Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

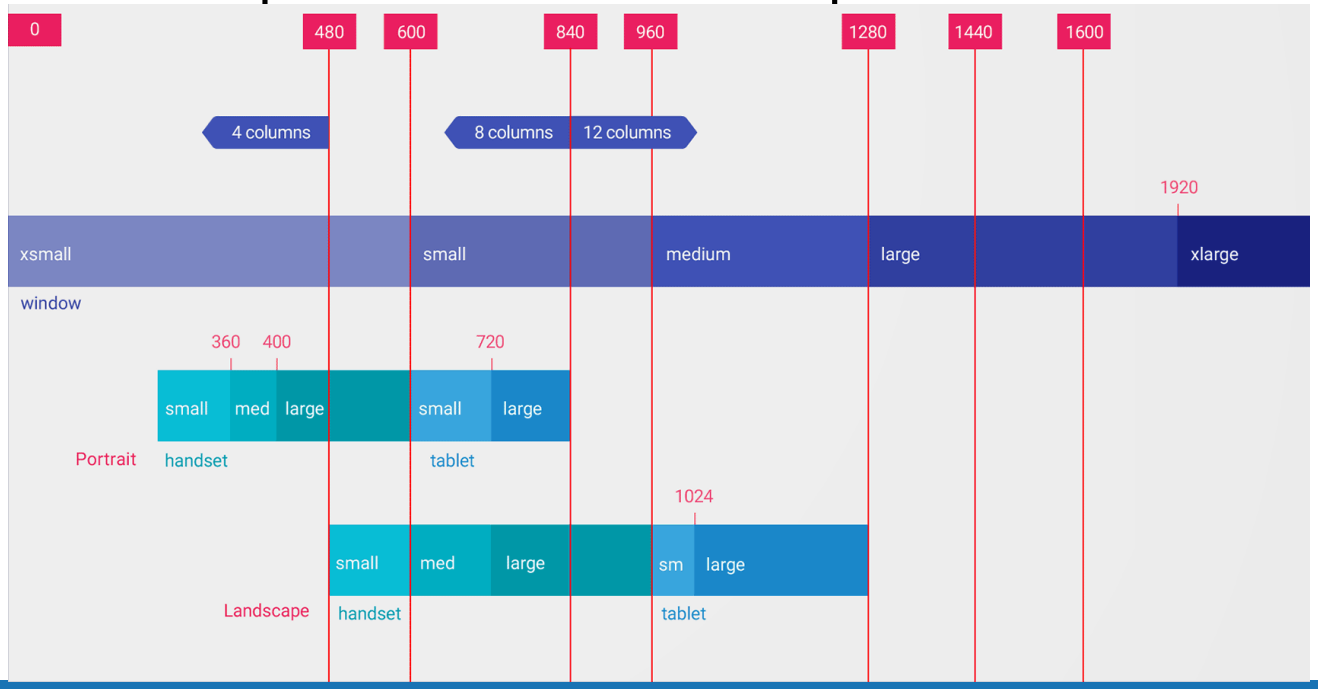
Android Development

Location, Maps

More on UI

Responsive design

- Make a great looking app on every design.
  - Don't think about specific devices!
  - Think about spaces and how much space is available

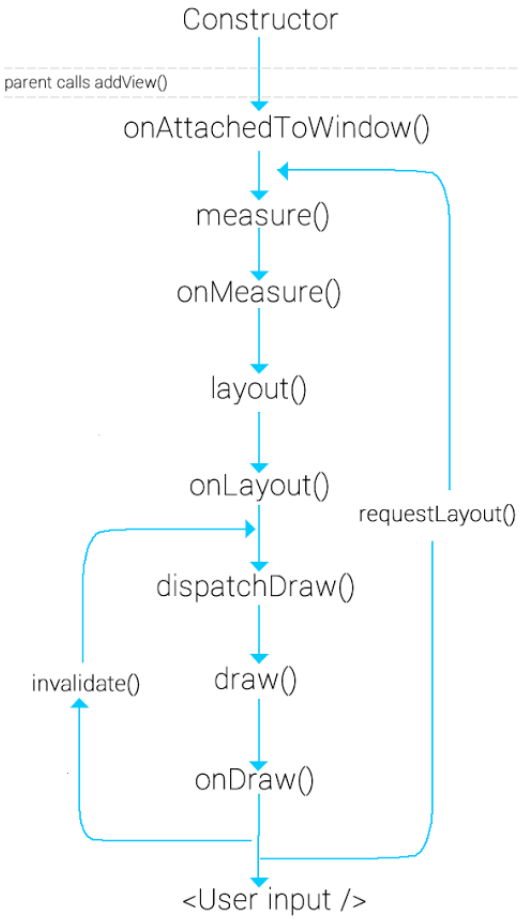


Tools for making a responsive design

1. AndroidManifest.xml possible to restrict the possible screen sizes
2. Make multiple layouts
   - res\layout-swXXXdp\main.xml
   - res\layout-wXXXdp\main.xml
- Possible restrictions
  - smallestWidth - sw<N>dp
  - Available screen width - w<N>dp
  - Available screen height - h<N>dp
- example
  - sw600dp
- Breakpoints
  - 320dp: typical phone sw (240x320 ldpi, 320x480 mdpi, 480x800 hdpi).
  - 600dp: 7" tablet (600x1024 mdpi).
  - 720dp: 10" tablet (720x1280 mdpi, 800x1280 mdpi).
  - Useful article

