

# Natural Selection: Interactive subset creation

Graham J. Wills, Bell Laboratories (Lucent Technologies)

## *Abstract*

*Visualization is a critical technology for understanding complex, data-rich systems. Effective visualizations make important features of the data immediately recognizable and enable the user to discover interesting and useful results by highlighting patterns. A key element of such systems is the ability to interact with displays of data by selecting a subset for further investigation. This operation is needed for use in linked views systems, and in drill-down analysis. It is a common manipulation in many other systems. It is as ubiquitous as selecting icons in a desktop GUI. It is therefore surprising to note that little research has been done on how selection can be implemented. This paper addresses this omission, presenting a taxonomy for selection mechanisms and discussing the interactions between branches of the taxonomy.*

**Key words:** Brushing, Selection, Exploratory Data Analysis, Taxonomy, Visualization

## 1. Approach

Figure 1 shows a common operation within a GUI operating system. The user is dragging a “rubber band” around a set of icons representing files. By doing so the user is selecting them for a future operation, such as a copy, move or delete operation. The operation is natural; the user indicates which of a set of objects are of interest and performs an operation on them.

---

Graham J. Wills is a member of technical staff in the Software Production Research Department (Data Visualization group) of Bell Laboratories; Room 2F-323 Shuman Blvd, Naperville IL 60566; email: [gwills@research.bell-labs.com](mailto:gwills@research.bell-labs.com); web: [www.bell-labs.com/~gwills](http://www.bell-labs.com/~gwills). The author thanks the referees for their helpful comments.



**Figure 1. Selection from a set of Icons**

But even this ubiquitous selection operation is poorly understood by most users. Suppose the user wants to modify the selection in Figure 1 either by adding a file to the selected group or removing one. It would be tedious to start over and de-select everything, then select the revised set. Worse, with a rectangular rubber-band system, many selections are impossible (e.g. selecting '96vis' and 'AIPA96' in fig.1). Clearly the ability to modify the selection is a necessity. Under Windows '95™, the user can hold down either the shift or control key while dragging out the selection rectangle [9]. These modify the selection in the following manner. When the control key is held depressed while dragging, the state (selected or not) of the icons is toggled when dragged over. Alternatively, when the user clicks and drags over a set of icons with the shift key depressed the icons are added to the current selection.

Additionally, if the user shift-clicks on an icon without dragging, the system selects only those icons in a box between the previous icon clicked on and this one. This action is useful in a list of items but can be confusing in a two way layout of icons as in Figure 1.

Windows '95 attempts to provide the user with two alternative ways of modifying the selection, by toggling items with the control key and by adding them via the shift key. The issues faced in this comparatively simple task are amplified in a data visualization system, where hundreds or thousands of objects may be routinely selected, where selections in one view of the data require updating that selection in another view of the data, and where users often want to indicate irregularly shaped areas of the view. In the following section we present a structure for classifying selection mechanisms that aims at encompassing the majority of visualization systems.

## **2. Basic Taxonomy**

The basic model assumed in this paper is that of a system which displays objects in (possibly) a number of views and allows the user to indicate interest in subsets using a pointer-like interface. The scope of this paper includes GUI operating systems [2, 9], linked-windows environments [5, 7], interactive statistical graphics systems [8, 11, 12], geographic information systems (GISs) [6, 7, 10], drawing programs and any data visualization systems containing elements of drill-down and focusing.

We make one further assumption which limits our scope. We assume that when the user indicates interest in a subset, that the degree of interest is a binary variable - either interested or uninterested. Throughout the rest of this paper we assume that both an item's current state and the user's indication of interest is a mapping from the visualization domain to  $\{0,1\}$ . Although this restriction is true for the overwhelming majority of visualization systems, there are cases where a continuous degree of interest is appropriate. For example, indicating a position on a map leads to a natural possibility that objects located closer to the point might be of greater interest than those further away. We will not deal with this type of selection as examples of this type of system are few, and experience is needed to gain a better understanding of issues associated with it.

With that preamble, we move on to define a taxonomy of selection systems. The first branch in the taxonomy is the issue of *selection memory*:

- **Memory:** In a system with memory, each selection operation is remembered and changing one selection runs through all existing selection operations to determine the final selection. An example is Ahlberg and Shneiderman's (1994) implementation of dynamic queries in the FilmFinder. Here, each variable maintains its own selection state (for example,  $1990 > year > 1980$  or  $actor = Connery$ ) and the overall selection is defined as the intersection of the individual variable selections.
- **Memoryless:** In a memoryless system, no record is kept of other selections. Only the current state is kept track of, with no knowledge of how that state was achieved.

