MapInfo Internal Precision :

potentials, practices and shortcomings

Jacques Paris, February-April 1996

updated for the MI 4.02 release

The capability of MapInfo for high precision mapping is not particularly well documented and users who would like to control the level of precision of their maps are too often unaware of the possibilities it offers. I have collected in this paper different informations relative to the definition of internal precision as implemented in MI, to different ways for setting the desired precision level, and to different limitations that cool the hopes for higher performance.

In order to understand that issue, I have started by mailing questions to Mapinfo-l. Later on, I switched to direct exchanges with various resource persons; I want to thank them and specially Joe Schwartz of MapInfo who has so patiently contributed to my education.

I am fully responsible for this material and my understanding of the collected information. If I explicitly quote from generally public messages, that does not modify the original value of these messages. As this paper started with a great deal of collaboration from MI-l-ers, it can still benefit from them; I will welcome any comments, editorial or technical, that would contribute to its quality, or to the author's moral.

jacques@paris-pc-gis.com

Jacques Paris 611-5350 MacDonald ave.
Côte-St-Luc H3X 3V2 Québec Canada
514-489-9200

MI 4.02 update

The 4.02 maintenance release (Easter weekend 1996) contains some modifications affecting internal precision performance. Some are a plus (see end of §8) :

- x and y coordinates are given with the same number of decimal digits;
- the number of decimals varies within a certain range with the level of internal precision.

These are the most obvious changes towards improvement. But I also found some new, or perhaps better documented, behaviors that should be carefully checked, specially at very large mapper scales (see §10 -b, -c, -e).

1 - Types of maps and Bounds

When a table is created in MapInfo format, through an import of a MIF table or by creation of a new table, it is defined according to one of two main types : earth (with two sub-types, lat/lon and projected) and non-earth.

A map is always "bounded" by a pair of x-y mini maxi. The bounding can be implicit (no mention is made of the limits that are set automatically by MI to default values depending on some map specifications) or explicit (the bounds are spelled out and form an integral and visible part of the table specifications). Implicit (default option) and explicit boundings are possible with lat/lon and projected maps, while explicit bounding is required for non-earth.

The bounds are stringent limits : it is impossible to add to a map an object that would lie outside the bounds (its coordinates would be clipped to the bounds ot the table), and it is impossible to change the bounds when there have been set.

2 - Internal storage of coordinates

The original coordinates that define the objects making up a map are stored in long integer format, in their genuine units coded in integer "internal" units, after some linear transformations according to the situation:

lat/lon "unbounded" is a unique case : the coordinates are registered in millionths of a degree and the range of internal units are set to correspond to +-90 millions for latitude, and to +-360 millions for longitude

in all the other situations, (lat/lon bounded, projected bounded and unbounded, non-earth necessarily bounded), the minimum x bound is set to -1 billion in internal units and the maximum to +1 billion of these units; and the same for minimum y and maximum.y The genuine coordinates are converted by linear transformation using these parameters into internal units. The x and y transformations are independent, being based on the x and y ranges that can be different.

note : General transformation that applies to longitude (x-) and latitude (y-)

ex (external coordinate)
exmax, exmin (maxi, mini of external coordinate, the bounds)
ix (internal coordinate) = int [(ex - exmin) / (exmax - exmin) * 2 billion] - 1 billion

3 - Finding the bounds of a map

There are different ways to find out the bounds on a MI map. One is to examine the header of the MIF file that was used to create the map, if it exists. However, if implicit bounds have been used, they will not be visible. It would be necessary most times to create such a MIF file. There is no need to export the entire map; just select one object, save copy as ... xxx, export xxx; the bounds are general values that have nothing to do with the actual contents of the map, or the MBR (minimum bounding rectangle) of the objects in the table.

Another way is to use an MBX application called PROJTOOL produced by R.G.Edwards (MapTools Co. edwardsrg@aol.com); one of its functions yields directly the data related to a map definition.

In MapBasic4, the TableInfo() function can give this information (coordsys, bounds..). It can be obtained in the MapBasic Window by typing and running (press on <enter> after typing) :

The parameters of the linear transformations (exmax, exmin) are kept in double precision (15 significant digits) as are the projection parameters; the internal coordinate as a long integer.

PRINT TABLEINFO(“MY_TABLE”,NN)

Caps are optional, I use them to offset the command in the text. MY_TABLE is the name of the open table for which the information is requested. NN is one of the following numbers :

NN =	30	to obtain	name of the projection
	29		coordsys clause
	25		mini X
	26		mini Y
	27		maxi X
	28		maxi Y

All values can be obtained successively by replacing the previous code and pressing <enter>. Results from 25 to 28 are included in the result from 29, with the difference that they might not be in the same coordinate system; with 29, one gets the coordinate system of the original table; with 25-28, it is the system presently in use, by default LAT-LON. To obtain the “original”, one should run before the print statement, the following command :

SET COORDSYS TABLE “MY_TABLE”

4 - Internal precision and virtual grid

The transformation of genuine coordinates into integer values create a virtual grid into which actual data is inserted. One could easily imagine that the distance between two successive internal values expressed in original map units gives the size of the grid (by default, vertical and horizontal grid sizes may not be equal, except for unbounded lat/lon). This grid definition is used to define the internal precision of a given map.

