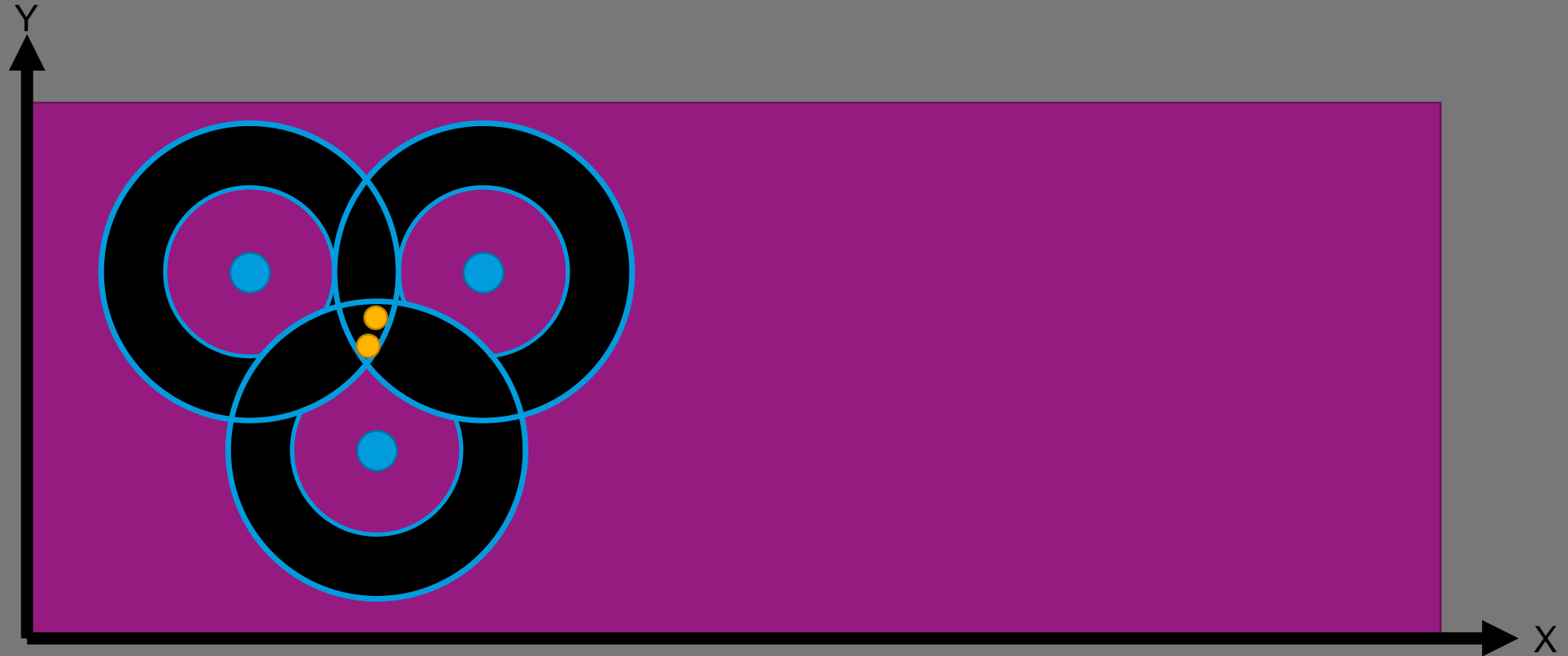
TRILATERATION (2D) IN THEORY



TRILATERATION (3D)

- Distance to each reference point yields a sphere.
- Overlapping spheres yields the border of a circle.
 - 4 reference points needed!

TRILATERATION (2D) IN PRACTICE



TESTING GPS

Use the app GPS test from the Play Store.

- <https://play.google.com/store/apps/details?id=com.chartcross.gptest>



- Number 1-32 are GPS.
- Number 65-96 are GLONASS.

FINDING CURRENT LOCATION

- LocationManager has a set of LocationProviders.
- Are protected by dangerous permissions.
 - ACCESS_COARSE_LOCATION
 - ACCESS_FINE_LOCATION

LOCATION - NETWORK

- Cell tower trilateration (signal strength).
- Public WiFi hotspots.

LOCATION PROVIDERS

```
LocationManager locationManager = (LocationManager)
    aContext.getSystemService(Context.LOCATION_SERVICE);
List<String> availableProviders =
    locationManager.getAllProviders();
for(String provider : availableProviders){
    Log.d("Available Providers", "Provider name: "+provider);
}
```

Provider name: passive

Provider name: gps

Provider name: network

LOCATION PROVIDERS

ACCURACY_LOW	> 500 meter
ACCURACY_MEDIUM	< 500 meter
ACCURACY_HIGH	< 100 meter

```
Criteria criteria = new Criteria();  
criteria.setAltitudeRequired(true);  
criteria.setAccuracy(Criteria.ACCURACY_HIGH);  
String name = locationManager.getBestProvider(criteria, false);  
boolean isEnabled = locationManager.isProviderEnabled(name);  
Location location = locationManager.getLastKnownLocation(name);  
double altitude = location.getAltitude();  
double latitude = location.getLatitude();  
double longitude = location.getLongitude();  
float accuracy = location.getAccuracy();
```

Enabled
only?

68% chance
within a circle
with this radius
[meters].

