The University Of Sheffield.

COM4510/6510
Software Development for Mobile Devices

Lecture 9: Contextualising user Actions (Part 1)

Dr Po Yang
The University of Sheffield
po.yang@sheffield.ac.uk

Lecture Overview

- Part 1:
  - Contextualising
  - Geofencing
- Part 2:
  - Activity Recognition
  - Combining AR and Geofencing
- Lab tutorial :
  - Preparing assignments

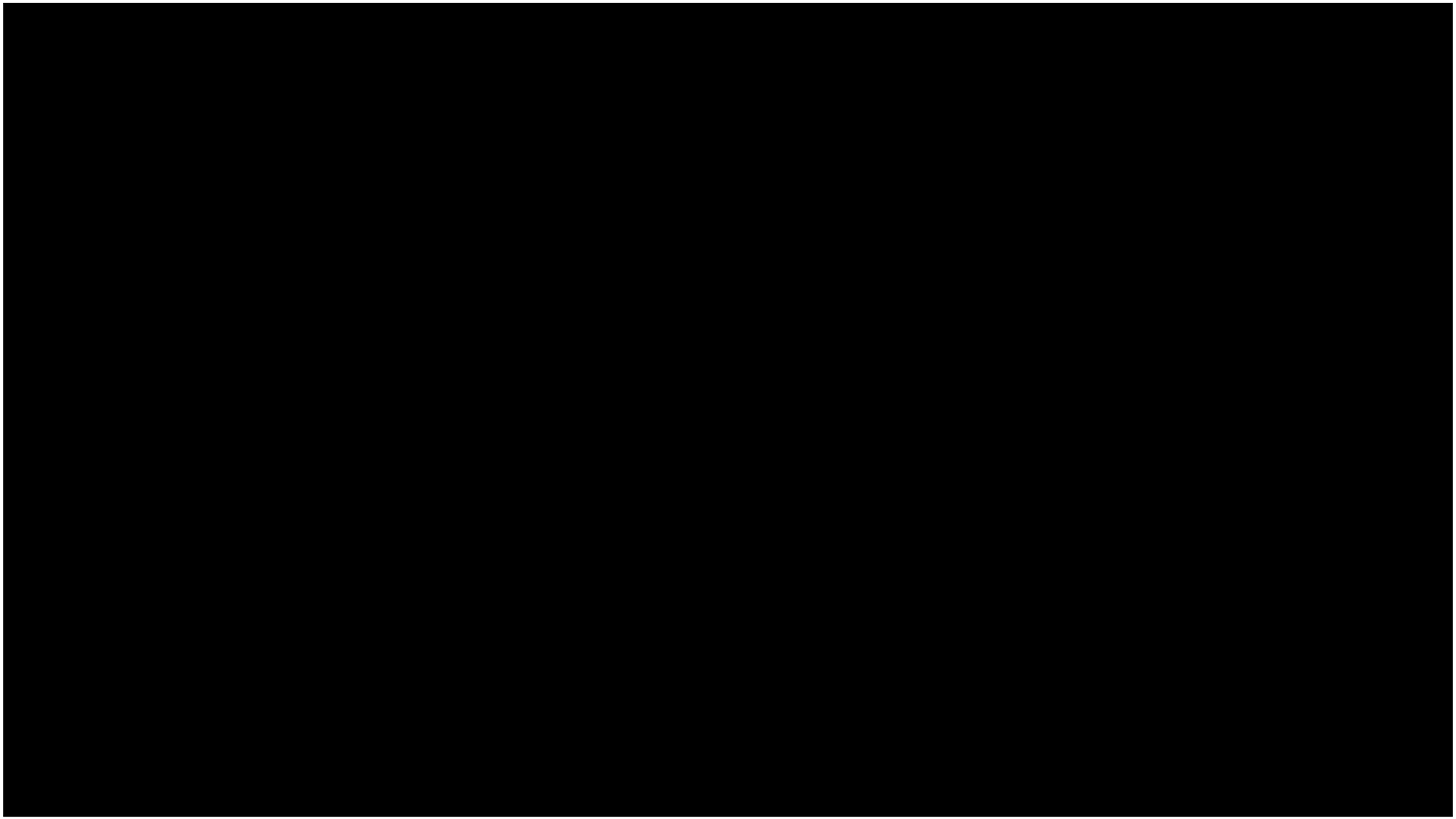
IMPORTANT NOTE

- GIVEN THE RESTRICTIONS ON THE USE OF GOOGLE MAPS
  - it is required that the package is declared in the Android Developer console for every application you develop
    - even for debugging
- So, in your assignment you can either use the package
