Mobile Programming

Ian Batten
igb@batten.eu.org

Today’s Content

- Exercise Feedback: all **excellent**
- Discussion of next exercise
- A bit of positioning theory, for the exercise
- Handlers

Exercise 1

- Everyone's code worked first time, which is brilliant
- All bar one worked correctly on a Nexus
  - Problem was trivial font issue, and it ran correctly in Emulator.
- I ended up using your binaries as I had problems compiling some exercises: need to discuss what has to be delivered (one was missing `build.xml`, for example). I used everyone's binaries to be fair, and I didn't mark down.

Exercise 1

- I am very happy to do one-to-one feedback and discussions
- I’d rather not do 6pm tonight
- But mail me and book a slot.

Exercise 2

- Questions…?
- Do we want to have a session in the lab next week?

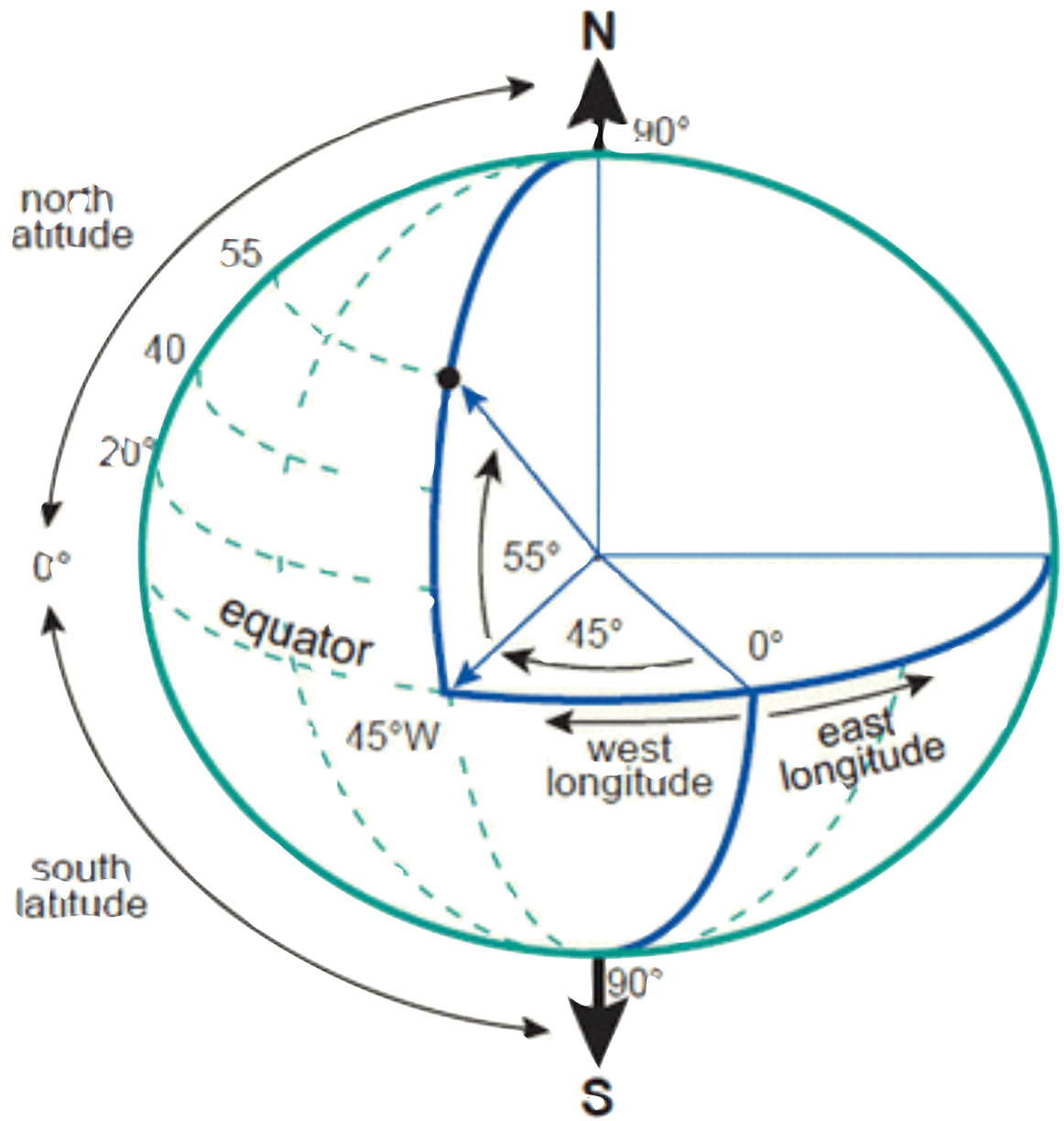
Basic Architecture

- An application which knows about locations, and sends those locations to a service, which…
- …tracks the phone's location, and continuously compares it with the locations and…
- …wakes the application up (or re-runs it, or something) when you approach a location.
- Service or IntentService, BroadcastReceiver…

If you can’t manage this…

- Then do it with threads in some other way. I want to see running code that works, and we can worry about the implementation details later.

Positioning



Sign conventions