The advantage of a system with memory is that it allows the user to make a query involving a number of variables with very little effort, and is very forgiving; if you make an error in adjusting one selection, it is easy to change it. The advantage of a memoryless system is in power and adaptability. It is hard to generalize a memory-based system while retaining an intuitive interface (imagine the difficulties in coordinating selections from a map, a scatterplot and a network view, for example), and it makes choosing different selection operations very difficult as

the method of combination of the different selections is typically fixed. From observation of existing systems, it appears that the memory-based approach is particularly good for directed queries and answering the question “Which objects satisfy this description?” and the memoryless approach is better for discovering structure in the data and answering the question “Are there unusual features in this data?”.

A selection system can also use a variety of selection tools to identify a subset. A *selection tool* is a mechanism which allows the user to differentiate an area of a visualization and thereby select data items which correspond to that differentiated area. There are many different types of tools – a complete enumeration is an impossible task, but we can classify selection tools firstly by the method they use to differentiate an area, and secondly by their relationship to the data.

*Area differentiation:*

- **Brushes:** With brushing tools, a pre-defined region (typically a rectangle) is dragged around the view, with a selection operation being performed each time the brush is dragged.
- **Lassos:** A lasso (often a rectangle “rubber-band”, polygon or freely defined region) is defined by drawing it onto the plot or indicating endpoints. When the shape is completed, the selection operation is performed.

*Data dependency:*

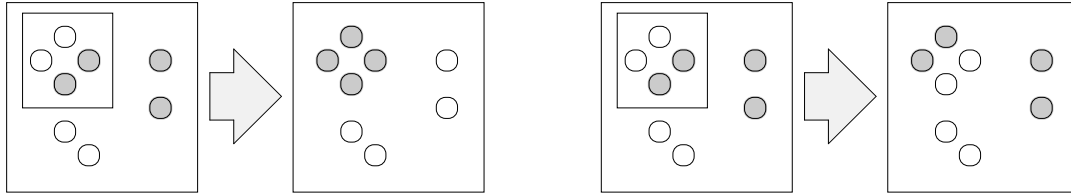
- **Data dependent:** When using the tool on a view of data, the area being indicated is modified by the data being displayed.
- **Data independent:** The area indicated by the tool is not affected by the underlying data in the view.

An example of a data dependent tool might be a brush for a geographic view that selects areas of the map, but the brush is constrained so that its center coincides with the center of the nearest town on the map. Another example would be a scatterplot brush, the radius of which alters so as always to encompass the same number of points.

The final element in the taxonomy is the selection operation. How does the system combine one set of states with another? In practice, as we saw with the example of Windows ’95 <sup>TM</sup> in section 1, a number of operations are available. In the following section we will examine the possible selection operations and build up a picture of reasonable *selection operation systems*. Note that although our discussion in the following section is based on the memoryless system, the operations described are equally applicable within a memory-based system.

### 3. Selection Calculus

In this section our goal is to study operations such as those displayed in Figure 2, where two alternative selection operations are demonstrated. The first choice replaces the current selection with the new one, the second performs an exclusive-or of the selection states. Our goal is to examine the value of all possible operations, and of all possible systems of operations, to see which are most appropriate for a visualization system.



**Figure 2. Two possible selection operations**

Since each item is either currently selected or not, and is either indicated by the user using a selection tool or is not, and the result of the operation is also binary, then a selection operator is a map of the form  $T: \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ , as shown in Table 1. With each  $T(.,.)$  having two possible results, it is clear that there are  $2^4 = 16$  possible operations. A selection system could consist of any subset of these operations, giving us  $2^{16} = 65,536$  possible systems. However, not all of these are useful, as we shall show.

S	I	S'
0	0	T(0,0)
0	1	T(0,1)
1	0	T(1,0)
1	1	T(1,1)

**Table 1. Combining a selection (S) and an indicated subset (I)**

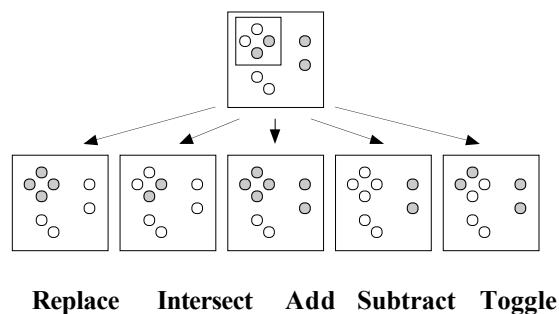
First, we will cut down the number of basic operations we consider. It seems unreasonable to consider an operation  $T: (0,0) \rightarrow 1$ , as such an operation corresponds to selecting something that was previously unselected and which the user did not indicate. Such operations run counter to the idea of focusing attention on interesting subsets

and we reject them<sup>1</sup>. This leaves us with the possible operations displayed in Table 2.