  - uk.ac.shef.oak.com4510
  - uk.ac.shef.oak.com6510
  - or in case of emergency you can ask me to add your own package (please do not - there are too many of you!)

Contextualising

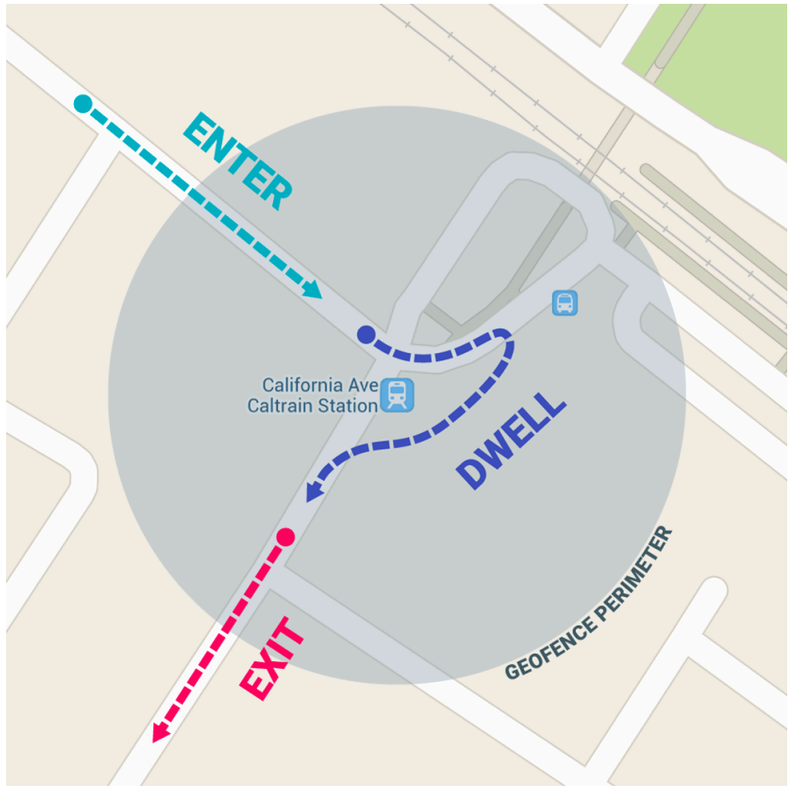
- Phones are with us **24/7**
- The context influences the services we can provide, e.g.:
  - users must not receive telephone calls when we are driving unless using a hands free device
  - users may receive specific services based on specific locations
    - e.g. task reminders when in the appropriate area

Contextual computing



What is important here is the idea, not the actual product (which failed from a commercial point of view btw)

The University Of Sheffield.



