



# Lecture 15: Location & Maps

## IN721: Design and Development of Applications for Mobile Devices

### Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Friday, 8 May

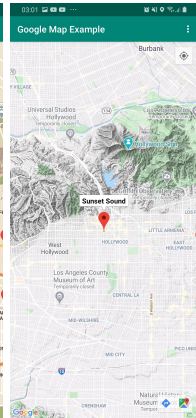
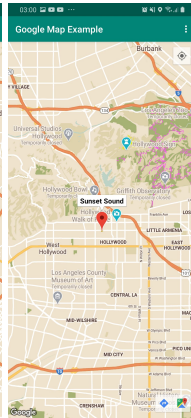
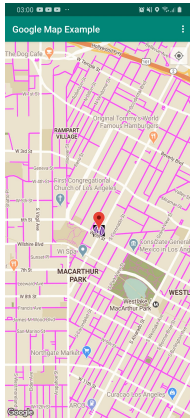
## LECTURE 14: CAMERA TOPICS

- ▶ Camera API

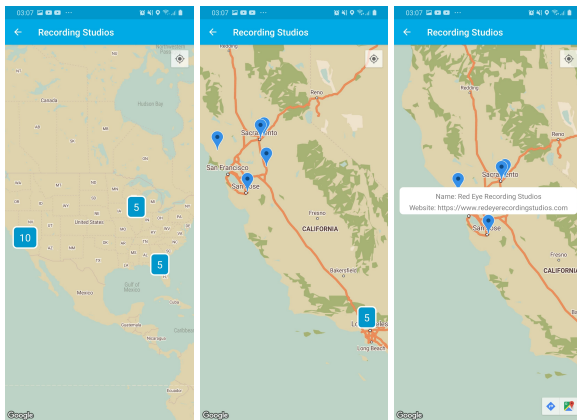
# LECTURE 15: LOCATION & MAPS TOPICS

- ▶ Location
- ▶ Google Maps

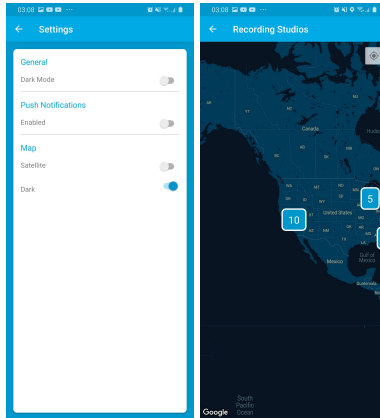
# TODAY'S PRACTICAL - PART ONE



# TODAY'S PRACTICAL - PART TWO (RESEARCH)



# TODAY'S PRACTICAL - PART TWO (RESEARCH)



# LOCATION - MAPSACTIVITY

- Create a new class called MapsActivity which extends AppCompatActivity or BaseActivity & OnMapReadyCallback

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {  
  
    private lateinit var map: GoogleMap  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_maps)  
        val mapFragment: SupportMapFragment = supportFragmentManager  
            .findFragmentById(R.id.map) as SupportMapFragment  
        mapFragment.getMapAsync(this@MapsActivity)  
    }  
}
```

# LOCATION - ONMAPREADY

- ▶ Override function - OnMapReady
- ▶ Reference to Google Map

```
override fun onMapReady(googleMap: GoogleMap) {  
    map = googleMap  
  
    val data: ArrayList<MapsData> = arrayListOf(  
        MapsData( name: "Sunset Sound", LatLng(34.0977818, -118.3349175), R.drawable.sunset_sound),  
        MapsData( name: "MIX Recording Studio", LatLng(34.0628191, -118.281333), R.drawable.mix_recording_studio),  
        MapsData( name: "SilverLake Recording Studios", LatLng(34.089849, -118.281676), R.drawable.silverlake_recording_studios)  
    )  
  
    val zoomLevel = 15f  
    map.moveCamera(CameraUpdateFactory.newLatLngZoom(data[1].location, zoomLevel))  
  
    for (d: MapsData in data) {  
        map.addMarker(  
            MarkerOptions()  
                .position(d.location)  
                .title(d.name)  
        )  
        val overlaySize = 100f  
        val googleOverlay: GroundOverlayOptions = GroundOverlayOptions()  
            .image(BitmapDescriptorFactory.fromResource(d.drawable))  
            .position(d.location, overlaySize)  
        map.addGroundOverlay(googleOverlay)  
    }  
  
    setMapLongClick(map)  
    setPointOfInterest(map)  
    setMapStyle(map)  
    enableMyLocation()  
}
```



# LOCATION - MENU

- ▶ Menu containing four items
- ▶ Change the map's type
  - ▶ Normal
  - ▶ Hybrid
  - ▶ Satellite
  - ▶ Terrain

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    val inflater: MenuInflater = menuInflater  
    inflater.inflate(R.menu.map_options, menu)  
    return true  
}  
  
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.itemId) {  
        R.id.normal_map -> map.mapType = GoogleMap.MAP_TYPE_NORMAL  
        R.id.hybrid_map -> map.mapType = GoogleMap.MAP_TYPE_HYBRID  
        R.id.satellite_map -> map.mapType = GoogleMap.MAP_TYPE_SATELLITE  
        R.id.terrain_map -> map.mapType = GoogleMap.MAP_TYPE_TERRAIN  
    }  
    return true  
}
```