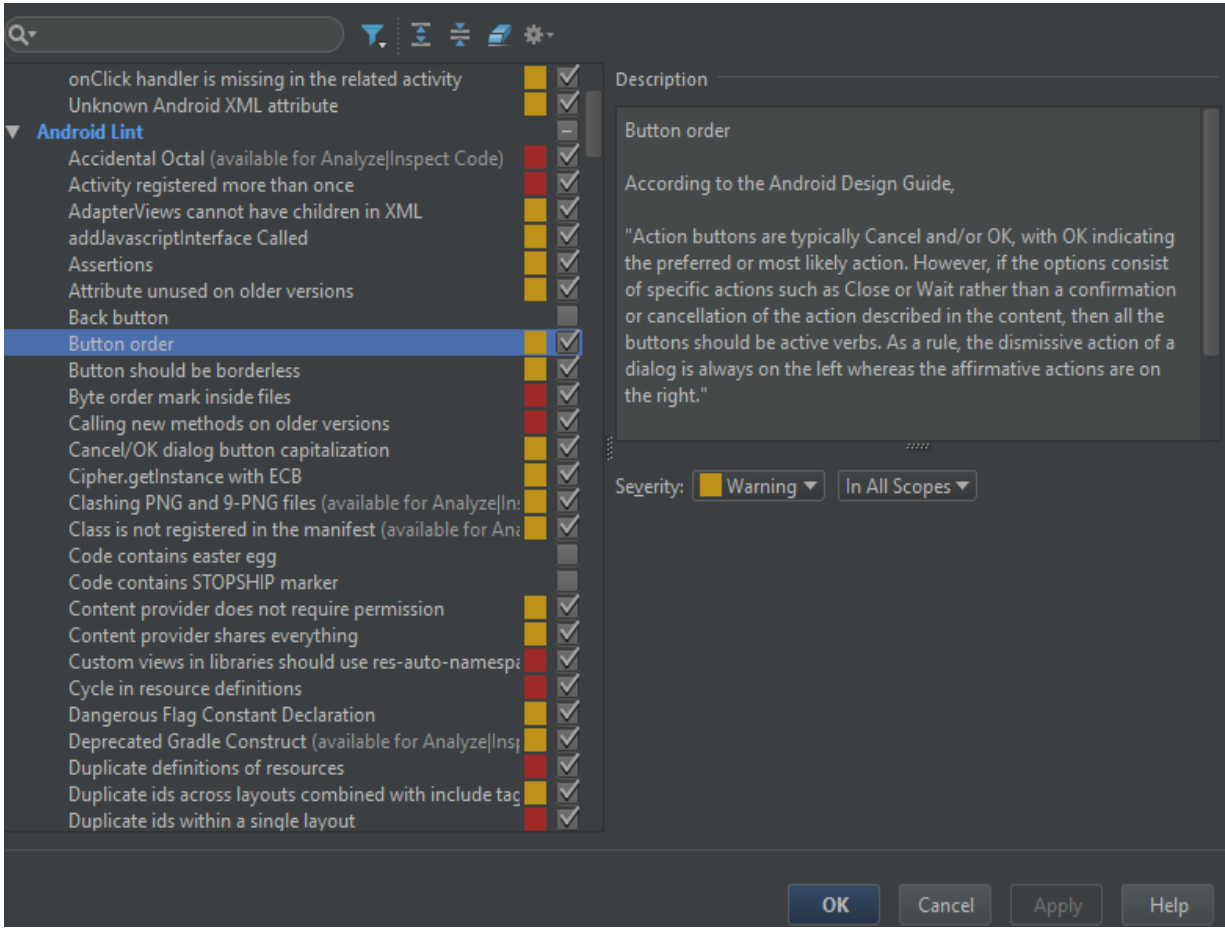
Custom View

- If we would like something new (not among the android default views or any external library) we still can inherent from any View.
- Override the methods starting with „on”.
- Constructor: we can gat the xml parameters here
- onDraw(): draw the View, have to be fast!
- onSizeChanged()
- onMeasure(): call the setMeasuredDimension() func. From inside



Lint

- Static code analyzer
- Gives the code warnings and errors
- Can be configured
- Ran by gradle build
- Gives view optimization tips in XML files.
- For example:
  „NestedWeights”
  „UselessParent”
  „USelessLeaf”



<include>

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width=”match_parent”
    android:layout_height="wrap_content"
    android:background="@color/titlebar_bg">

    <ImageView android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:src="@drawable/gafricalogo" />
</FrameLayout>


<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    android:background="@color/app_bg"
    android:gravity="center_horizontal">

    <include layout="@layout/titlebar"/>

    ...

</LinearLayout>
```

<merge>

```
<merge xmlns:android="http://schemas.android.com/apk/res/android">

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/add"/>

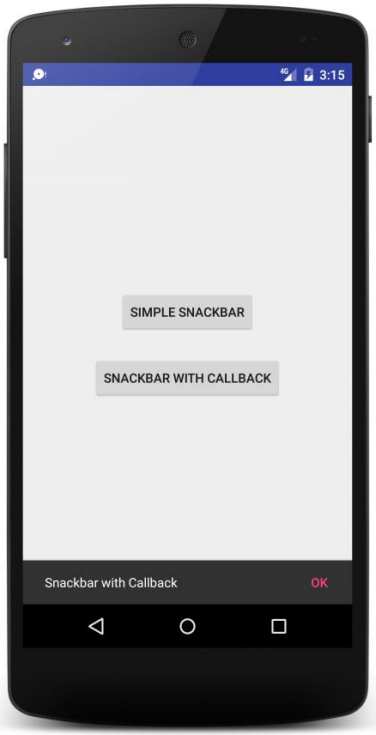
    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/delete"/>

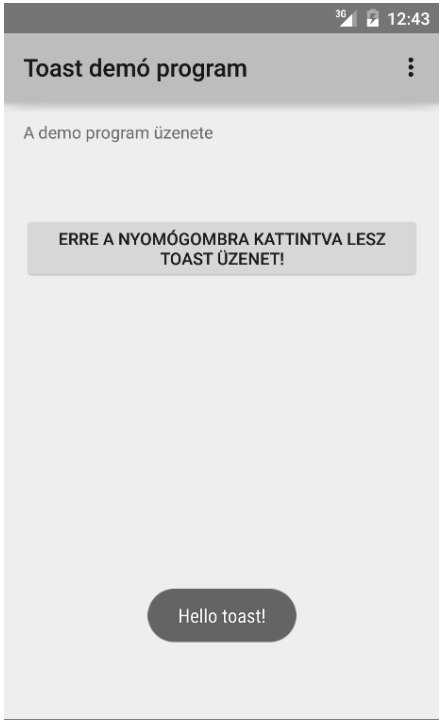
</merge>
```

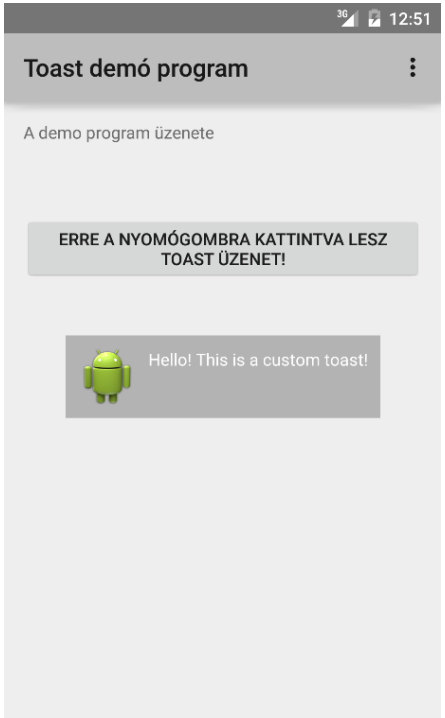
- The <merge> tag is replaced by the parent tag of the <include>.

Sending messages

- `Toast` message
- Snackbar







Toast