Geofencing

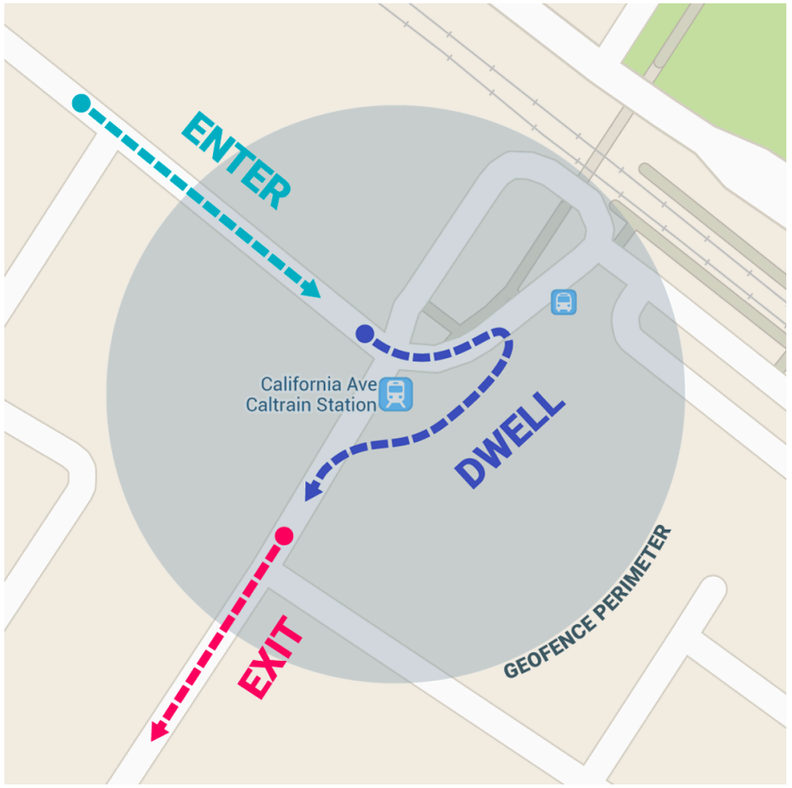
Context based on locations

Geofences

https://developer.android.com/training/location/geofencing

- Geofencing combines awareness of the user's current location
  - with awareness of the user's proximity to locations that may be of interest
  - specified via latitude and longitude
- A **radius** is added to allow for imprecision in detection
  - or simply to support proximity rather than matching
- The **latitude, longitude, and radius** define a geofence, creating a circular area, or fence, around the location of interest

- You can have multiple active geofences, Max 100 per user
- For each **geofence** **Location Services** can send entrance and exit events
  - dwell time before event triggering can be specified
  - duration of any geofence can be specified in milliseconds (if useful)
- Note: geofences are fine with battery power - they balance precision for a progressive use of power



Requesting permissions

- You will need:
  - Add the permissions to the Android manifest.

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

  - Create a method that checks for permissions.

```
@TargetApi(29)
private fun foregroundAndBackgroundLocationPermissionApproved(): Boolean {
    val foregroundLocationApproved = (
            PackageManager.PERMISSION_GRANTED ==
            ActivityCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION))
    val backgroundPermissionApproved =
        if (runningQOrLater) {
            PackageManager.PERMISSION_GRANTED ==
            ActivityCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_BACKGROUND_LOCATION
            )
        } else {
            true
        }
    return foregroundLocationApproved && backgroundPermissionApproved
}
```

Requesting permissions

- You will need:
  - Request those permissions by calling that method.

```
@TargetApi(29 )
private fun requestForegroundAndBackgroundLocationPermissions() {
    if (foregroundAndBackgroundLocationPermissionApproved())
        return
    var permissionsArray = arrayOf(Manifest.permission.ACCESS_FINE_LOCATION)
    val resultCode = when {
        runningQOrLater -> {
            permissionsArray += Manifest.permission.ACCESS_BACKGROUND_LOCATION
            REQUEST_FOREGROUND_AND_BACKGROUND_PERMISSION_RESULT_CODE
        }
        else -> REQUEST_FOREGROUND_ONLY_PERMISSIONS_REQUEST_CODE
    }
    Log.d(TAG, "Request foreground only location permission")
    ActivityCompat.requestPermissions(
        this@HuntMainActivity,
        permissionsArray,
        resultCode
    )
}
```