- By convention, North and East are positive, South and West are negative.
- London is roughly 51.5N 0W (the carpark of the O2 Arena, Greenwich).
- Birmingham is roughly 52.5N 2W, or 52.5,-2 (a point near the Titford Canal, which is the highest point on the Birmingham canal network at 511’)
- So an application that works in the UK needs to cope with crossing the zero meridian of longitude.
- 2W is the centreline of the National Grid projection used for UK maps.

Angles

- Angles can be expressed in degrees, minutes and seconds of an arc (ie 60ths and 3600ths)
  - 52° 30’ 30”
- or as decimal degrees:
  - 52 + (30 / 60) + (30 / 3600) = 52.508333333°
- or in ludicrous GPS format:
  - 5230.5 = 5° 30.5’

Distances

- Reasonable approximation over short to medium distances is “Haversine formula”.
  - Assumes earth is a sphere, which is untrue
  - Requires “locally sensible” approximation of radius of earth; 6367km reasonable value for England, 6373km reasonable value for eastern USA.
  - Requires all angles to be in radians (1 radian = 180/π = 57.2957795°)
    - π = 4.arctan(1) gives you an accurate π in your math context.

Haversine

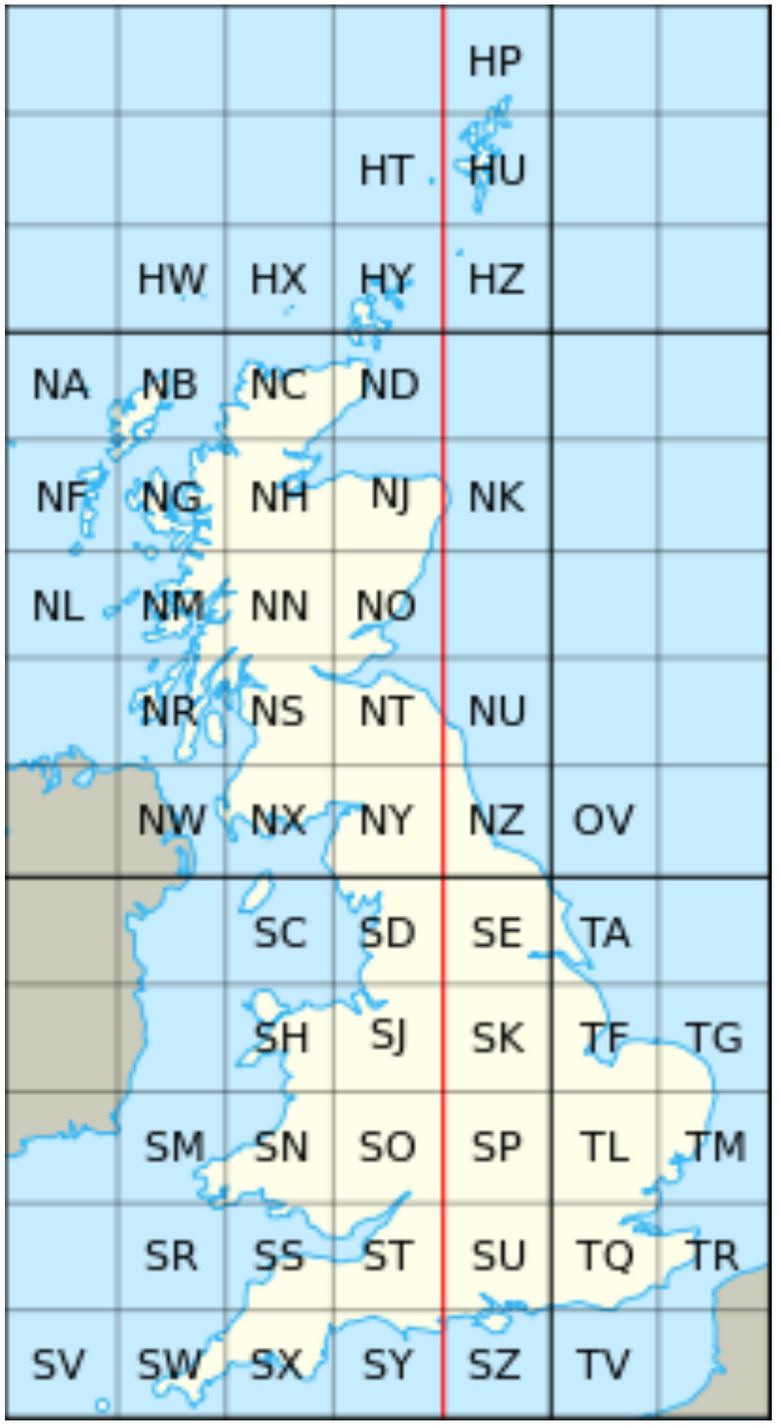
```
dlon = lon2 - lon1
dlat = lat2 - lat1
a = (sin(dlat/2))^2 + cos(lat1) *
     cos(lat2) * (sin(dlon/2))^2
c = 2 * atan2( sqrt(a), sqrt(1-a) )
d = R * c (where R is the radius of the Earth)
```

