Selecting objects using their characteristics

Jacques Paris

jacques@paris-pc-gis.com

September 2001

Table of contents

This document explores ways to use objects own characteristics in selections. “Own characteristics” exclude tabular data and limit this search to the nature of the objects, their geographical definitions, their contents and their styles. It is not intended to be a guide to SQL expression writing but to show how these special features of a GIS program can be used in a “where” expression.

After identifying the tools useful for extracting these characteristics, we will look into the operators and functions that can handle relative geographic relationships, and close with some examples of implementation.

> MapInfo does not allow comparing two objects directly and globally; an expression in a selection like “Object1=Object2” will not be accepted. Comparisons can only be made on specific elements of their definition. A possible difference between two objects can be more easily proven than a perfect identity. For details on that subject, see the “Identity.doc” document distributed with the SkimAll application (PPCC site, Services and Products).

> As the MI requesters share the same working space as the MapBasic window, the variables defined in the MBWindow are directly available to the requesters. All the “where” conditions presented in this document can be used indifferently in the SQL requester or as part of full Select statement in the MBWindow.

1 - Basic tools

1 – 1 Using the nature of the objects

We are dealing here essentially with object types that are limited. The following list gives the exact spelling of the names and their numerical code:

Arc	1
Ellipse	2
Line	3
Polyline	4
Point	5
Frame	6
Region	7
Rectangle	8
Rounded Rectangle	9
Text	10

From the MapBasic window or from a MI requester, object type can be determined in two different ways: by calling the ObjectInfo() function or by querying obj directly. In both cases, a comparison between a function that can take multiple forms (string, smallint, logical, float…) or obj that is a “form” in itself and another term (string or smallint) will not be accepted without having forced both terms of the comparison to be of the same type. The standard way to handle this is to use the Str$() function on the left side and to enclose the right part between double quote signs.

The general ways to write the alternative equalities are

Str$(ObjectInfo(obj,1))="2"

or

Str$(obj)="Rounded Rectangle"

1 – 2 Using the geographical definition of the objects

Geographical definition includes information of different nature. They include data on the structure of the object

- number of sections of a polyline or of polygons in a region. Using the formula

  Str$(ObjectInfo(obj,21))>"1"

  will select all multiple sections/polygons objects

- number of nodes in a polyline or in an object. If we use

  Str$(ObjectInfo(obj,20))<"200"

  we will get all the objects (polylines and polygons) with less than 200 nodes

on its overall size

- length of a linear object

  ObjectLen(obj, "m")>1000

  selects all the linear objects with a length over 1000 meters.

- length of the border of a closed object

  Perimeter(obj, "km")<1000

  picks up all regions with perimeters inferior to 1000 km.

- area of a closed object

  Area(obj, "sq mi")>0

  is a way to select all closed objects

on its absolute position by the coordinates of

- a point object

  ObjectGeography(obj,1)>484000 and ObjectGeography(obj,1)<486000

selects all points with longitude between 484000 and 486000 in the coordinate system of the table if its is the one that is in use (Set Coordsys Table … before Select…)

- the centroid of any type of object

  CentroidX(obj)>484000 and CentroidY(obj)<4735000

chooses all the points in the lower right quarter from 484000, 4735000. See note above about coordinates.

- the beginning and ending nodes of a line

  XBeg: ObjectGeography(obj,1) Ybeg: ObjectGeography(obj,2)
  Xend: ObjectGeography(obj,3) Yend: ObjectGeography(obj,4)

- the opposite corners of the MBR of any object

  Xmini: ObjectGeography(obj,1) Ymini: ObjectGeography(obj,2)
  Xmaxi: ObjectGeography(obj,3) YMaxi: ObjectGeography(obj,4)

1 – 3 Using the contents of the objects

The only type of objects that has some real contents is Text, and its contents are the text string itself. Selection can be made with

Str$(ObjectInfo(obj,3))=”Any text”

1 – 4 Using the style of the objects

Global style clause

The values returned by ObjectInfo() for style parameters are in the form of a style clause, such as “Pen(1,2,0)”. If the selection is made on a complete style, the second part of the expression must be also a complete style definition. To achieve this requirement, the reference can be a string variable or a style variable. One must make also sure that both sides of the expression are of the same type.