```
Context context = getApplicationContext();
String text = "Hello toast!";

int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

- Parameters
  - Size
  - Location
  - Text
  - Duration

Customized Toast

- ```
  LayoutInflater inflater = getLayoutInflater();
  View layout = inflater.inflate(R.layout.toast_layout, (ViewGroup)
  findViewById(R.id.toast_layout_root));
  ImageView image = (ImageView) layout.findViewById(R.id.image);
  image.setImageResource(R.drawable.ic_launcher);
  ```
- ```
  TextView text = (TextView) layout.findViewById(R.id.text);
  text.setText("Hello! This is a custom toast!");
  ```
- ```
  Toast toast = new Toast(getApplicationContext());
  toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
  toast.setDuration(Toast.LENGTH_LONG);
  toast.setView(layout);
  toast.show();
  ```
  ```

Layout for custom toast

```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
    android:id="@+id/toast_layout_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#DAAA"
    android:orientation="horizontal"
    android:padding="10dp" >
    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginRight="10dp" />
    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:textColor="#FFF" />
</LinearLayout>
```

Snackbar

```
Snackbar mySnackbar = Snackbar.make(findViewById(R.id.myCoordinatorLayout),
                                    R.string.email_archived, Snackbar.LENGTH_SHORT);
mySnackbar.setAction(R.string.undo_string, new MyUndoListener());
mySnackbar.show();
```

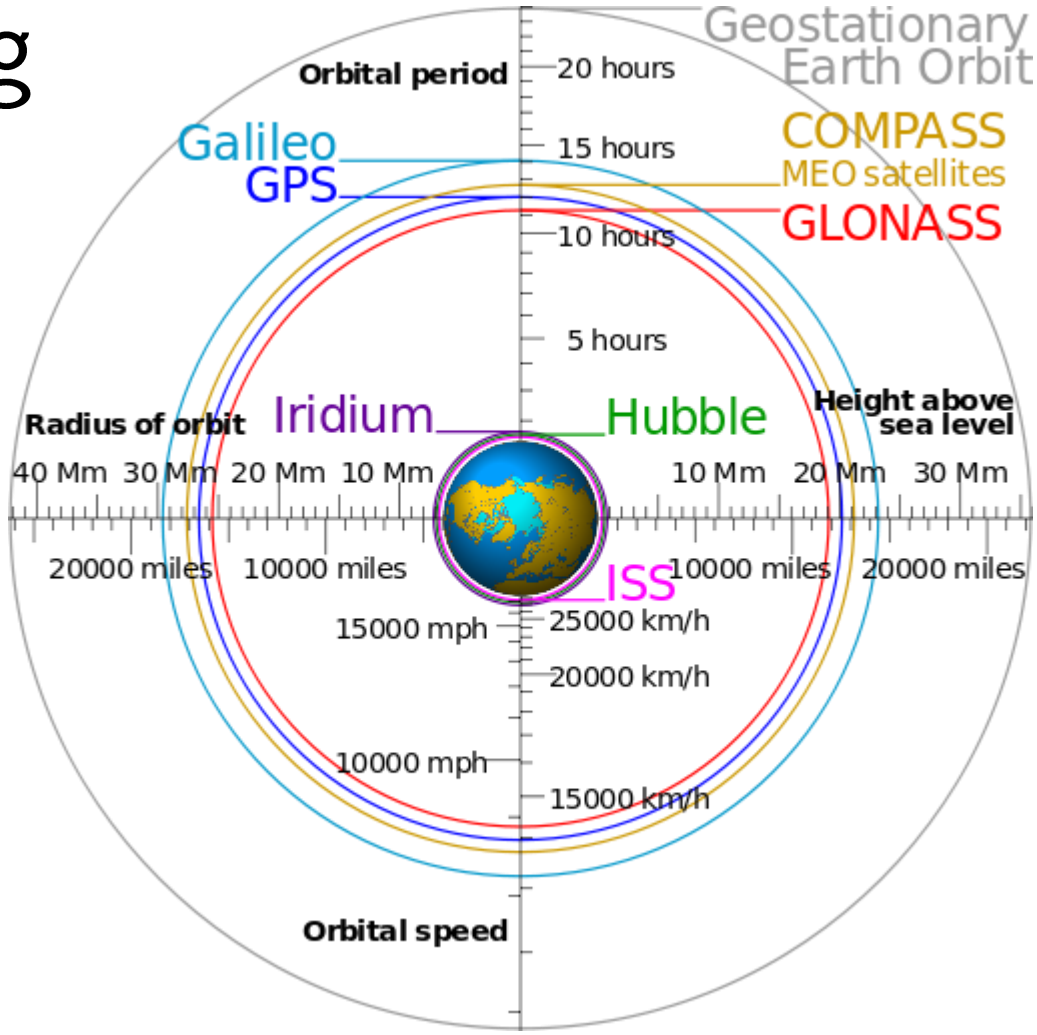
```
public class MyUndoListener implements View.OnClickListener{

