**Practical**

This week's lecture looked atthe process of adding lines and polygons as an overlay to a Google Map.  The first part of this week's practical is simply to re-create the examples given in the lecture. Using the files listed for the polygon examples, you should try to re-create the examples shown. The first two of these examples, you will recall, are very simple, and create a pair of square polygons. Try altering the characteristics of these polygons. You might also try adding in some extra points.

The third example is more complex; it uses (simplified) ward boundary data to create a number of polygons for wards in Leeds.  In places the boundaries are a little crude (compared to the originals) and as the polygons have been generalised independently of each other, they do not always meet seamlessly.  
  
**Task**Copy the files above to your workspace on www.personal.leeds.ac.uk, and get the first three examples working properly.

**An example with more polygons**

We'll now look at a case with more polygons than example 3 above.  This data file:

* [mapdata\_leeds\_eg2.js](https://vlebb.leeds.ac.uk/bbcswebdav/users/geo6odw/geog58705871_2010-11/wk8/wk8_files/mapdata_leeds_eg2.js)

contains polygon data for all wards in Leeds.  In terms of GIS data it is a small data set, but for a Google Maps application it is relatively large and detailed.  The level of performance with a large data set depends on the client capabilities (i.e. the user's browser and computer) rather than on any server capabilities.  However, this is not a fixed constraint: over time the average computer becomes more powerful, and at the same time newer versions of browser software bring better performance.

**Task**Using the html and mapsetup files from example 3 above, generate a new example using the **mapdata\_leeds\_eg2.js** data file.

* What is the performance like on your browser?
* Do you think that a data file of this size is suitable for general use?

All wards will be shaded the same colour using this sample data.  Can you think of ways of experimenting with alternate shading?  You could download and append some attribute data, or use an alternative approach such as a random number generator. A random number can be generated in JavaScript using Math.random() (e.g. var randNo = Math.random())