String variable in Mapasic window

```
Dim String_Var as string
String_Var=”Pen(1,3,0)”
```

Selection “where”

```
Str$(ObjectInfo(obj,2))=String_Var
```

Style variable in MapBasic window

```
Dim Style_Var as pen
Style_Var=MakePen(1,3,0)
```

Selection “where”

```
Str$(ObjectInfo(obj,2))=Str$(Style_Var)
```

Note: the string or the style variable can be “filled” in different ways, such as

```
St…_Var=currentpen()
St…_Var=objectinfo(selection.obj,2)
```

Single style element

Several single style elements can be used independently. We have to deal with the fact that the value of the argument of the function used to extract a style element may be different for the various types of styles. Let us recap first those parameters:

	Brush	Pen	Font	Symbol
Pattern	1	2		
Pointsize			3	3
Color		4		2
forecolor	2		4	
backcolor	3		5	
Font name			1	
… style			2	
Pen width		1		
… interleaved		6		
… index		5		

Symbol kind				7
… code				1
… font_name				5
… font_style				6
… angle				4
… custom_name				8
… custom_style				9

Single element selections must thus be made on sets of objects of the same type to avoid any possible confusion. It is not possible for example to select in one operation all the “red” objects in a table containing several types.

StyleAttr(objectinfo(obj,2),4)<> StyleAttr (currentpen(),4)

will pick up all the lines/plines with a color different from the current pen definition. If mixed types are present, it will select text objects with a different front color, will deal with regions on the basis of their border style (no value 4 for brush, but acceptable value for pen), mis-treat symbols by using their angle (value 4) not their color (value 2). To avoid any confusion, these expressions should be used in conjunction with a selection by object type. Selecting only lines and plines (no arcs) with a color different from the current pen would require the following expression:

(Str$(ObjectInfo(obj,1)=”3” or ObjectInfo(obj1,)=”4”) and
StyleAttr (objectinfo(obj,2),4)<> StyleAttr (currentpen(),4)

As the StyleAttr() function requires for input a style clause, the possibility of using directly a string variable that existed in the global style situation is not available in the single style element conditions. If the selection is done on the basis of a single value obtained from a given object, it is still possible to use the Mapbasic window to set up the proper variable. (Selection of regions with brush color the same as the selected object)

MapBasic window commands

```
Dim O as object
O = selection.obj
Dim I_color as integer
I_color=StyleAttr(O,2)
```

Select “where”

StyleAttr(objectinfo(obj,2),2)=I_color

> Note: the I_Color definition could be avoided by replacing the right part of the expression by StyleAttr(O,2). The choice is left to the user who must take into consideration the extra time required for processing that more complex expression; with the latest machines and with smallish tables, that impact may be negligeable.

2 – Geographic operators and two-objects functions

2 – 1 Operators available for a selection

These are the standard MI operators that take into account the relative geographic definitions of the objects. There are 3 families, Contains (Contains, Contains Part, Contains Entire), Within (Within, Partly Within, Entirely Within) and Intersects.

I will not elaborate on MI description and example of those key words. I will simply insist on some points.

From the basic definition of these operators, it is obvious that they cannot be applied to every kind of objects without limitations. Let us summarize the kind of operators logically useable in the general expression “obj *oper* objectA” (the second one being the reference) and which objects are selected.

Only situations that were found as producing a non-empty selection are detailed in the following table; we have found no way to fill a selection if the reference is a text object.

	Reference object		
Map object	Symbol	Linear object	Closed object
Symbol	I All symbols at same location	(*1)	C Part, W (*3), I All symbols located within ref.
Text	I All texts with MBR containing ref.	I All texts with MBR bisected by ref.	C Part, Partly W, I All texts with MBR bisected by ref. W All texts with MBR centroid within ref. Entirely W All texts entirely within ref.
Linear object	(*1)	I All objects bisected by ref.	C Part, Partly W, I All objects bisected by ref. W All objects with centroid within ref. Entirely W All objects entirely within ref.
Closed object	C (*2), I All objects containing the point	C Part, Partly W, I All objects bisected by ref. C All objects containing the centroid of ref. C Entire All objects containing entirely ref.	C All objects containing the centroid of ref. C Entire All objects containing entirely ref, C Part All objects that bisect ref. W All objects with centroid within ref. Entirely W All objects entirely within ref. Partly W, I All objects bisected by ref.