    @Override
    public void onClick(View v) {

        // Code to undo the user's last action
    }
}
```

Location

Global Positioning



Google Location Service

- Part of Google Play Services
- Different capabilities
  - To access securely to the location engine
    - Simple API
    - Immediate access
    - Energy efficient
    - Flexible
  - Recognizing the activity of the user
    - What is he/she doing: walking, running, etc.
    - Energy efficient
    - Context-aware service extension
    - Further possibilities
  - Geofencing API
    - Notification on arriving specified places
    - Energy efficient
    - Simple API

Connection

- Application permission

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

Connection

- Google API client
  - ```
    mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();
    ```
- Connecting to the API
  - `mGoogleApiClient.connect()`
- Closing the connection
  - `mGoogleApiClient.disconnect()`
    ```

Connection

- Direct access

```
FusedLocationProviderClient client =
        LocationServices.getFusedLocationProviderClient(this);

// Get the last known location
client.getLastLocation()
        .addOnCompleteListener(this, new OnCompleteListener<Location>() {
            @Override
            public void onComplete(@NonNull Task<Location> task) {
                // ...
            }
        });
```

Last known location – `getLastLocation()`

- Once you have created the Location Services client you can get the last known location of a user's device.
  - When your app is connected to these you can use the fused location provider's getLastLocation() method to retrieve the device location.

```
fusedLocationClient.lastLocation
        .addOnSuccessListener { location : Location? ->
            // Got last known location.
            // In some rare situations this can be null.
        }
```

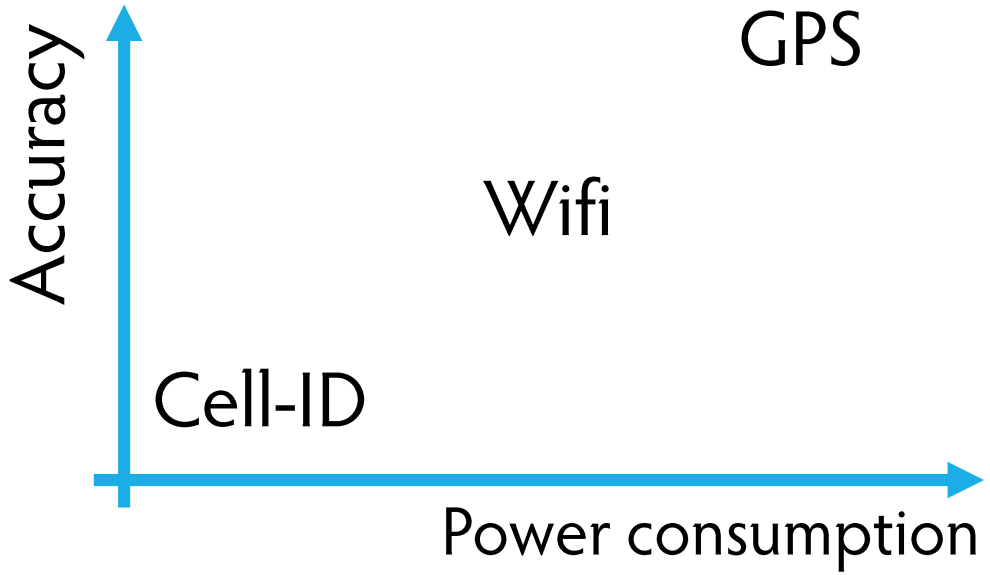
- The getLastLocation() method returns a Task that you can use to get a Location object with the latitude and longitude coordinates of a geographic location.

Maintain a current best estimate

- You might expect the Location object that's contained in the most recent call to getLastLocation() to be the most accurate.
  - However, because the accuracy of a location varies, the most recent value isn't necessarily the best.
  - You should include logic for choosing which location to show based on several criteria.
  - The set of criteria also varies, depending on the use cases of your app and results from field testing.
- To validate the accuracy of a location that's returned from getLastLocation(), complete steps that include the following:
  - Check whether the location retrieved is significantly newer than the previously fetched location.
  - Check whether the accuracy claimed by the location is better or worse than that of the previous estimate.
  - Check the provider that's associated with the new location. Decide whether you trust this provider more than the one used in your app's cached location.

Location Accuracy

- GPS, Cell-ID, and Wi-Fi can each provide a clue to users location.
  - Determining which to use and trust is a matter of trade-offs in accuracy, speed, and battery-efficiency.



Accessing to the position continuously

- First the parameters should be specified for the request
  - `LocationRequest` class has to be used
- Possibilities
  - `setInterval()` – How often is the location required
  - `setFastestInterval()` – The available fastest interval
  - `setPriority()` – The priority of the request
    - `PRIORITY_BALANCED_POWER_ACCURACY`
    - `PRIORITY_HIGH_ACCURACY`
    - `PRIORITY_LOW_POWER`
    - `PRIORITY_NO_POWER`
- Example

```
val locationRequest = LocationRequest.create()?.apply {
    interval = 10000
    fastestInterval = 5000
    priority = LocationRequest.PRIORITY_HIGH_ACCURACY
}
```

Accessing to the position continuously

- The Google API will call the `onConnect()` function upon connection is completed.
- Then the request can ben sent by invoking `requestLocationUpdates()`
  - An interface must be bypassed, which is a callback (`LocationListener`)
  - Abandon the request `removeLocationUpdates()`

Accessing to the position continuously

- Example – Request

```
override fun onResume() {
    super.onResume()
    if (requestingLocationUpdates)
startLocationUpdates()
}

private fun startLocationUpdates() {
    fusedLocationClient.requestLocationUpdates(l
ocationRequest,
            locationCallback,
            Looper.getMainLooper())
}
```

Accessing to the position continuously

- Example - Callback

```
private lateinit var locationCallback: LocationCallback

override fun onCreate(savedInstanceState: Bundle?) {


    locationCallback = object : LocationCallback() {
        override fun onLocationResult(
                            locationResult: LocationResult?) {
            locationResult ?: return
            for (location in locationResult.locations){

            }
        }
    }
}
```

