5COSC023W - MOBILE APPLICATION DEVELOPMENT
Lecture 11: Maps, Location and Runtime Permissions

Dr Dimitris C. Dracopoulos

Get the Last Known Location

Steps (set up):

1. For Google Play Services add in the **dependencies** section in the `build.gradle` module file (adjust versions for the latest):

```
implementation("com.google.android.gms:play-services-location:21.3.0")
// Android Maps Compose composables for the Maps SDK for Android
implementation("com.google.maps.android:maps-compose:4.3.3")
// Maps SDK for Android
implementation("com.google.android.gms:play-services-maps:19.1.0")

// Optionally, you can include the Compose utils library for Clustering,
// Street View metadata checks, etc.
implementation("com.google.maps.android:maps-compose-utils:4.3.3")

// Optionally, you can include the widgets library for ScaleBar, etc.
implementation("com.google.maps.android:maps-compose-widgets:4.3.3")
```

The following link describes all the Google Play services that you could include in an Android app:

https://developers.google.com/android/guides/setup

Get the Last Known Location (cont'd)

2. Add the `ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION` permissions in the manifest file of the project:

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

3. Obtain an Google Maps API (for usage with Maps) (follow the instructions in `https://developers.google.com/maps/documentation/android-sdk/get-api-key`).
4. Add the following inside the manifest file (if using the maps), inside the `application` tag but outside your activities:

```
<meta-data
          android:name="com.google.android.geo.API_KEY"
          android:value="YOUR_API_KEY" />
```

   replacing `YOUR_API_KEY` with your own API key that you got from the previous step.

Get the Last Known Location (cont'd)

Steps (Kotlin code):

1. Check if the permission is granted by the user, otherwise request the permission by calling `ActivityCompat.requestPermissions`.
2. Implement in your activity the `onRequestPemissionsResult()` callback method which will receive the permissions result.
3. Create a `FusedLocationProviderClient` object:

```
mFusedLocationClient =
        LocationServices.getFusedLocationProviderClient(this);
```

4. Call `getLastLocation()` (or access the `lastLocation` property) on the `FusedLocationProviderClient` object returning a `Task` object. Alternativaly call `getCurrentLocation()` (last location might be out of date)
5. Call `addOnSuccessListener()` method on the task and pass it an object which implements the `OnSuccessListener<Location>` interface.

How to Receive Location Updates

Steps (Kotlin code):

1. Create a `LocationRequest` object containing the requirements of the request (update frequency, accuracy).
2. Create a `LocationCallback` as part of the activity and override its `onLocationResult()` method which is called periodically with the location updates.
3. Call `requestLocationUpdates()` on the `FusedLocationProviderClient` object and pass it the `LocationRequest` and the `LocationCallBack` objects.

The Location and Maps Code

The Kotlin code for the maps and location application developed in the lecture can be found in the following link (you need to add the additional stuff in the manifest and Gradle build files as described in the lecture slides):

`https://ddracopo.github.io/DOCUM/courses/5cosc023w/location_maps_composable_app.zip`

The Location and Maps Application

The `MainActivity.kt`:

```
package uk.ac.westminster.mapscomposableapp

import android.content.Intent
import android.content.pm.PackageManager
import android.location.Address
import android.location.Geocoder
import android.location.Location
import android.os.Bundle
import android.os.Looper
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.Column
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationCallback
import com.google.android.gms.location.LocationRequest
import com.google.android.gms.location.LocationResult
import com.google.android.gms.location.LocationServices
import com.google.android.gms.location.Priority
import java.util.Locale
```

The Location and Maps Application (cont'd)

```
class MainActivity : ComponentActivity() {
    lateinit var fusedLocationProviderClient: FusedLocationProviderClient

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GUI()
        }

        fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this)
    }
```

The Location and Maps Application (cont'd)

```
fun getLocation(locationCallback: LocationCallback) {
    if (ContextCompat.checkSelfPermission(
            this,
            android.Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED
    ) {
        // if not granted, request the permission
        ActivityCompat.requestPermissions(
            this,
            arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),
            1
        )
    } else {  // permission has been granted
        var locationRequest =
            LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY, 10000L).build()
        fusedLocationProviderClient.requestLocationUpdates(
            locationRequest,
            locationCallback,
            Looper.getMainLooper()
        )

        // Example of how to get the last location now (without any periodic updates)
        fusedLocationProviderClient.lastLocation.addOnSuccessListener { location ->
            lastLocationToString(location)
        }
    }
}
```