Requesting permissions

- You will need:
  - Handle the result of asking the user for the permissions.

```
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String>,
    grantResults: IntArray
) {
    Log.d(TAG, "onRequestPermissionResult")

    if (
        grantResults.isEmpty() ||
        grantResults[LOCATION_PERMISSION_INDEX] == PackageManager.PERMISSION_DENIED ||
        (requestCode == REQUEST_FOREGROUND_AND_BACKGROUND_PERMISSION_RESULT_CODE &&
                grantResults[BACKGROUND_LOCATION_PERMISSION_INDEX] ==
                PackageManager.PERMISSION_DENIED))
    {
        Snackbar.make(
            binding.activityMapsMain,
            R.string.permission_denied_explanation,
            Snackbar.LENGTH_INDEFINITE
        )
            .setAction(R.string.settings) {
                startActivity(Intent().apply {
                    action = Settings.ACTION_APPLICATION_DETAILS_SETTINGS
                    data = Uri.fromParts("package", BuildConfig.APPLICATION_ID, null)
                    flags = Intent.FLAG_ACTIVITY_NEW_TASK
                })
            }.show()
    } else {
        checkDeviceLocationSettingsAndStartGeofence()
    }
}
```

Checking device location

- add code to check that a user has their device location enabled, and if not, display an activity where they can turn it on using a location request.

```
private fun checkDeviceLocationSettingsAndStartGeofence(resolve:Boolean = true) {
    val locationRequest = LocationRequest.create().apply {
        priority = LocationRequest.PRIORITY_LOW_POWER
    }
    val builder = LocationSettingsRequest.Builder().addLocationRequest(locationRequest)
    val settingsClient = LocationServices.getSettingsClient(this)
    val locationSettingsResponseTask =
        settingsClient.checkLocationSettings(builder.build())
    locationSettingsResponseTask.addOnFailureListener { exception ->
        if (exception is ResolvableApiException && resolve){
            try {
                exception.startResolutionForResult(this@HuntMainActivity,
                    REQUEST_TURN_DEVICE_LOCATION_ON)
            } catch (sendEx: IntentSender.SendIntentException) {
                Log.d(TAG, "Error getting location settings resolution: " + sendEx.message
            }
        } else {
            Snackbar.make(
                binding.activityMapsMain,
                R.string.location_required_error, Snackbar.LENGTH_INDEFINITE
            ).setAction(android.R.string.ok) {
                checkDeviceLocationSettingsAndStartGeofence()
            }.show()
        }
    }
    locationSettingsResponseTask.addOnCompleteListener {
        if ( it.isSuccessful ) {
            addGeofenceForClue()
        }
    }
}
```

Adding geofences

- Create a Pending Intent

```
private val geofencePendingIntent: PendingIntent by lazy {
    val intent = Intent(this, GeofenceBroadcastReceiver::class.java)
    intent.action = ACTION_GEOFENCE_EVENT
    PendingIntent.getBroadcast(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT)
}
```

- Add a Geofencing Client

```
geofencingClient = LocationServices.getGeofencingClient(this)
```

PendingIntent

- A PendingIntent is a token that you give to a foreign application
  - e.g. NotificationManager , AlarmManager , Home Screen AppWidgetManager , or other 3rd party applications
  - to allow the foreign application to use your application's permissions to execute a predefined piece of code that you provide

PendingIntent

- By giving a PendingIntent to another application, you are granting it the right to perform the operation you have specified as if the other application was yourself
  - with the same permissions and identity
- Be careful about how you build the PendingIntent:
  - almost always, for example, the base Intent you supply should have the component name explicitly set to one of your own components, to ensure it is ultimately sent there and nowhere else

Creating a request

- GEOFENCE_TRANSITION_ENTER triggers when a device enters a geofence,
- GEOFENCE_TRANSITION_EXIT triggers when a device exits a geofence
- INITIAL_TRIGGER_ENTER tells Location services that GEOFENCE_TRANSITION_ENTER should be triggered if the device is already inside the geofence
- INITIAL_TRIGGER_DWELL triggers events only when the user stops for a defined duration within a geofence
    - reduce "alert spam" resulting from large numbers notifications when a device briefly enters and exits geofences