CREATING LOCATION LISTENERS

```
LocationListener locationManager = new LocationListener() {  
    @Override  
    public void onLocationChanged(Location location) { }  
    @Override  
    public void onProviderEnabled(String provider) { }  
    @Override  
    public void onProviderDisabled(String provider) { }  
};
```

USING LOCATION LISTENERS

```
locationManager.requestLocationUpdates(  
    locationManager.GPS_PROVIDER, 0, 0, locationListener  
);  
locationManager.requestLocationUpdates(  
    locationManager.NETWORK_PROVIDER, 0, 0, locationListener  
);
```



minTime
[milliseconds]

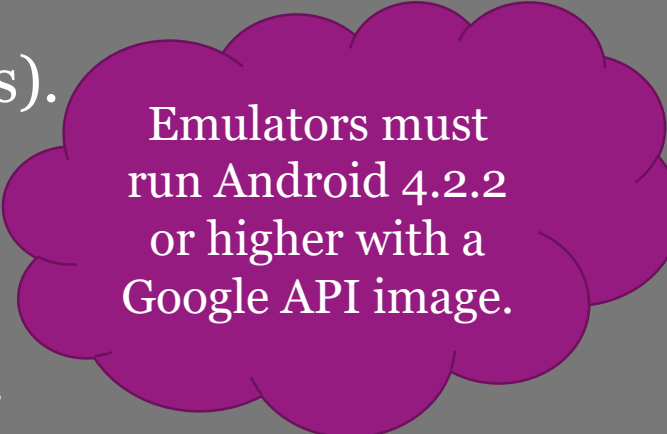
minDistance
[meters]

```
locationManager.removeUpdates(locationListener);
```

GOOGLE PLAY SERVICES

Services provided by Google.

- Not all devices got this (Google Play Store → Google Play Services).
- Includes:
 - Android Pay (Google Wallet).
 - Google Account Login.
 - Google Cast (control Android TV & Chromecast devices).
 - Firebase Cloud Messaging.
 - Google Maps.
 - Google Mobile Ads
 - Google Location Services (the fused location provider).
- Usage in Android Studio: <https://developers.google.com/android/guides/setup>



Emulators must
run Android 4.2.2
or higher with a
Google API image.

THE FUSED LOCATION PROVIDER

- In Android SDK manager: install Google Repository.

```
dependencies {  
    [...]  
    compile 'com.google.android.gms:play-services-location:8.3.0'  
}
```

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

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

GETTING READY

```
GoogleApiClient.Builder clientBuilder =  
    new GoogleApiClient.Builder(aContext);  
clientBuilder.addApi(LocationServices.API);  
clientBuilder.addConnectionCallbacks(  
    new GoogleApiClient.ConnectionCallbacks() {  
        @Override  
        public void onConnected(Bundle bundle) { }  
        @Override  
        public void onConnectionSuspended(int cause) { }  
    }  
);
```


GETTING READY

```
clientBuilder.addOnConnectionFailedListener(  
    new GoogleApiClient.OnConnectionFailedListener() {  
        @Override  
        public void onConnectionFailed(  
            ConnectionResult connectionResult) {  
        }  
    }  
);  
GoogleApiClient playServices = clientBuilder.build();  
playServices.connect();
```

GETTING THE LOCATION

```
Location loc = LocationServices.  
    FusedLocationApi.getLastLocation(playServices);  
  
playServices.disconnect();
```

USING LOCATION LISTENERS

```
LocationRequest locationRequest = new LocationRequest();  
locationRequest.setInterval(1000);  
locationRequest.setNumUpdates(10);  
locationRequest.setExpirationDuration(10000);  
locationRequest.setPriority(  
    LocationRequest.PRIORITY_HIGH_ACCURACY  
);
```

USING LOCATION LISTENERS

```
LocationSettingsRequest.Builder builder =  
    new LocationSettingsRequest.Builder();  
builder.addLocationRequest(locationRequest);  
LocationSettingsRequest locationSettingsRequest =  
    builder.build();
```

USING LOCATION LISTENERS

```
PendingResult<LocationSettingsResult> result =  
    LocationServices.SettingsApi.checkLocationSettings(  
        playServices, locationSettingsRequest  
    );  
result.setResultCallback(  
    new ResultCallback<LocationSettingsResult>() {  
        @Override  
        public void onResult(LocationSettingsResult result) {  
            // Check result.getStatus().getStatusCode().  
        }  
    }  
);
```

USING LOCATION LISTENERS

```
public void onResult(LocationSettingsResult result) {  
    switch(result.getStatus().getStatusCode()) {  
        case LocationSettingsStatusCodes.SUCCESS:  
            // The request can be handled :D  
            break;  
        case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:  
            // The user needs to activate more accurate providers.  
            break;  
        default:  
            // We're screwed :'(  
    }  
}
```

USING LOCATION LISTENERS

```
switch (result.getStatus().getStatusCode()) {  
    case LocationSettingsStatusCodes.SUCCESS:  
        LocationServices.FusedLocationApi.  
            requestLocationUpdates(  
                playServices,  
                locationRequest,  
                new LocationListener() {  
                    @Override  
                    public void onLocationChanged(Location location) { }  
                }  
            );  
}
```

USES FEATURES

- Hides your app from devices lacking necessary features on the Play Store.

```
<manifest ...>  
    <uses-feature android:name="android.hardware.location" />  
    <uses-feature android:name="android.hardware.location.gps" />  
    <uses-feature android:name="android.hardware.camera" />  
    <uses-feature android:name="android.hardware.wifi" />  
    <uses-feature android:name="android.hardware.telephony" />  
</manifest>
```

<http://developer.android.com/guide/topics/manifest/uses-feature-element.html#hw-features>

USES FEATURES

- Some `<uses-permissions>` implies `<uses-feature>`.
- Use `uses-feature` to change that.

```
<manifest ...>
  <uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-feature android:required="false"
    android:name="android.hardware.location" />
  <uses-feature android:required="false"
    android:name="android.hardware.location.gps" />
</manifest>
```