Accessing to the position continuously

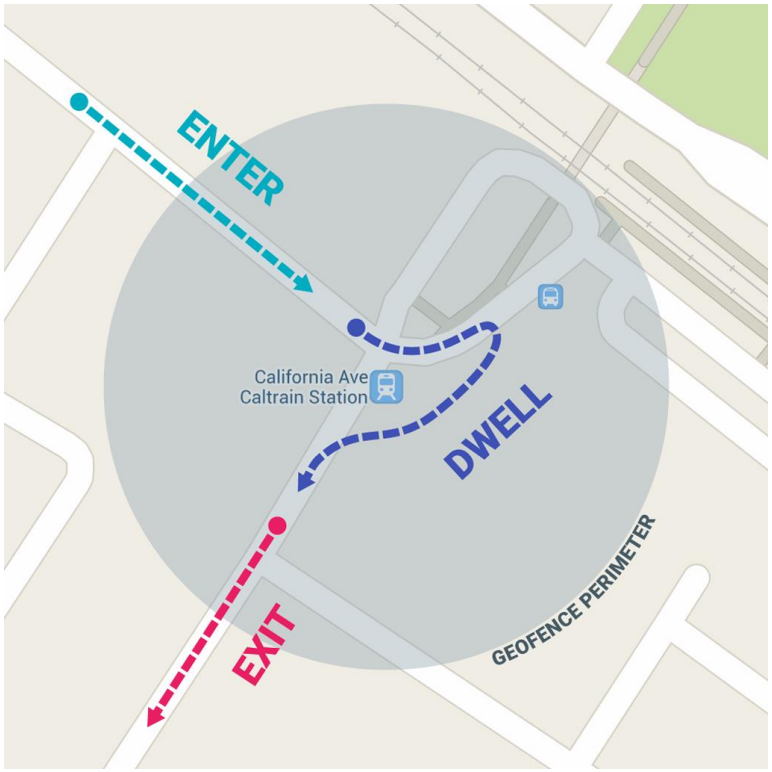
- Example – Stop

```
override fun onPause() {
    super.onPause()
    stopLocationUpdates()
}

private fun stopLocationUpdates() {
    fusedLocationClient.removeLocationUpdates(locationCall
back)
}
```

Creating and monitoring a geofence

- Notification about that a user enters/leaves a specific location.



Creating a location

- A builder is used to create a geofence (`Geofence.Builder`)
- Example

```
geofenceList.add(Geofence.Builder()

    .setRequestId(entry.key)
    .setCircularRegion(
        entry.value.latitude,
        entry.value.longitude,
        Constants.GEOFENCE_RADIUS_IN_METERS
    )

    .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or
                        Geofence.GEOFENCE_TRANSITION_EXIT)

    .build())
```

Parameters

- Functions
  - `setCircularRegion` – parameters are the center and the diameter
  - `setExpirationDuration` – to set the lifetime of the geofence
  - `setTransitionTypes` – to specify which activities will trigger notification (you can add more, they are connected with ‚or')
    - `GEOFENCE_TRANSITION_DWELL`
    - `GEOFENCE_TRANSITION_ENTER`
    - `GEOFENCE_TRANSITION_EXIT`
  - `setLoiteringDelay` – to specify the duration which has to be elapsed before the user considered as staying at the location

Creating a request

- Once the Geofence object is ready, it has to be added to a GeofenceRequest.
  - Several Geofence-s can be added in to one request
  - The initial trigger should be specified
- Example

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

Intent and Geofence API

- In addition an Intent object has to be specified which will be used to call back
- Intent
  - `Intent intent = new Intent(this, GeofenceTransitionsIntentService.class);`
  - `// We will use this Intent`
    `PendingIntent.getService(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);`
- API
  - `LocationServices.GeofencingApi.addGeofences(mGoogleApiClient, mGeofencingRequest, mGeofencePendingIntent()).setResultCallback(this);`

Receiving notifications

- Through a service

```
public class GeofenceTransitionsIntentService extends IntentService {
   …
  protected void onHandleIntent(Intent intent) {
    GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);
    if (geofencingEvent.hasError()) { … }
    int geofenceTransition = geofencingEvent.getGeofenceTransition();
    if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER ||
        geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT)
    {
      List triggeringGeofences =
          geofencingEvent.getTriggeringGeofences();
      String geofenceTransitionDetails =
          getGeofenceTransitionDetails(this, geofenceTransition,
                    triggeringGeofences);
          …
    } else {
      …
    }
}
```

Receiving notifications

- Do not forget to add the service in `AndroidManifest.xml` file

```
<application
    android:allowBackup="true">
    ...
    <service android:name=".GeofenceTransitionsIntentService"/>
<application/>
```

Unregister

- Similar to register

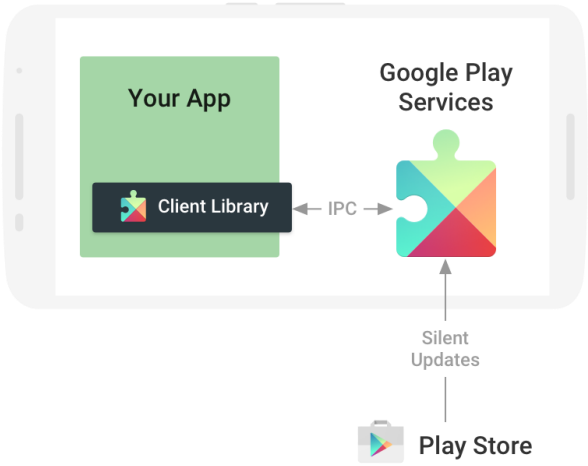
```
geofencingClient?.removeGeofences(geofencePendingIntent)?.run {
    addOnSuccessListener {
        // Geofences removed
        // ...
    }
    addOnFailureListener {
        // Failed to remove geofences
        // ...
    }
}
```

- A callback interface (`ResultCallback`) is required

Maps API

Google Play Services (GMS)

- Google Play Services is a proprietary background service and API package for Android devices from Google.
  - It is not part of the Android OS!
- You can get the Google Play Services from the Play Store.
  - The updates can come to the devices faster then an OS update.
- The app contains client libraries which can be called only by GMS services.
- If there is no GMS the client app will ask for it.