Adding geofences

- Add geofences

```
private fun addGeofenceForClue() {
    if (viewModel.geofenceIsActive()) return
    val currentGeofenceIndex = viewModel.nextGeofenceIndex()
    if(currentGeofenceIndex >= GeofencingConstants.NUM_LANDMARKS) {
        removeGeofences()
        viewModel.geofenceActivated()
        return
    }
    val currentGeofenceData = GeofencingConstants.LANDMARK_DATA[currentGeofenceIndex]

    val geofence = Geofence.Builder()
        .setRequestId(currentGeofenceData.id)
        .setCircularRegion(currentGeofenceData.latLong.latitude,
            currentGeofenceData.latLong.longitude,
            GeofencingConstants.GEOFENCE_RADIUS_IN_METERS
        )
        .setExpirationDuration(GeofencingConstants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
        .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER)
        .build()

    val geofencingRequest = GeofencingRequest.Builder()
        .setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER)
        .addGeofence(geofence)
        .build()

    geofencingClient.removeGeofences(geofencePendingIntent)?.run {
        addOnCompleteListener {
            geofencingClient.addGeofences(geofencingRequest, geofencePendingIntent)?.run {...}
        }
    }
}
```

Removing Geofences

- When you no longer need geofences, it is a best practice to remove them, which stops monitoring, in order to save battery and CPU cycles.

```
private fun removeGeofences() {
    if (!foregroundAndBackgroundLocationPermissionApproved()) {
        return
    }
    geofencingClient.removeGeofences(geofencePendingIntent)?.run {
        addOnSuccessListener {
            Log.d(TAG, getString(R.string.geofences_removed))
            Toast.makeText(applicationContext, R.string.geofences_remo
                .show()
        }
        addOnFailureListener {
            Log.d(TAG, getString(R.string.geofences_not_removed))
        }
    }
}
```

Best practices

- Reduce power consumption
  - Set the notification responsiveness to a higher value to avoid false alarms
  - Use a larger geofence radius
    - for locations where a user spends a significant amount of time, such as home or work
      - to avoid false enter/exit events
    - for indoor locations (esp tall buildings)
- Choose the optimal radius for your geofence
  - 100m suggested in city - some km in the middle of nowhere

The University Of Sheffield.

Activity Recognition

Context based on Actions

A/R

https://developer.android.com/guide/topics/location/transitions

- It might be necessary to design your app to identify when a user **starts or stops a particular activity**
  - e.g., biking, or driving
- Why?
  - e.g. a mileage tracking app could start tracking miles when a user starts driving,
  - a messaging app could mute all conversations until the user stops driving
- The **Activity Recognition Transition API** detects changes in the user's activity
  - Apps subscribe to transitions in activities of interest and the API notifies your app only when needed

How to

- Permissions

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.myapp">

  <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
  …
</manifest>
```

- You must implement the following:
  - An **ActivityTransitionRequest** object that specifies the type of activity and transition
  - A **PendingIntent** callback where your app receives notifications

Activities

- Constants in the **DetectedActivity** class
- The **Transition API** supports the following activities:
    - IN_VEHICLE
    - ON_BICYCLE
    - RUNNING
    - STILL
    - WALKING
- **A transition type of ACTIVITY_TRANSITION_ENTER or ACTIVITY_TRANSITION_EXIT. For more information, refer to the ActivityTransition class.**

ActivityTransitionRequest

- To create the ActivityTransitionRequest object, you must create
  - a list ofActivityTransition objects, which represent the transition that you want to receive notifications about
    - An ActivityTransition object includes the following data:
      - An activity type (e.g. Walking)
      - A transition type: ACTIVITY_TRANSITION_ENTER or ACTIVITY_TRANSITION_EXIT

Example

```
val transitions = mutableListOf<ActivityTransition>()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.IN_VEHICLE)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_ENTER)
          .build()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.IN_VEHICLE)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_EXIT)
          .build()

transitions +=
        ActivityTransition.Builder()
          .setActivityType(DetectedActivity.WALKING)
          .setActivityTransition(ActivityTransition.ACTIVITY_TRANSITION_EXIT)
          .build()
```

Then create a request