(*1) It may be possible that if the centroid of line/pline would be exactly centered on the reference point, some selection be possible. This eventuality has not been tested.
(*2) Variants (Contains Part, Contains Entire) are irrelevant but would have the same results
(*3) Variants (Partly Within, Entirely Within) are irrelevant but would have the same results

Note 1: an object that is entirely within another one is considered also to be bisected by it. E.g., "All texts with MBR bisected by ref." includes also all texts entirely within ref.

Note 2: two objects that have a single node in common are "partly within/contains part" and are considered as intersecting each other.

Note 3: Contains and Within are relative misnomers because in order for A to contain B, it suffices that the centroid of B be within A. Conversely A is within B if the centroid of A is within B.

2 – 2 The "second" object

One object in a comparison is the "column" obj, and we know that it cannot be it for the second one. We have seen that it must be defined outside the command containing such expressions. We will use the MapBasic window to define it. For example, one way of doing it is to select the object then do the following

MapBasic window commands

```
Dim O as object
O = selection.obj
```

Select "where"

```
Obj entirely within O
```

Select all the objects entirely within O, including O itself if it is in the table on which the selection is applied. Selected objects may have common node(s) with O, but they have no node outside O.

2 – 3 Two-objects geographic functions

There are several functions that require two objects as argument and return some information concerning their relative geographical structures. For example:

We have designated the objects O1 and O2; these names must be replaced by obj and O in the appropriate order as it applies.

Function	Return value	Typical use in selection
AreaOverlap(O1,O2)	Surface of the common area to both objects in current area units	Regions overlapping by a set amount
ProportionOverlap(O1,O2)	Proportion of area of overlap (O1,O2) relative to	Regions overlapping by a set proportion of O1.

	area of O1	
OverlayNodes(O1,O2)	Object with all the nodes of O1 plus nodes where O2 intersects O1	Used in conjunction with ObjectInfo(obj,20) to detect change in the number of nodes.
Combine(O1,O2)	Object as the union of O1 and O2	Used in conjunction with some other function (area, length..) to compare to set values, or with ObjectInfo(obj,21) to find if objects merged or remained separate sections/polygons.
Erase(O1,O2)	Object left when O1 is erased by O2	Used in conjunction with other functions such as area()
Overlap(O1,O2)	Object as the intersection of O1 and O2	Used in conjunction with other functions such as area()
Buffer(O,*resol,width,"unit"*)	Object as the buffer of O	Used in "proximity" selection with geographic operators

"*Used in conjunction with …*" means that the expression is made with embedded functions;

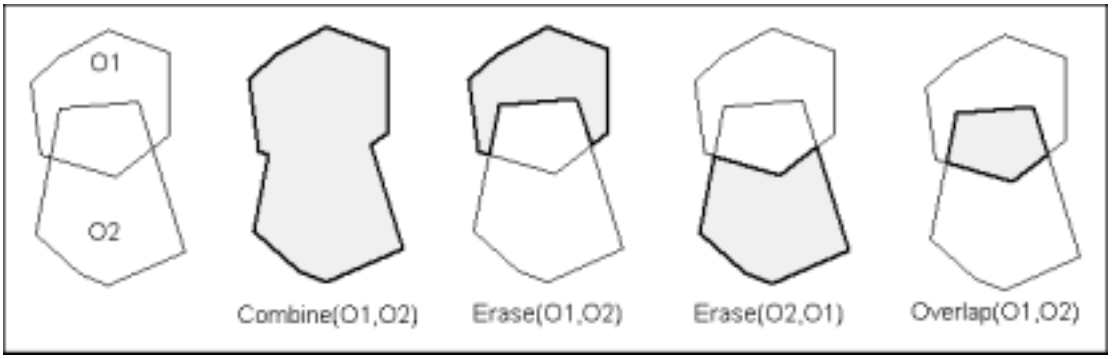
ObjectInfo(OverlayNodes(obj,O),21) – Objectinfo(obj,21)>1

selects the objects that intersect O at more than one place outside existing nodes

obj Intersects O and Area(erase(obj,O),"sq km")>50"

will select the regions that intersects O but that have at least 50 sq km outside O.