Google Play Services – parts

- **Google Maps**: For embedding Google Maps and Street View
- **Google Account Login**: Log in with Google account
- **Google Drive**: API for storing and synchronizing with Drive
- **Google Fit:** user movement related API
- **Google Play Game services:** game stats, multiplayer, game saves and more
- **Google Location APIs**: API for getting the current location of the user efficiently
- **Android Wear**: API for smartwatches
- **Google Mobile Ads**: API for google add platform
- **Google Cloud Messaging**: API for push notifications
- **Google Analytics:** API for live statistics and user feedback
- **Google Wallet, Mobile Vision, Google Cast, and so on the full list**: https://developers.google.com/android/guides/setup

Add GMS to the project

1. In the SDK Manager download *Google Repository*, *Android Support Repository*
2. In the build.gradle add:

```
dependencies {
    ...
    implementation 'com.google.android.gms:play-services:7.8.0'
}
```

If you only need a part from GMS (for example maps):

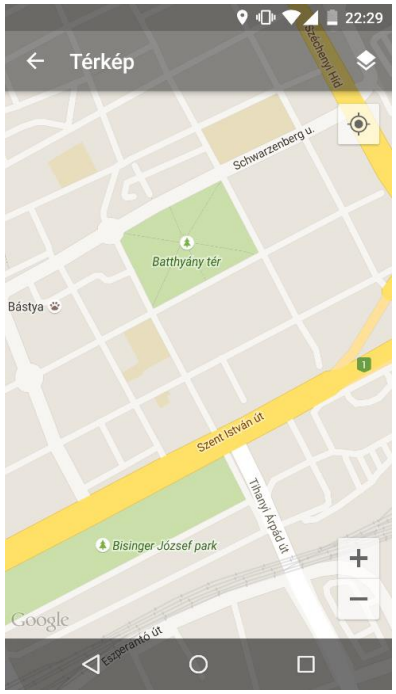
```
dependencies {
    ...
    implementation 'com.google.android.gms:play-services-maps:16.1.0'
}
```

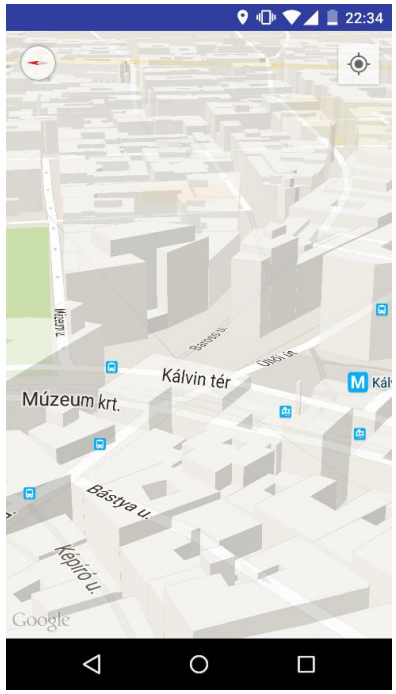
Note, that the version number of the library might be different

3. Sync Project with Gradle Files

Maps API

- Map appearance
  - Normal
  - Hybrid
  - Satellite
  - Terain
  - Indoor
  - Lite mode
- Street View
- Draw on map
  - Marker
  - Popup
  - Custom shapes
  - Custom layers
- You can interact with the map (zoom, pan, tilt, rotate, compass)





Maps – generate API key

- To use the Maps API you need to generate an API key.
- The key needs to be in the app.
- With the key Google can monitor the usage rate of each service.
  - Map API is free until a certain number of server calls.
  - Without an API key the map is not working
- To generate a key you will need SHA-1 fingerprint and the package name of the app.
- The generated key needs to beaded to AndroidManifest.
- Now we will use Android Studio instead.

Maps – editing manifest

- To use maps we need internet access.
- If we want to show the users current location, we need to add more permissions to the manifest.

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

- If you are targeting 8.3 or **earlier versions** of the Google Play services SDK, you must request

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

Try!

- The Anroid Studio can do the hard work for us
- File → New → Google → Google Maps Activity → Finish
  - To open it directly make it the launcher activity
- Open up *src/debug/res/values/google_maps_api.xml* !
- Open the long commented link to generate the api key.
- Copy the key to the YOUR_KEY_HERE placeholder !
- Open MapsActivity-t!
- If done correctly a map will appear
- Remove the unnecessary modules!

MapFragment

- The `MapFragment` is a `Fragment` it can present a map.
- It can be used as any other `Fragment`:
  - Can be added from xml.
  - Dynamically, from code trough `FragmentManager`
- This class only makes an encapsulation, the map functionality is trough `GoogleMap`.
- An instance of this can be obtained from MapFragment
  - We need to implement the `OnMapReadyCallback` interface
  - `Call MapFragment getMapAsync(callback)` method
  - In the `onMapReady()` `method we get the GoogleMap object.`

Add marker

- We can add markers to the map with the `addMarker()` method
- This is waiting for a `MarkerOptions` object.
  - This is a Fluent builder

`position()`: `LatLng` coordinates of the marker
`title()`: label of the marker
`snippet()`: text under marker
`draggable()`: can be dragged
`icon()`: image of the marker
Etc.

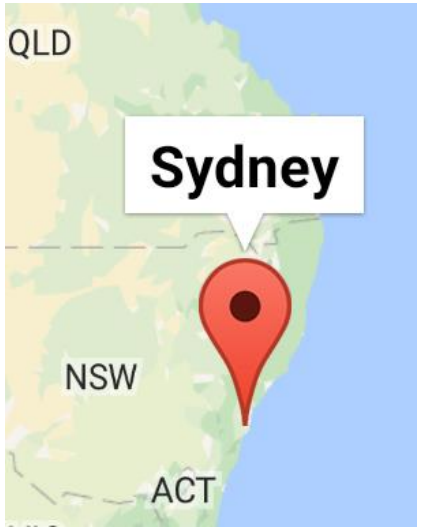
```
mMap.addMarker(new MarkerOptions().position(
new LatLng(-34, 151)).title("Sydney"));
```

Info label

- The info label can present information about the marker
  - By default, it shows the title
  - It can be customised
  - The info label is shown when tapped