```
val request = ActivityTransitionRequest(transitions)
```

- Create a Task and pass an intent
- then define what to do if the request fails/ succeed

```
// myPendingIntent is the instance of PendingIntent where the app receives callbacks.
val task = ActivityRecognition.getClient(context)
        .requestActivityTransitionUpdates(request, myPendingIntent)

task.addOnSuccessListener {
    // Handle success
}

task.addOnFailureListener { e: Exception ->
    // Handle error
}
```

Process activity transition events

- When the requested activity transition occurs, you app receives an Intent callback. An ActivityTransitionResult object can be extracted from the Intent, which includes a list of ActivityTransitionEvent objects.
- The events are ordered in chronological order, for example, if an app requests for the IN_VEHICLE activity type on the ACTIVITY_TRANSITION_ENTER and ACTIVITY_TRANSITION_EXIT transitions, then it receives an ActivityTransitionEvent object when the user starts driving, and another one when the user transitions to any other activity.
- You can implement your callback by creating a subclass of BroadcastReceiver and implementing the onReceive() method to get the list of activity transition events.

```
override fun onReceive(context: Context, intent: Intent) {
    if (ActivityTransitionResult.hasResult(intent)) {
        val result = ActivityTransitionResult.extractResult(intent)!!
        for (event in result.transitionEvents) {
            // chronological sequence of events....
        }
    }
}
```

Deregister for activity transition updates

- You can deregister for activity transition updates by calling the `removeActivityTransitionUpdates()` method of the `ActivityRecognitionClient` and passing your `PendingIntent` object as a parameter:

```
// myPendingIntent is the instance of PendingIntent where the app receives callbacks.
val task = ActivityRecognition.getClient(context)
        .removeActivityTransitionUpdates(myPendingIntent)