S	I	A	B	C	D	E	F	G	H
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

**Table 2. Eight candidate operations**

Of these, we eliminate **A**, which simply de-selects everything. A menu item or key stroke is more appropriate for this operation. Furthermore, several of the other operations can duplicate its effects simply by indicating a null set of items. **B** is the *intersect* operation (or logical AND). It is very useful for focusing down on subsets and is retained. **C**, the *subtract* operation, and **H**, the *add* operation, are especially common in brushing. Both are retained. **D** leaves the selection unchanged, ignoring the indicated set, and is therefore rejected. **E** selects only those items that were indicated, but previously unselected. This operation is similar to the *toggle* operation **G**, but is less intuitive and so we reject it, but retain **G**, which is a common operation in GUI and drawing programs. **F** is the common *replace* operation, and it too is retained. Figure 3 summarizes the set of retained selection operations.



**Figure 3. Basic Selection Operations**

Having now cut down to 5 operations, we have only 32 systems to consider. In evaluating them we use the three following criteria. Selection systems should be:

<sup>1</sup> The “Select All” operation, which sets the state of every item to be selected, is a useful operation, but as it does not require a subset indication, it is better implemented as a keystroke or menu item.

- **Simple.** A system with fewer operations is better, everything else being equal.
- **Powerful.** Users should be able to select any subset of items. The easier and faster it is for them to do this, the better the system.
- **Forgiving.** Selecting the right subset can take a little time. Making a selection that is only slightly wrong should be easy to rectify. An ‘undo’ key is not a panacea for this problem.

Consider the simplest set of systems; those having only one operation. The intersect and subtract operations only reduce the number of selected items and are therefore not very powerful<sup>2</sup>. Replace will only be powerful enough to select any given subset for certain types of tool and data, and add, although powerful enough, is unforgiving. Selecting one item too many means the user must start over. The toggle operation, however, is both powerful and forgiving, allowing items to be added or removed from the selection at will. If there is a ‘de-select all’ menu operation or keystroke, it subsumes the functionality of the replace operation, the most common default operation within icon-based GUI operating systems. The toggle operation is the only generally practical one-operation system.

The next simplest system has two operations. There are ten possibilities:

- |                        |                         |
|------------------------|-------------------------|
| (a) Replace, Add       | (f) Intersect, Subtract |
| (b) Replace, Intersect | (g) Intersect, Toggle   |
| (c) Replace, Subtract  | (h) Add, Subtract       |
| (d) Replace, Toggle    | (i) Add, Toggle         |
| (e) Intersect, Add     | (j) Subtract, Toggle    |

We reject (a) for being unforgiving as we cannot remove a few points from the selection. (b) and (c) are unforgiving for the opposite reason; points cannot be added to the selection. (f) lacks the power to select anything and is also rejected. The remaining six systems are plausible candidates, but a few have been widely implemented,

---

<sup>2</sup> It could be argued that the user might select all the items via a menu choice or keystroke and then reduce the number of selected items using these operations, but this is both tedious and unintuitive. It is more natural to say “this is interesting” than “everything except this is boring”.

whereas others have not been used. One reason might be that certain operations are simpler to understand (such as *replace*) and are therefore widely used. Others, such as *intersect*, require a more sophisticated understanding of the process and are used less frequently. Another reason is that certain operations are most natural for certain common occurring situations. The most common situation is for the user to identify a subset from a set of items all of which are currently unselected. *Replace*, *add* and *toggle* are the obvious winners here.

With these criteria in mind, we have ordered the systems in Table 3 with the potentially most useful listed first. Table 3 also orders the operations within a system showing which should be the primary (default) operation. Typically this would be implemented by making this operation the operation which occurs when selecting with the primary mouse button without holding down any modifier keys. The secondary operation might use a different mouse button (if available) or require holding down a modifier key.

