Optimal Map Layer Ordering
LayerOrder.mbx Version 1.0, 19 July 2002

Background

When a set of MapInfo tables are opened with the *Current Mapper* option, MapInfo will create a map with layers ordered *points, lines, regions* with *text* possibly placed before regions. It seems this rule applies to layers with mixed content as well. A layer with points and regions will be grouped with points. A layer with regions and lines will be grouped with lines.

Similarly when layers are added to an existing mapper, the above rules apply, except that a text only layer is added above points.

This default layer ordering provides a good start to an optimal display order, but does not account for other properties of the table or of the style override which impact the display order.

Definition

For the purposes of this application, the *optimal display order* is defined as the order which will minimise the number of objects that are hidden beneath others.

As a simple example, a parks or a lakes layer with solid fill brush style should be positioned on top of a states layer. A suburb boundary layer with no fill or transparent brush style would be positioned on top of a layer with a solid fill style.

The *optimal display order* is not necessarily the optimal order for cartographic output.

Layers may be ordered for other purposes, such as drill-down or "point object within boundary" type of analysis. This might typically be the reverse of the display order.

Because of the sampling method (see next section) , data variability (especially mixed layer object type content) and other criteria that a user may apply in setting layer order, the *optimal display order* is of course not necessarily the best for particular user circumstances. It may well be closer than the default MapInfo order though.

Method

LayerOrder builds a classification table in a form where a sorted SQL select can be used to build the layer order. The sort order is *Object Type, Override Style, Object Style, Contiguous Region Parameter, Average Object Size Parameter*. The *table name* and *table path* are also stored to provide a reliable index for locating a specific table details.

Keywords are used for each entry under these column headings with a prepended sort character. For example, when sorting by the column that classifies a region table as to whether its objects are contiguous or non-contiguous, the keywords are *bCONTIG* and *aNON-CONTIG* respectively. This means that sparse (non-contiguous) region objects will be sorted ahead of contiguous region objects for this particular sort criterion.

The build process uses a sampling method to extract a few hundred records from each table. Therefore it may miss some object types, but provides performance benefits for large tables where a comprehensive group process is considerably slower. An option may be provided in a future revision to perform the comprehensive process.

Note: From a philosophical standpoint, the table classification information is probably better stored as metadata than the *lmTableClass* table. The single reason for not doing this is the read-only table. Until MapInfo has a central writable repository of table metadata, only the table publisher can write this information.

Classifying Tables

LayerOrder > Build Table Class for Open Tables

This function will take all open tables and add them to the *lmTableClass* catalogue table. The attributes that are computed for each table are:

Classification	Column Name	Description
Object type	ObjectType	Value of aTEXT, bPOINT, cLINE, dREGION or combination
Override style	OrideStyle	Blank for table classification. Used in map layer reordering to store the actual override style for particular layer objects. This will be sorted ahead of the native object style for the table. The values and meaning are the same as *Object Style*.
Object Style	ObjStyle	The *style* is actually the portion of style relating to visibility: aNOFILL, bTRANSPARENT, cFILL For lines, the style is: aINVIS, b*penwidth* So for regions, a transparent fill style will appear before a solid fill style. For lines, an invisible line style will appear before a style of pen width 5 units. [Nb: Some applications may require thicker line styles to appear above thinner line styles – this may be provided as a future option].
Contiguous Region Parameter	GeomContig	For regions only: aNON-CONTIG, bCONTIGUOUS as described previously
Average Object Size Parameter		For regions: object area in sq km. For other objects: object count This attribute is most significant for regions where all other things being equal, smaller size objects should appear above larger size objects. The best example is the single blue rectangle that often sits underneath everything else to provide a water background. This parameter forces this layer to be below a country layer. For other object types, a more subjective approach where a smaller object count is taken to be more significant (higher in the order) than a larger object count.

If a table already exists in *lmTableClass*, its attributes will be updated.

Mapping Some Tables

LayerOrder > Map Selected Tables

If records in the lmTableClass or lmLayerOrder are selected, these will be mapped in a single mapper in the optimal display order.

Reordering Layers in an existing Mapper

LayerOrder > Reorder Layers in Current Map

This will attempt to reorder layers based on information previously built. If a table hasn't been classified, it will be reported and its position will remain unchanged.

Build Usage Suggestion

The build process doesn't take long (2 secs for the *average* size table), but it is convenient to build the catalogue for all your current tables in one step. To achieve this:

1. Use Windows Explorer, file find to locate all TABs on your MapInfo data drives, in turn.
2. Select all of the files and drag into a MapInfo Professional session.
3. *Tools > LayerOrder > Build Table Class for Open Tables*

Note that query tables will be classified according to their base table path.

System Issues

LayerOrder.mbx has been compiled for MapInfo 5.5 or later. It will function on earlier versions if the MBX is patched appropriately.

Provision of Source Code

This tool was originally developed as a part of the *LayMan (v2)* (www.acenet.com.au/~pwaight) application to provide a map and layer reorder facility. Please feel free to incorporate this tool into your applications.

Source coded is provided to encourage those with sufficient interest to extend the capabilities of the program based on additional classification information. I request that source code changes be emailed to me at pwaight@acenet.com.au to allow incorporation with future versions.

Raster tables should be capable of being handled using the existing classifications but is not in the current version. A function to build a single table might be useful. See the source code for other documented limitations.

End Of Document