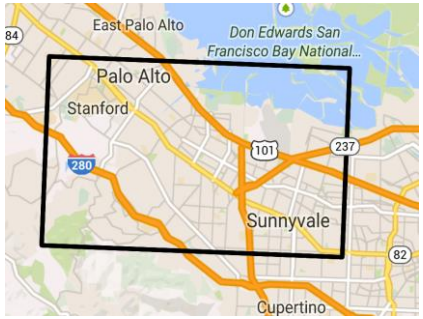
It can be also forced to appear and disappear

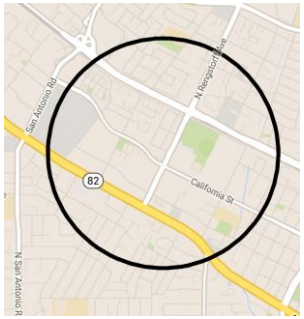
```
Marker sydney = mMap.addMarker(
new MarkerOptions().position(
new LatLng(-34, 151)).title("Sydney"));
sydney.showInfoWindow();
// sydney.hideInfoWindow();
```



GoogleMap – drawing

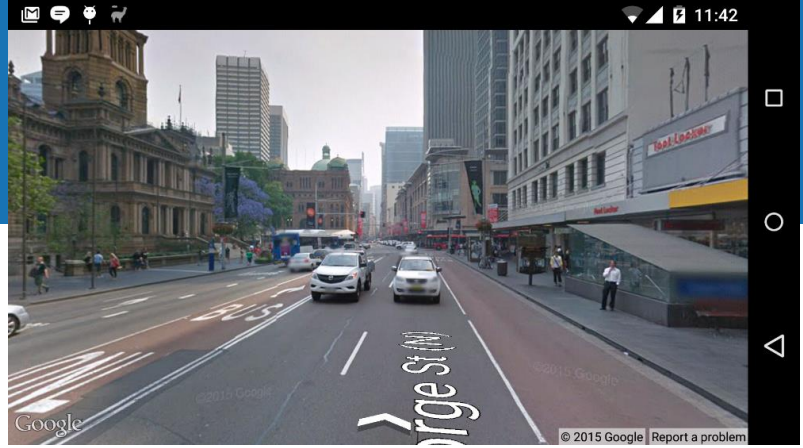
- Shape drawing
  - addCircle()
  - addPolyLine()
  - addPolygon()
  - Stb.





- Ground Overlays
  - A new layer can be added to the map which is moving with the map.
    - addGroundOverlay()
- Tile Overlays
  - You can change the zoom layers of the map and add custom images.
    - addTileOverlay()

StreetView



- Very similar to MapFragment
- You need to use `StreetViewPanoramaFragment`
- The model class is `StreetViewPanorama`
- To get it use `getStreetViewPanoramaAsync()` and give it `OnStreetViewPanoramaReadyCallback`
- In the `onStreetViewPanoramaReady` method we can get the `StreetViewPanorama` object

GoogleMap – user interaction

- With the `UiSettings` interface we can control what UI elements appear
  - We can get this with the help of `getUiSettings()` method.
- Zoom: `setZoomControlsEnabled(true)`
- Compass: `setCompassEnabled(boolean)`
- Current location: `setMyLocationButtonEnabled(boolean)`
- Map toolbar: `setMapToolbarEnabled(boolean)`
- The map can handle gestures:
  - Zoom: double tap, two finger tap, pinch/stretch
  - Scroll/pan
  - Rotate
- We can subscribe or block theese events.

GoogleMap – change map view

- You can change the map programatically on the fly.
- The map is modelled by a camera with theese parameters:
  - Target (location)
  - Zoom
  - Bearing (orientation)
  - Tilt (viewing angle)

- The change of the camera is represented by `CameraUpdate` class
  - An instance of this can be made by the `CameraUpdateFactory`
  - To change the camera we need to pass this object to the GoogleMap `moveCamera()` or `animateCamera()` method
  - The first is an instant change the second is with an animation

Example

```
// Move the camera instantly to Sydney with a zoom of 15.
map.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(-33.88, 151.21), 15));

// Zoom in, animating the camera.
map.animateCamera(CameraUpdateFactory.zoomIn());

// Zoom out to zoom level 10, animating with a duration of 2 seconds.
map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);

// Construct a CameraPosition focusing on Mountain View and animate the camera to
that position.
CameraPosition cameraPosition = new CameraPosition.Builder()
    .target(new LatLng(37.4, -122.1)) // Sets the center of the map to Mountain
View
    .zoom(17)                   // Sets the zoom
    .bearing(90)                // Sets the orientation of the camera to east
    .tilt(30)                   // Sets the tilt of the camera to 30 degrees
    .build();                   // Creates a CameraPosition from the builder
map.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

Android NDK

NDK

- Native Development Kit
  - To develop external libraries in C, C++ language.
    - These libs can be used in applications
    - Pros and contras as well
- What is required?
  - NDK - https://developer.android.com/tools/sdk/ndk/index.html
  - SDK Manager
    - SDK Tools: LLDB, Cmake, NDK

Content of NDK

- NKD contains the API, documentation, sample applications
  - Tools for compiling C/C++ source to native libraries
  - Tools for include libs to .apk files, in order to install on devices or on the emulator
  - System headers, libraries (which are supported by all version, starting from 1.5, and for native Activities from 2.3)
  - Documentation, sample codes, and tutorials
- The supported architectures

Suported architectures

ABI	Supported Instruction Sets	Notes
armeabi-v7a	Armeabi Thumb-2 VFPv3-D16	Incompatible with ARMv5/v6 devices.
arm64-v8a	AArch64	
x86	x86 (IA-32) MMX SSE/2/3 SSSE3	No support for MOVBE or SSE4.
x86_64	x86-64 MMX SSE/2/3 SSSE3 SSE4.1, 4.2 POPCNT	

Tools for developments

- The required cross-toolchain (compiler, linker, ...) is in the NDK
- Available system headers available (and guarenteed)
  - libc (C library) headers
  - libm (math library) headers
  - JNI interface headers
  - libz (Zlib compression) headers
  - liblog (Android logging) header
  - OpenGL ES 1.1 and OpenGL ES 2.0, 3.2 (3D graphics libraries) headers
  - libjnigraphics (Pixel buffer access) header (for Android 2.2 and above).
  - A extremely minimal set of headers for C++ support
  - OpenSL ES native audio libraries
  - Android native application APIS
  - OpenMAX AL
  - Vulkan API

Differences – example

API LEVEL 3

- arc-arm

API LEVEL 21

- arch-arm
  arch-arm64
  arch-mips
  arch-mips64
  arch-x86
  arch-x86_64

Example – Hello

- Native (C++) file (jni/hello-jni.cpp)