The Location and Maps Application (cont'd)

```
// callback function which is called when the user grants or
// denies a runtime permission
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
   super.onRequestPermissionsResult(requestCode, permissions, grantResults)
   if (requestCode == 1 && grantResults.size > 0
               && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // permission is granted do something
   }
   else {
       /* permission was denied by the user -
          request permission again or display a message to the user /
   }
}
```

The Location and Maps Application (cont'd)

```
fun lastLocationToString(lastLocation: Location?): String {
    var tvLocationString = ""


    if (lastLocation != null) {
        var address: Address? = null

        var geocoder: Geocoder = Geocoder(this, Locale.getDefault())
        var addresses: List<Address> =
            geocoder.getFromLocation(lastLocation!!.latitude, lastLocation!!.longitude, 1)
                                as List<Address>

        if (addresses != null && addresses.size > 0) {
            address = addresses[0]
        }

        tvLocationString = "Latitude: " + lastLocation!!.latitude + ", Longitude: " +
                            lastLocation!!.longitude
        if (address != null)
            tvLocationString += address.toString()
    }
    else
        tvLocationString = "Location is not available"

    return tvLocationString
}
```

The Location and Maps Application (cont'd)

```
@Composable
    fun GUI() {
        var locationString by remember{ mutableStateOf("") }
        var currentLocation by remember{ mutableStateOf<Location?>(null) }

        var locationCallback: LocationCallback = remember{object : LocationCallback() {
            override fun onLocationResult(p0: LocationResult) {
                super.onLocationResult(p0)
                currentLocation = p0.lastLocation
                locationString = lastLocationToString(currentLocation)
            }
        }}

        Column {
            Text(text=locationString)
            Button(onClick = {
                getLocation(locationCallback)}
            ) {
                Text("Get Location")
            }
            Button(onClick = {
                showMap(currentLocation) }
            ) {
                Text("Show Map")
            }
        }
    }

    fun showMap(location: Location?) {
        var i = Intent(this, MapsActivity::class.java)
        i.putExtra("latitude", location?.latitude)
        i.putExtra("longitude", location?.longitude)
        startActivity(i)
    }
}
```

The Maps Activity

File `MapsActivity.kt`:

```
package uk.ac.westminster.mapscomposableapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import com.google.android.gms.maps.model.CameraPosition
import com.google.android.gms.maps.model.LatLng
import com.google.maps.android.compose.GoogleMap
import com.google.maps.android.compose.Marker
import com.google.maps.android.compose.MarkerState
import com.google.maps.android.compose.rememberCameraPositionState
```

The Maps Activity (cont'd)

```
class MapsActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            GUI()
        }
    }

    @Composable
    fun GUI() {
        val location = LatLng(
            intent.getDoubleExtra("latitude", 0.0),
            intent.getDoubleExtra("longitude", 0.0)
        )
        val cameraPositionState = rememberCameraPositionState {
            position = CameraPosition.fromLatLngZoom(location, 10f)
        }
        GoogleMap(
            modifier = Modifier.fillMaxSize(),
            cameraPositionState = cameraPositionState
        ) {
            Marker(
                state = MarkerState(position = location),
                title = "Current Location",
                snippet = "Marker in Current location"
            )
        }
    }
}
```

Kotlin Applications Developed in this Module

1. Lottery app - Predict what the next National Lottery numbers will be.
2. Lost Dogs - Notify owners by email for their lost dog based on recognising the dog image.
3. Identify the dog breed based on random dog images.
4. The Tic Tac Toe Game (the Computer player attacks and defends in a logical manner)
5. User management system in a database.
6. The Book Finder app (retrieve details of a book from Internet)
7. Shopping management (add products and calculate their total cost by adding them to a database)
8. Quiz app - displaying multiple choice questions to the user with "True" or "False" alternatives.
9. Display the current and last location of a user in a map.
10. Cocktails app (display recipe and picture of a cocktail by searching the Internet).
11. The Memory game - Highlighting squares as green in a grid for a few seconds and challenge the user to recall the hidden squares.

Coursework apps:

1. Dice game - Human competes vs the Computer in a dice game.
2. Movies knowledge application using Web services and Databases. The User is searching for movies and actors/actresses performed in a movie.