Test your code

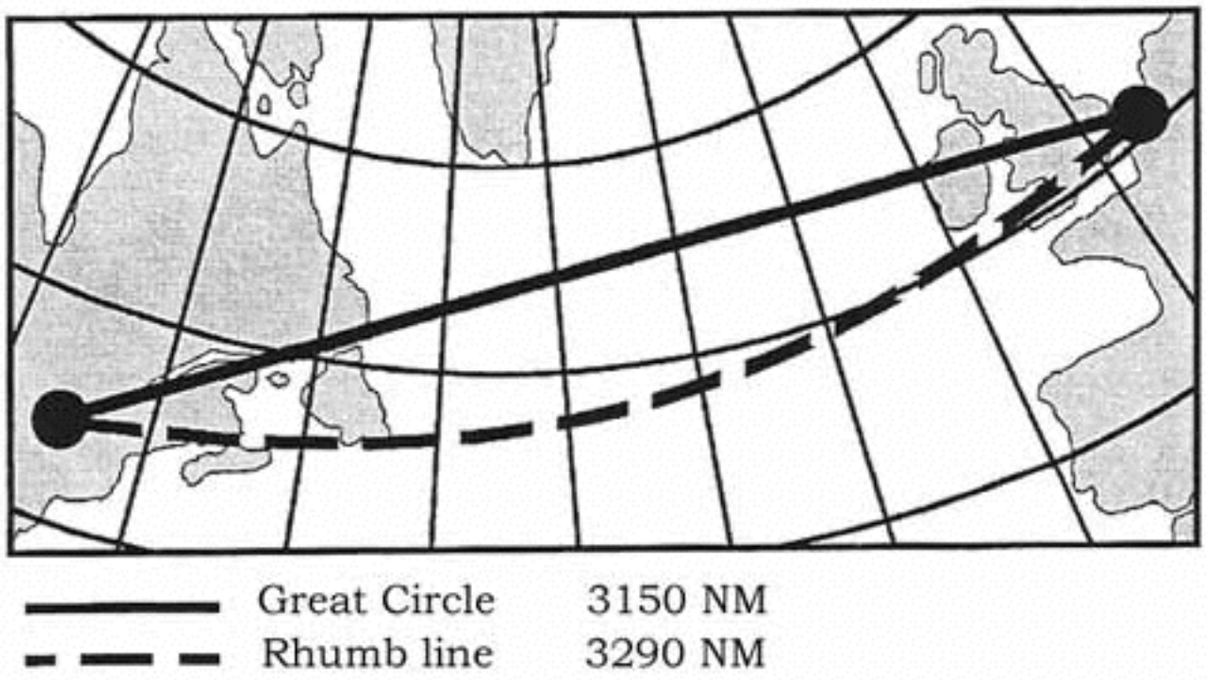
- 52° 27’ 1.3446” N, 1° 55’ 49.3763” W
- 52° 25.0320’N 1° 57.5047’W
- 3.7km
  - Note two different formats!