	<b>Primary</b>	<b>Secondary</b>	<b>Notes</b>
d	Replace	Toggle	Drawing package standard. Mac OS method.
h	Add	Subtract	Scatterplot brushing 'paint' mode
e	Add	Intersect	Boolean OR and AND operations.
g	Toggle	Intersect	Similar to (e), but not as powerful
i	Add	Toggle	Similar to (d),(h), but less intuitive than either
j	Toggle	Subtract	Less forgiving than (h)

**Table 3. 2-Operation Selection Systems**

Conceivably, there are situations where three- or four-operation systems are necessary, but experience seems to indicate that they are few and far between. It seems that a suitable two-operation system is both powerful and forgiving enough, and the addition of a third operation causes a loss of simplicity which overwhelms the improvement in power. Furthermore, requiring three mouse buttons would be a strain on the interface and the alternative method, using modifier keys, requires the user to associate particular keys with different alternatives. The example of Windows '95, where few users understand the 'shift' and 'control' selection operations, as compared to the Mac OS (Apple Computer 1986), where there is only one modifier key for the single alternative, shows the difficulty with this approach. Beyond two-operation systems, the loss in simplicity suggests the next step should

be the complete system – all five operations. Here we have surrendered any hope of the user remembering key combinations or mouse chords, and a visual reminder of the chosen operation is needed.

From 65,536 possible systems, we have therefore narrowed the field to the following five recommendations for generic visualization systems:

- Toggle only
- Replace/Toggle
- Add/Subtract
- Add/Intersect
- Replace/Toggle/Add/Subtract/Intersect

#### **4. Discussion**

Having constructed the taxonomy of selection systems, based on memory, area differentiation, data dependency and operation system, some discussion of how these features interact is appropriate.

The data dependency axis is orthogonal to the other choices, as it modifies the tool rather than dictates a choice of tool. A data dependent brush appears to be more useful than a data dependent lasso, however. The choice of a memory-based system makes a considerable difference on the choice of lasso- or brush-based tool. Multiple brushes maintained in different data views would be hard to control – the brush is by its nature transitory. One possibility is to allow lasso selectors in many views, but brush only in one view. This allows the brush selection to be constrained by other selections allowing a potentially useful conditioning functionality. The choice of system depends also on the area differentiation choice. Toggle operations are problematic with brushes, as toggling whenever the brush moves makes items within the brush vibrate in an unpleasant manner. Toggling an item only when it leaves or enters the brush is a solution to this, but in general it seems the *replace* operation and the *add/subtract* system are most appropriate.

We now look at some examples of selection systems to show where they lie in the taxonomy. First, we look at brushing scatterplot matrices, for example as described in Cleveland, W.S. and McGill, R. (1984, 1985). This paradigm is memoryless and data-independent, with a brush tool. The default operation is *replace*, termed transitory mode in the literature, with an alternative mode (paint mode) being the *add/subtract* system. Labeling mode is simply an *add/subtract* system operating on a different state for the item, namely label visibility.

Revisiting our original example, the basic Windows '95™ OS system is a memoryless, data-independent, lasso based system with *replace/toggle* operations, the secondary operation accessible via the control key. It also has an idiosyncratic alternative system using the shift key, which is memory-based (remembering the previous selected item), data-dependent (it performs differently if there is an icon at the initial mouse location) and uses a lasso. The operation used here is the *replace* operation (in the case where an icon was clicked on) or the *add* operation (if an icon was not clicked on).

The FilmFinder dynamic queries interface is memory-based (for each variable in the database there is an associated selection criterion), and uses lassos (the user typically selects a variable range by modifying endpoints on a slider, equivalent to dragging over a range in a linear view). The operation system is either *replace* (for ordinal variables) or *toggle* (for categorical variables). Although the sliders and check-boxes in FilmFinder do not constitute what most people think of as a visualization system, we can classify them as selection tools nonetheless.

As a final example, we will look at a system the author has developed for visualizing generic data using a linked views paradigm, the Exploratory Data Visualization system, EDV. The linked views paradigm is outlined in Eick and Wills (1995).

## 5. Implementation example

The purpose of EDV is to provide an environment where new data views can easily be created and added to the system so that they can be evaluated in a realistic and powerful environment. This solves one serious problem with prototyping visualization tools; it can be very hard to assess their impact if they are not used in conjunction with other tools. EDV also facilitates comparison of different approaches in order to see where the strengths and weaknesses of each lie. EDV incorporates eight different views, with more under development. These include a modified boxplot/parallel-coordinates view, a spreadsheet view, a graph view, a table view and a rose diagram, as well as standard plots like histograms, bar charts and scatterplots. These are linked together so that selections in one view can be seen and modified in another. In this system it is practical to evaluate which views are useful for which operations. EDV has been used on data sets with over 50,000 cases and sets with over 300 variables, showing it can be used for analysis of realistic, important data sets.