# LOCATION

- ▶ Private function - setMapLongClick
- ▶ Reference to Google Map
- ▶ setOnMapLongClickListener
  - ▶ Add a new marker to the map
  - ▶ Add marker options - position, title, snippet & icon

```
private fun setMapLongClick(map: GoogleMap) {  
    map.setOnMapLongClickListener { latLng ->  
        val snippet: String =  
            "Latitude: " + latLng.latitude + ", " + "Longitude: " + latLng.longitude  
        map.addMarker(  
            MarkerOptions()  
                .position(latLng)  
                .title(getString("Dropped Pin"))  
                .snippet(snippet)  
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE))  
        )  
    }  
}
```

# LOCATION

- ▶ Private function - setPointOfInterest
- ▶ Reference to Google Map
- ▶ Displays each MapsData object's name in an info window
- ▶ Default marker colour - red

```
private fun setPointOfInterest(map: GoogleMap) {  
    map.setOnPoiClickListener { poi ->  
        val poiMarker: Marker = map.addMarker(  
            MarkerOptions()  
                .position(poi.latLng)  
                .title(poi.name)  
        )  
        poiMarker.showInfoWindow()  
    }  
}
```

# LOCATION - MAP STYLE

- ▶ Private function - setMapStyle
- ▶ Reference to Google Map
- ▶ Loads raw resource style from raw resource directory
- ▶ JSON file containing map styling information
- ▶ Error checking - loading & missing resource style
- ▶ Resource - <https://mapstyle.withgoogle.com/>

```
private fun setMapStyle(map: GoogleMap) {  
    try {  
        val success: Boolean = map.setMapStyle(  
            MapStyleOptions.loadRawResourceStyle(  
                this,  
                R.raw.map_style  
            )  
        )  
        if (!success) {  
            Toast.makeText( context: this@MapsActivity, text: "Error loading map style.", Toast.LENGTH_LONG)  
                .show()  
        }  
    } catch (e: Resources.NotFoundException) {  
        Toast.makeText( context: this@MapsActivity, text: "Map style not found.", Toast.LENGTH_LONG).show()  
    }  
}
```

# LOCATION - PERMISSIONS

- ▶ Private function - enableMyLocation
- ▶ Check if permissions are granted by the user
- ▶ If true, enabled location, else, request permission

```
private fun isPermissionGranted(): Boolean {  
    return ContextCompat.checkSelfPermission(  
        context: this@MapsActivity,  
        Manifest.permission.ACCESS_FINE_LOCATION  
    ) == PackageManager.PERMISSION_GRANTED  
}
```

# LOCATION - PERMISSIONS

- ▶ Private function - isPermissionGranted
- ▶ Check if location permission has been allowed/denied by the user
  - ▶ Returns a boolean value

```
override fun onRequestPermissionsResult(  
    requestCode: Int,  
    permissions: Array<String>,  
    grantResults: IntArray  
) {  
    if (requestCode == REQUEST_LOCATION_PERMISSION) {  
        if (grantResults.contains(PackageManager.PERMISSION_GRANTED)) {  
            enableMyLocation()  
        }  
    }  
}
```



# LOCATION - PERMISSIONS

- ▶ Private function - requestPermissions
- ▶ ArrayList of permissions - we declare the permissions in AndroidManifest.xml
- ▶ REQUEST\_CODE\_PERMISSION = 1
  - ▶ Create a companion object or constant
  - ▶ Used to check user action for corresponding request
  - ▶ User will be prompt to allow or deny these permissions

```
private fun enableMyLocation() {  
    if (isPermissionGranted()) {  
        map.isMyLocationEnabled = true  
    } else {  
        ActivityCompat.requestPermissions(  
            activity: this@MapsActivity,  
            arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),  
            REQUEST_LOCATION_PERMISSION  
        )  
    }  
}
```

# LOCATION - MAPSDATA

- ▶ Create a class called MapData
- ▶ MapData object contains a name, latitude/longitude & drawable

```
 class MapData (  
    var name: String,  
    var location: LatLng,  
    var drawable: Int  
 )
```



# PRACTICAL

- ▶ Series of tasks covering today's lecture
- ▶ Worth 3% of your final mark for the Design and Development of Applications for Mobile Devices course
- ▶ Deadline: Friday, 12 June at 5pm