Internal precision is of interest because it indicates the ability of the software to differentiate between points with different coordinates during input by assigning them different internal values, and during processing by keeping them separate and by showing their distinct coordinates. Original measures are rounded up to the closest integer value during transformation; "restituted" decimal coordinates look like the original only if the rounding of the decimal equivalent of the internal integer values reduces significantly the number of decimal digits. It is very possible that two points showing the same coordinates (e.g. in an object info window) are in fact distinct in the map internal representation. (see below for more on this subject)

5 - How to measure internal precision

There are two measures applied to the vertical and to the horizontal directions. Calculations are identical, and if the results are different, one should take the largest value that is the least favourable one for defining precision. The measures are expressed in the units of the original map, be they degrees or meters, or whatever.

The general expression is

(bound maxi - bound mini) / 2 billion

ex.1 an UTM map bounded in the x direction to 0 and 2 000 000 meters will have an internal x precision of 2 million / 2 billion meters, i.e. 1 millimeter. If its y bounds were 3 000 000 to 5 000 000 meters, the y precision will also be 1 millimeter.

ex.2 a lat/lon "unbounded" as a definition of 1 millionth of a degree; that gives 11.11 cms in the y direction and 11.11*cos(lat) in the x.

ex.3 a lat/lon bounded to 20 and 40 degrees North and -90 and -70 in longitude has a precision of 1/100 million of a degree, 100 times superior to the unbounded one.

6 - Setting bounds

There are several ways to set bounds, but none to change the bounds to an existing table .

The first one would be when creating a new table, as Joe Schwartz explained it recently on this list :
>... display the MapBasic window in MapInfo and create a mappable table called FOO. You
> should see this line appear in the MapBasic window :
**> Create Map for FOO Coordsys Earth Projection 1,0
> Let's say you want to set the boounds for this table to 2*2 degrees, ranging from 110-112
> degrees longitude and 41-43 degrees latitude. Just position the cursor at the end of the
> above line and change it to this :
> Create Map for FOO Coordsys Earth Projection 1,0 Bounds (110,41) (112,43)
> You should do this before adding any objects to FOO because it will delete all existing
> objects.

I would precise that the line shown **> can specify any other projection definition. I know of two ways to obtain it : when creating a new file and if no mapper is open, use the button "Projection" to select the good one before saving the new file, or if a mapper is open with one layer, the new map could be added to the mapper and will have the same "projection" as the mapper with the implicit bounds of that projection. Note that the mapper projection is defined by the first map that is added to the mapper. Even if other maps with different projections are added later on, and the first one is removed, the mapper projection remains that of the first one.

The last line of the above quote needs clarification : changing the bounds will clear only the obj column from the table; the other data columns are left intact. Of course with no object to map, the table has limited use. But if it contained only symbols, the x and y coordinates could be saved in two new columns before changing the bounds, and used for "creating points" with the new bounds.

A second technique is to introduce the proper bounds specifications in the header of a MIF file. As many format translators generate MIF files, adding the bounds (either through requests of the translator itself or with a text editor later) is very feasible. For an existing table, exporting it as a MIF file, changing (or adding) the bounds and reimporting it is also possible. In this case, the existing coordinates do not gain in precision; they stay at the level they were before and may even be slightly altered because of the various roundings, but manipulations and creations will benefit from the new precision.

Another approach is to use the PROJTOOL.MBX already mentionned; one of its functions can take care of the operations corresponding to the first technique using the MapBasic window.

There is also hope in another direction. From Joe Schwartz (January 12) :
"I would prefer to make this feature accessible from the user interface. For example, we could put a "Bounds..." button in the Choose Projection dialog to let you examine and change the coordinate system bounds"
And again (January 16) :
"I hope to add this capability to the MapInfo user interface in a future version".
[Not yet in 4.02]

7 - Practical uses of precision level setting

The most important advantage of setting the precision level is to be able to answer the demands of some professionals that are not satisfied with the only indication relative to precision of the 1/1 000 000 of a degree. It is indeed possible to attain precision levels well beyond the generally acceptable one millimeter.

The increased precision is generally obtained by a limitation of the scope of any map (see ex.1 above). A limitation to 2 000kms in the nord-south and east-west directions will insure the 1 millimiter precision. Note that the east-west limitation is not really a constraint in "segmented" projections (UTM width is 6 degrees) because the metric coordinates are never negative and do not exceed 1 000kms.

Another advantage of controlled bound setting is the garantee to work with a unique "basic grid"; If several projected maps from different origins and/or with different projections are piled/tiled in the same mapper, their basic grids can be in perfect alignment if similar and compatible bounds are chosen for each map. Several topological problems are avoided; for example, tracing in one layer using another as reference will give the new nodes exactly the same "internal" coordinates. - copying from one to the other will avoid any distorsions, - moving nodes to have them coincide in differents layers become possible whereas they may be preventing from taking the required position by non-coincidental grids..