EDV is a memoryless system, with selection tools (at present) all being data-independent. It provides a choice of tools from a menu (

Figure 4) and allows the user to choose whatever 2-operation system they prefer using the window of figure 5.

The default choices are a rectangular lasso and the *replace/toggle* system.



**Figure 4. Tools**



**Figure 5. System Choices**

Figures 6-8 show a variety of selection operations in operation. The data being analyzed is a set of baseball statistics for hitters in 1987. This is part of a data set supplied for a competition organized by the ASA. In figure 6, the analyst is examining the relationships between three variables, *salary* (the object of the study), *years* (number of years in the league) and *hits* (a measure of ability). A scatterplot of *salary* against *hits* shows two groups with different salaries for their performances. The players in the lower group, who are paid poorly for their performances, have been selected and the linked histogram shows that most of them have been in the league for only a few years. This selection was performed with a free-form lasso carefully drawn around the set using *add/subtract* operation. A brush with the same system of operations could have been used to similar effect. Note that there is one player who has been 10 years in the league and is still paid below what his performance indicates. We could either use the *subtract* operation and remove the newer players or the *intersect* operation and select that player directly from the histogram to identify him.

In figure 7 a wide brush was dragged (using the *replace* operation) over a smoothed salary histogram and the resulting pattern observed in a table of years versus position. Older left field (LF) players get paid more than older catchers (C), and utility fielders (UT) just don't get the big bucks.

In figure 8 we show side by side boxplots, for continuous variables, and bubble-plots, for categorical variables. Making a selection in any linked view shows the selection by superimposing parallel coordinates over the view. In figure 8 we have dragged a small rectangular brush over position and are currently selecting third base players.