This is all rather hard…

- This is why most countries use some sort of Mercator (or more commonly transverse Mercator) which allows you to use Pythagoras calculations on simple (x, y) co-ordinates
  - Point1 = SP 029 799
  - 402900m north, 279900m east from a point 400km W and 100km N of 49N 2W.
  - Point2 = SP 048 836
  - 404800, 283600
  - distance = 100*sqrt(19*19+37*37)=4160m.



- Note huge difference in distance, 3.7km v 4.1km
- Partly there are unit conversion problems, but also:
  - Central scale factor of OSGB maps
  - Straight line on Mercator is a Rhumb Line, not a Great Circle
  - Assumptions of spheres different
  - etc.



But for our purposes

- Close enough is close enough!
- Over distances of 10m or 20m, it doesn’t matter.

Handlers

- Basis for AsyncThread and friends
- Allows a piece of code or a message to be passed between threads

```
public class MainActivity extends Activity {

   private Handler handler;
   private ProgressBar progressBar;

   /** Called when the activity is first created. */

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      handler = new Handler();
      progressBar = (ProgressBar) findViewById(R.id.progressBar1);

   }

   public void startProgress(View view) {

      new Thread(new Task()).start();
   }

   // Class Task in next slide

}
```

```
class Task implements Runnable {
    @Override
    public void run() {
      for (int i = 0; i <= 20; i++) {
        final int value = i;
        try {
          Thread.sleep(1000);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        handler.post(new Runnable() {
          @Override
          public void run() {
            progressBar.setProgress(value);
          }
        });
      }
    }
}
```

A closure, sort of

Messages

- Override “handleMessage (Message)” in the Handler
- Get an empty message with hander.obtainMessage()
- Get the bundle with message.getData ();
- Update the Bundle
  - bundle.putString (“key”, “value”);
- Send it with handler.sendMessage (message)
- handleMessage method is called in other thread

```java
public class MainActivity extends Activity {

    private Thread workingthread = null;

    //message handler of the main UI thread
    //the handler will be passed once the background thread is created
    //and it will be triggered once a message is received
    final Handler mHandler = new Handler(){

      public void handleMessage(Message msg) {
        Bundle b;
        if(msg.what==1){

            b=msg.getData();

            //log the data received
            Log.d("data key 1", String.valueOf(b.getInt("k1")));
            Log.d("data key 2", String.valueOf(b.getInt("k2")));
            Log.d("data key 3", String.valueOf(b.getInt("k3")));

        }
        super.handleMessage(msg);
      }
    };

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        workingthread=new JobThread();
        workingthread.start(mHandler);
    }

}
```

```java
public class JobThread extends Thread{

    private Handler hd;

    public JobThread(msgHandler){
        //constructor
        //store a reference of the message handler
        hd = msgHandler;
    }
    public void run() {
        //do some work here

        //create the bundle
        Bundle b = new Bundle(4);

        //add integer data to the bundle, everyone with a key
        b.putInt("key1", 4);
        b.putInt("key2", 7);
        b.putInt("key3", 91);

        //create a message from the message handler to send it back to the main UI
        Message msg = hd.obtainMessage();

        //specify the type of message
        msg.what = 1;

        //attach the bundle to the message
        msg.setData(b);

        //send the message back to main UI thread
        hd.sendMessage(msg);

    }
}
```

Intent

- You can start Activities, too.

```
Intent intent = new Intent();
intent.setClass (this, Other_Activity.class);
intent.putExtra ("EXTRA_ID", "SOME DATAS");
startActivity (intent);
```

- Then in other Activity

```
@Override
protected void onCreate (Bundle savedInstanceState) {
  super.onCreate (savedInstanceState);
  Bundle extras = getIntent ().getExtras ();
  if (extras != null) {
   String data = extras.getString ("EXTRA_ID");
   if (data != null) {
        // do stuff
   }
}
```