task.addOnSuccessListener {
    myPendingIntent.cancel()
}

task.addOnFailureListener { e: Exception ->
    Log.e("MYCOMPONENT", e.message)
}
```

The University Of Sheffield.

Combining the two

Applications

- With **A/R and geofencing** you can create cool apps that for example
  - Tracking detailed locations only in some contexts, e.g.
    - MOTOR INSURANCE: tracking locations only when driving
  - HOW?
    - Set up A/R in your app
    - Create a background service tracking the locations using HIGH_ACCURACY
    - start A/R
      - when the transition IN_VEHICLE ENTER is detected, start tracking locations
      - when the transition IN_VEHICLE EXIT is detected, stop tracking locations

- Tracking location of employees on **large sites** (e.g. steelworks)
  - This is allowed for some specific cases only (e.g. safety)
    - You must track only when on site, not when they are at home
- How?
  - Set a geofence around the site
  - When the user enters the site, start tracking locations
- Note it only works for very large sites as location tracking not very good indoors

- Tracking your employees only when they are driving for work
- How?
  - set a geofence around your company site
  - set A/R on the app
  - when the user is driving and has just left the geofence, start tracking

- Create a tourist's app
- When the user has **entered your site and is no longer driving**, provide suggestions and informations
- How?
  - set a geofence around your site
  - detect transition IN_VEHICLE and Exit