The rules to follow are simple : for a given project, choose a precision level, translate it in a given value that leads preferably to "round" measures of precision (1mm, 0.5cm ...) and should be used for width and height, then apply the appropriate bounds to all the maps.

8 - Precautions with MIF files

If one wants to export/reimport in MIF format, one must make sure of the precison with which the coordinates are written down. The number of decimals must be such that they correspond to the level of precision of the map before exportation. In an unbounded UTM projection, the precision is 1cm less a micron in latitude (just over 10 000kms for 1 billion positions) and about double in the x direction. Because of possible roundings, if one wants the full garantee of not loosing information by giving the same coordinates to 2 distinct points, one would require coordinates posted in millimeters.

In the 3.05 version, the number of decimals were often excessive. Version 4 trunquates them, theoretically in relation to the precision level but not symetrically (more decimals for X than for Y). The symetry is restored in 4.02.

The upgrade makes also MI more responsive to a certain extent to internal precision level. For ex., in an MTM table without specific bounds, object location (with the object info window) and export coordinates are in cms (2 decimals with a metric standard); for a bound definition, they are in millimeters. However, there is no difference when the bounds width varies; at 2 000 000 meters (internal precision 1 mm) or at 200 000, there are only 3 decimal digits.

9 - Truncation of results as a limit to precision

The restitution of coordinates in native units that should be done using the complementary equation to that used for coding internal coordinates (see above), yields a floating number that is truncated to 10 significant digits, apparently because the internal coordinates have only 10 digits. This constraint is there now, even if one may dream of ways to recombine double precision parameters and long integer in a "longer" decimal number.

As a consequence, one should be aware that the precision level has limits. As an example, in a MTM or UTM projection, latitude cannot be given with more that the millimeter level (the US-Canada border is roughly 4 500 000 meters North, leaving three digits for millimeters). But for safety reason, one should also consider the roundings that are possibly done on the last digit and put his trust more on a lower level of precision, the cm in this case. But in non-earth tables, this constraint can be pushed back further. This is another example that precision considerations are specific to each work environment and that the user must understand the various parameters that determine the results.

10 - Working at the limits of the precision level

When zooming a mapper window to a small multiple of the internal precision, say 10 times, one can observe some interesting and some strange behaviors.

a/ If a layer is editable, adding symbols will reveal the **basic grid** (add two symbols in the same place, then drag one progressively further horizontally or vertically from the origin until it does not snap back on the first when released, then use that distance as a guide for adding more).

b/ You may wonder at not seeing the results of some editing operations : a **window redraw** is pratically required after each operation (addition or move) to see the results on the screen. *(This has yet to be recognized as a bug, but I think it is one)*

c/ The **tape measure** gives weird readings, bouncing by increments that have nothing to do with the precision level nor with pixel definition; it seems that it could be due to the truncation mentionned above. The 4.02 behavior is more easily documented :

> On a UTM table bounded to 2 million meters in both directions, with a zoom of 1 meter on a default size mapper, the tape measure gave the following readings (in cms) 9.49 13.43 16.44 18.99 21.23 23.25 25.12 .... with asymptotically decreasing “jumps”.
>
> The tape measure traces on the screen a “measured” path that is offset from the cursor position (in my experiments much more to the right than to the top) and the nodes of this path are not constrained by the basic grid. However, this may be due to the graphic card more than to MI because I have not been able to reproduce it on some other PCs.

d/ The **snap action** is useless in the sense that the snapping distance within the window becomes inferior to the basic grid size; there is a zoom at which the basic grid overtakes the

"set" snapping distance.

> estimate of snapping distance expressed in real world measurement :
> p is pixel size (ex.: .28 mm), S the scale factor for the mapper (ex.: 1/10),
> n number of pixels for snapping as set in preferences
> snapping distance = n * p / S (all expressed in the same unit)

e/ It is necessary to change the "**distance units**" used to post some mapping information (zoom or tape measure) to be able to read with the same accuracy than the one availbale in the window; truncation has also its impact here. But the change has not effect on cursor location nor on object info window, the number of decimals there being determined by the existence of bounds.

> In MI 4.02, one can observe a counterproductive effect of the change of distance units : with the setup in c/ above, using mm as distance units would round up the last digit whereas with cm it would be shown (above series in mm > 90 130 160 190 ...)

f/ Editing techniques are also limited to direct positionning and shaping of objects because the object info tool does not necessarily accept coordinates with the adequate number of digits.

11 - Conclusions

As it is (version 4 and even better 4.0.2), MapInfo is a very good tool for working at a high precision level. But the onus is on the user : he must know how to discover the practical limits of the program in various circumstances and how to set his work environment in order to obtain the desired precision level.

As an example, the program is now sufficiently performing so that the millimeter level can be easily attained and maintained with projected maps in the metric system.

Improvements beyond those identified as in the making are still desirable mainly in the area of mathematical handling and of coordinates/dimensions posting, in order to have homogeneous precision throughout all the functions of the program dealing with those aspects. One can also dream of an information tool that would give the necessary parameters related to precision level of a given table (existing or being created) or of a mapper.