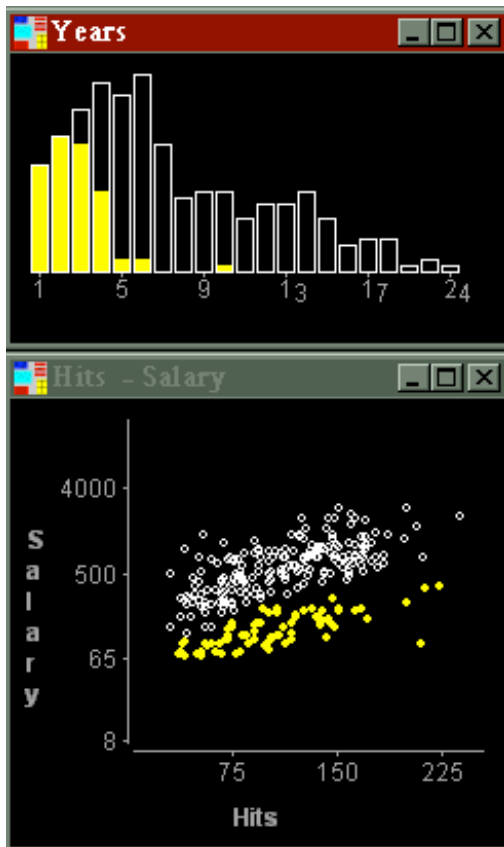


Figure 6. Selection from Scatterplot

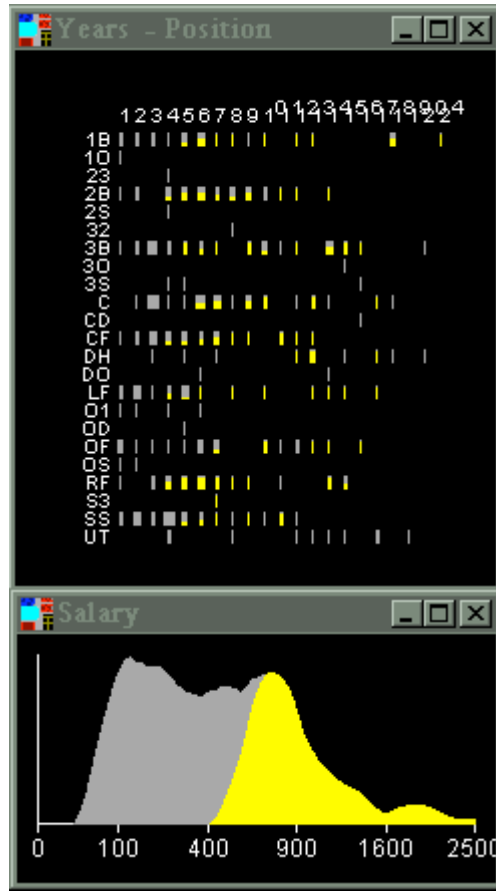


Figure 7. Selection from Histogram

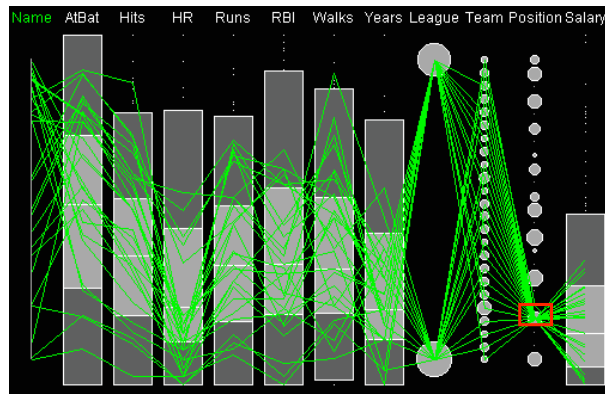


Figure 8. Brushing over third base players

## 6. Conclusions

A good visualization system has an appropriate, powerful, intuitive and forgiving selection system. It is easy to

look at such a system and be unaware of the decisions that were made to make it so. In this paper we have presented a taxonomy that allows system builders to decide on the features that they require and gives a framework for building such a system. Our suggestion of 524,288 possible systems<sup>3</sup> was more in fun than serious, as within the taxonomy there are many different choices that can be made – we did not discuss brush shape, lasso type or methods for combining memory-based selection for example. This framework is the result of considering both the current state of the art and historical antecedents and in the future we look forward to new inventions that will require us to expand it.

## 7. References

- [1] Ahlberg, C. and Shneiderman, B. (1994) *Visual Information Seeking: Tight Coupling Of Dynamic Query Filters With Starfield Displays*. Proceedings Chi '94
- [2] Apple Computer (1986) *Human Interface Guidelines: The Apple Desktop Interface* Addison-Wesley, Reading, Massachusetts
- [3] Cleveland, W.S. and McGill, R. (1984) *The Many Faces Of A Scatterplot* Journal of the American Statistical Association, 79 pp. 807-822
- [4] Cleveland, W.S. and McGill, R., eds. (1988) *Dynamic Graphics For Statistics* Wadsworth & Brooks, California
- [5] Eick, S. G. and Wills, G.J. (1995) *High Interaction Graphics* European Journal Of Operations Research #81 (1995) pp. 445-459
- [6] Fotheringham, A.S. (1992) *Exploratory Spatial Data Analysis And GIS* Environment And Planning A 1992; pp. 1675-1677
- [7] Haslett, J., Wills, G. and Unwin, A. (1990) *Spider - An Interactive Statistical Tool for the Analysis of Spatial Data* Int. J. Geographical Information Systems 4 No. 3, pp. 285-296
- [8] McDonald J. A., Stuetzle W. and Buja A. (1990) *Painting Multiple Views Of Complex Objects* ECOOP/OOPSLA '90 Proceedings
- [9] Microsoft (1995) *Microsoft Windows Online Help* Windows '95 OS
- [10] Muller, Jean-Claude (1993) *Latest Developments In GIS/LIS* Int. J. Geographical Information Systems 1993, Vol.7 #4; pp. 293-303
- [11] Swayne, D.F., Cook, D. and Buja, A. (1991) *XGobi: Interactive Graphics In The X Window System With A Link To S.* American Statistical Association 1991 Proceedings of the Section on Statistical Graphics. ASA, VA

---

<sup>3</sup> 2<sup>16</sup> operation systems x 2 (memory/memoryless) x 2 (data dependent/independent) x 2 (brush/lasso)

- [12] Velleman, P.F. (1988) *The Datadesk Handbook* Odesta Corporation
- [13] Hoaglin, D. C. and Velleman, P.F. (1995) *A Critical Look At Some Analyses Of Major League Baseball Salaries*  
American Statistician 1995, Vol.49 #3, Pp. 277-285
- [14] Tierney, L. (1990) *Lisp-Stat: An Object-Oriented Environment For Statistical Computing And Dynamic Graphics*  
Wiley, New York, NY