The relationships between Combine(), Overlap() and Erase() are best described by this diagram:



3 – Some examples

3 – 1 How to eliminate the “constant” object from the selection

If the object serving of reference is in the table on which the selection is applied, (e.g. by using in the MapBasic window O=selection.obj and making that variable the right side of the expression), it is included in the selection, and be sometimes the only selection. If we do not want it, then we must add an extra condition to exclude it. We can do that if we know the rowid of that object by adding to the where condition “ and rowid<>row_of_object”.

The MB script would thus be:

```
Dim O as object
Dim i_row as integer
O = selection.obj
i_row=commandinfo(2)
```

and the where expression:

```
Obj intersects O and rowid<>i_row
```

That condition creates a selection of all the objects intersecting O, excluding O itself

3 – 2 How to eliminate selected not-really-intersecting regions

If we do not want to include in a selection obtained by a “Contains Part”, “Partially Within” or “Intersects” operator those regions that touches to the reference objects but do not really overlap it, we can add to the “where” expression and extra condition that will check for eventual false overlaps. We can use the AreaOverlap() function for that.

Remember also that if object A is totally within object B, it is considered as intersected by B. Excluding “internal” objects would require an extra condition such as “and not obj entirely within O” and to be complete for sake of symmetry “and not obj contains entire O”. The following statement will only select those regions that overlap in part the reference object or are overlapped in part by it.

```
Obj intersects O and rowid<>i_row and AreaOverlap(obj,O)>0
and not obj entirely within O and not obj contains entire O
```

3 – 3 How to select all “red” objects in a mixed table

One must take into consideration that the color is defined in different “clauses” obtained differently for different types of objects and that color must be extracted with a special function from those clauses. To cover all eventualities, the where expression should contain all the following parts

Note 1 - We use “r_col” (integer) to ferry the “red” color rather that the rgb value for red to get a more general expression

```
Dim r_col as integer
```

r_col=255

Note 2 – Some objects have more than one color: regions have two, fill and border, sometimes 3 when the pattern has fore and back color; text may have two in some cases with fore and back color. We will use only the forecolor for regions and texts. If the others were required, extra parts should be added to the expression.

Note 3 – We have not included “Frame” because it is not found in a map.

where expression

```
Str$(obj)=”Arc” and StryleAttr(ObjectInfo(obj,2) or
Str$(obj)=”Ellipse” and StryleAttr(ObjectInfo(obj,3) or
Str$(obj)=”Line” and StryleAttr(ObjectInfo(obj,2) or
Str$(obj)=”Polyline” and StryleAttr(ObjectInfo(obj,2) or
Str$(obj)=”Point” and StryleAttr(ObjectInfo(obj,2) or
Str$(obj)=”Region” and StryleAttr(ObjectInfo(obj,3) or
Str$(obj)=”Rectangle” and StryleAttr(ObjectInfo(obj,3) or
Str$(obj)=”Rounded Rectangle” and StryleAttr(ObjectInfo(obj,3) or
Str$(obj)=”Text” and StryleAttr(ObjectInfo(obj,4)
= r_col
```

3 – 4 How to select all “paths” getting closer than a given distance from an object.

The “paths” are continuous polylines stored in table “paths”. The object is a region made into the variable object O from another table. We want to know all the paths that do not intersect the object but are getting closer than a given distance DistMini expressed in “distance units”

The “do not intersect” is translated by the negation of an intersection and the betting closer by an intersection with the region inflated (bufferd) by DistMini

```
not obj intersects O and obj intersects Buffer(O,20, DistMini, “distance unit”)
```