- When user is on site and has stopped driving, provide the information

The University Of Sheffield.

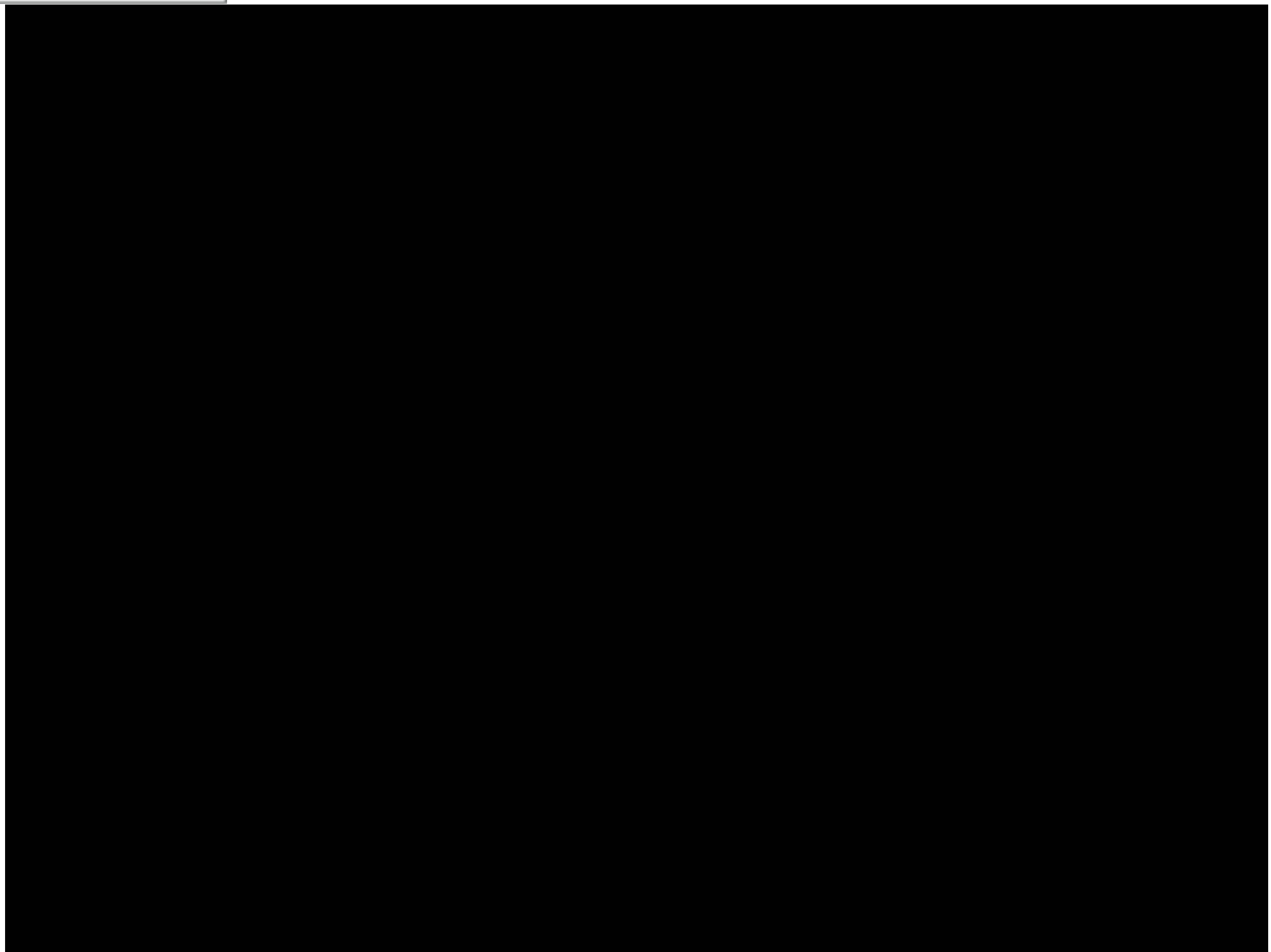
No longer office machines

The tip of the iceberg

- Phones are not computers
  - They have far better capabilities and possibilities
  - Only issue is battery power
- It has been a long journey from the office machines

The University Of Sheffield.

No longer office machines



In my Eyes

https://www.ted.com/talks/sergey_brin_why_google_glass?language=en#t-47041

In my home

amazon echo

amazon.com/echo

The University Of Sheffield.

Testing

The University Of Sheffield.

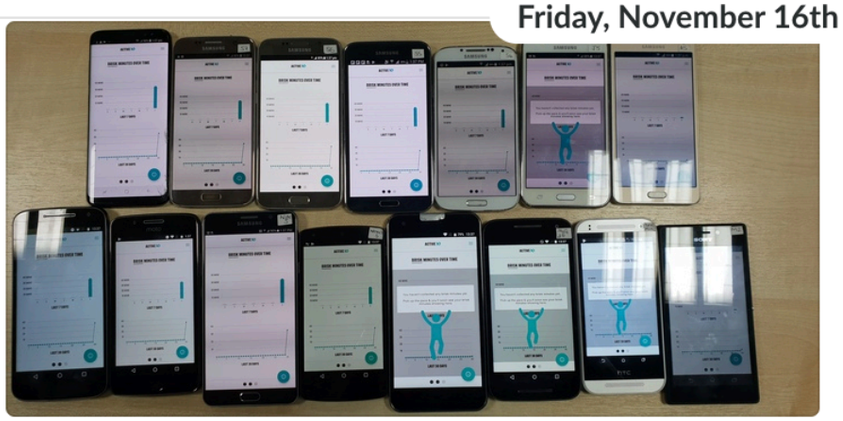
- Complex
- Difficult
- Lengthy
- Requires
  - Junit testing
  - Testing on real devices
  - Tests on different Android Version x devices
  - Independent testers

The University Of Sheffield.

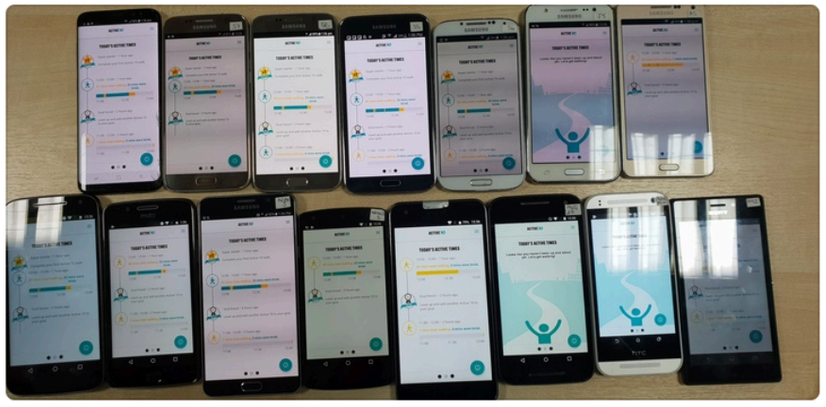
#general

☆ | 15 | 0 | Company-wide announcements and

Friday, November 16th

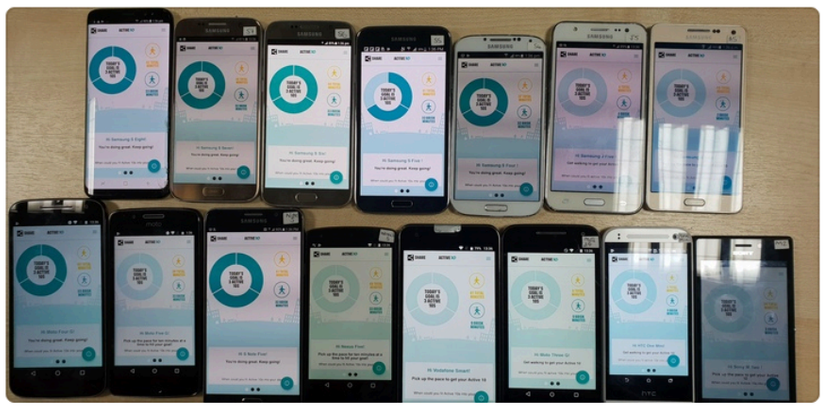


20181116_133643.jpg



let me send you the Android versions. Give me a minute

20181116_133617.jpg



Siri 2:18 PM

Samsung Galaxy S4 Jelly Bean Android 4.4.2
Samsung Galaxy S5 Lollipop Android 5.0
Samsung Galaxy S6 Nougat Android 7.0
Samsung Galaxy S7 Oreo Android 8.0
Samsung Galaxy S8 Oreo Android 8.0

Summary

- Part 1:
  - Contextualising
  - Geofencing
- Part 2:
  - Activity Recognition
  - Combining AR and Geofencing
- Lab tutorial :
  - Preparing assignments