```
#include <string.h>
#include <jni.h> jstring
Java_com_example_hellojni_HelloJni_stringFromJNI(JNIEnv*
env,jobject thiss)
{ return (*env)->NewStringUTF(env, "Hello from JNI!"); }
```

Example – Hello

- HelloJni.java

```
package com.example.hellojni;
import android.app.Activity;
import android.widget.TextView;
import android.os.Bundle;
public class HelloJni extends Activity {
        public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText( stringFromJNI() );
        setContentView(tv);
        }
        public native String stringFromJNI;
        public native String unimplementedStringFromJNI;

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

Example

- Two `native` functions have been declared in the Java code
  - However, the corresponding C++ functions do not exist
  - But the Java code compiles and runs
  - The implementation of native function only trigger, when they are called at the first time
- The library is loaded in the static initialization block of the Java code
  - A good example for static initialization blocks.
  - The lib is extracted from the .apk file once it is installed on the device
    - /data/data/package.neve/lib/libhello_jni.so

Native and corresponding Java types

- boolean
  byte
  char
  short
  int
  long
  float
  double
  void
  java.lang.Class
  java.lang.String
  array
  -- double array

- jboolean
  jbyte
  jchar
  jshort
  jint
  jlong
  jfloat
  jdouble
  void
  jclass
  jstring
  jarray
  -- jdoubleArray

Native Activity – example

- AndroidManifest.xml

```
<uses-sdk android:minSdkVersion="9" />
<!-- This .apk has no Java code itself, so set hasCode to false. -->
<application android:label="@string/app_name" android:hasCode="false">
  <activity android:name="android.app.NativeActivity "
            android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden">
    <meta-data android:name="android.app.lib_name„
               android:value="native-activity" />
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
</application>
```

Native Activity – example

- Android.mk
  - ```
    LOCAL_MODULE    := native-activity
    LOCAL_SRC_FILES := main.c
    LOCAL_LDLIBS    := -llog -landroid -lEGL -
    lGLESv1_CM
    ```
  - Next has to be defined to handle the life cycle of Activity
    ```
    LOCAL_STATIC_LIBRARIES :=
    android_native_app_glue
    $(call import-module,android/native_app_glue)
    ```
- Application.mk
  - `APP_PLATFORM := android-10`
    ```

Native Activity – highlights

- ```
  #include <jni.h>
  #include <errno.h>
  #include <EGL/egl.h>
  #include <GLES/gl.h> #include <android/sensor.h>
  #include <android/log.h>
  #include <android_native_app_glue.h>
  ```
- ```
  #define LOGI(...)((void)__android_log_print(ANDROID_LOG_INFO,
  "native-activity", __VA_ARGS__))
  #define LOGW(...)((void)__android_log_print(ANDROID_LOG_WARN,
  "native-activity", __VA_ARGS__))
  ```

Native Activity – highlights

```
static void engine_draw_frame(struct engine* engine) {
  if (engine->display == NULL) {
  return;
  }
  glClearColor(((float)engine->state.x)/engine->width,engine>state.angle,
  ((float)engine->state.y)/engine->height,1);
  glClear(GL_COLOR_BUFFER_BIT);
  eglSwapBuffers(engine->display, engine->surface);
}
```

Native Activity – highlights

```
static int32_t engine_handle_input(struct android_app* app, AInputEvent*
  event) {
  struct engine* engine = (struct engine*)app->userData;
  if (AInputEvent_getType(event) == AINPUT_EVENT_TYPE_MOTION) {
    engine->animating = 1;
    engine->state.x = AMotionEvent_getX(event, 0);
    engine->state.y = AMotionEvent_getY(event, 0);
    return 1;
  }
  return 0;
}
```

Native Activity – highlights

```
void android_main(struct android_app* state) {
  struct engine engine;
  app_dummy();
  memset(&engine, 0, sizeof(engine));
  state->userData = &engine;
  state->onAppCmd = engine_handle_cmd;
  state->onInputEvent = engine_handle_input;
  engine.app = state;

  engine.sensorManager = ASensorManager_getInstance();
  engine.accelerometerSensor =
  ASensorManager_getDefaultSensor(engine.sensorManager,
  ASENSER_TYPE_ACCELEROMETER);
  engine.sensorEventQueue =
  ASensorManager_createEventQueue(engine.sensorManager,
  state->looper,
  LOOPER_ID_USER, NULL, NULL);
  if (state->savedState != NULL) {
    engine.state = *(struct saved_state*)state->savedState;
  }
```

Native Activity – highlights

```
while (1) {
   int ident;
   int events;
   struct android_poll_source* source;
   while ((ident=ALooper_pollAll(engine.animating ? 0 : -1, NULL, &events,(void**)&source)) >= 0) {
     if (source != NULL) {
       source->process(state, source);
     }
     if (ident == LOOPER_ID_USER) {
       if (engine.accelerometerSensor != NULL) {
         ASensorEvent event;
           while (ASensorEventQueue_getEvents(engine.sensorEventQueue,&event, 1) > 0) {
             LOGI("accelerometer: x=%f y=%f z=%f",event.acceleration.x, event.acceleration.y,event.acceleration.z);
           }
       }
   }
     if (state->destroyRequested != 0) {
       engine_term_display(&engine);
       return;
     }
   }
   if (engine.animating) {
     engine.state.angle += .01f;
     if (engine.state.angle > 1) {
       engine.state.angle = 0;
     }
   engine_draw_frame(&engine);
   }
}
```

Homework

- Create an application with two activity.
  - The first activity should contain a list of countries, with images.
  - If you tap on a country in the second activity a map should appear, and you should place a marker on the map
  - You also need to add some description about the country to the marker.
    - (please zoom into the marker's position)
  - On the map you should be able to zoom, pan, drag.

Build process, testing

Next week