Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J. van der Vegt

Editing objects of a hierarchical structured drawing

Computer Science/Department of Interactive Systems Report CS-R9074 December

Editing Objects of a Hierarchical Structured Drawing

Jantien van der Vegt

Centre for Mathematics and Computer Science
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
e-mail: jantien@cwi.nl

ABSTRACT

Creating and changing a drawing with one of the current interactive drawing programs, e.g. MacDraw (Apple, 1984), Illustrator (Adobe, 1987), Freehand (Aldus, 1987) or Gargoyle (Pier, Bier and Stone, 1988), is not easy when the number of objects becomes large. From psychological theories of visual perception it is known that humans reduce a large number of objects by perceiving groups of objects as single objects (Koffka, 1935; Miller, 1956), which implies that the mental representation of a drawing has a *hierarchical structure* (Buffart, 1986; Palmer, 1977). Although the current drawing programs do allow the user to create a drawing with a hierarchical structure, only the objects at the highest level of the hierarchy can be selected and edited. This paper presents a different approach which allows the user to select and edit objects at each level of the hierarchy. The main concept, a *current object*, allows the user to travel through the hierarchical structure of the drawing in order to select objects at each level of the hierarchy. This approach not only enables the users to use the hierarchical structure that corresponds to their mental representation, but also eliminates some inconsistencies in the current drawing programs. A drawing program has been made in which the proposed way of *structure editing* has been realized. The first impressions of this implementation will be discussed.

1987 CR Categories: I.3.4. [Computer Graphics] Graphics Utilities
I.3.6. [Computer Graphics] Methodology and Techniques

Key Words and Phrases: Drawing Programs, Structure Editing

1. Introduction

Creating and changing simple drawings on a computer screen with one of the current interactive drawing programs, like MacDraw (Apple, 1984), Illustrator (Adobe, 1987), Freehand (Aldus, 1987) or Gargoyle (Pier, Bier and Stone, 1988), has become very easy, due to the well designed user interface and interaction style of these programs. Unfortunately, a good user interface or interaction style does not guarantee that a program is easy to use. Indeed, it is first of all the concepts employed in a program which should match the needs of its users. If these concepts are not appropriate for the tasks to be performed, the program won't be easy to use at all. In this paper it will be shown why the concepts of the current drawing programs are not sufficient for creating and changing more *complex* drawings.

The complexity of a drawing strongly depends on the number of objects. Due to the limited capacity of their short-term memory (Miller, 1956), humans are not able to mentally represent a large number of objects. In accordance with the so-called "Gestalt" theory (Koffka, 1935), humans will tend to organize (or cluster) groups of objects into single objects in order to reduce the complexity. For example, in Figure 1, a group of two triangles will be perceived as a single object, in order to organize the whole drawing into the manageable size of four objects.

Report CS-R9074
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

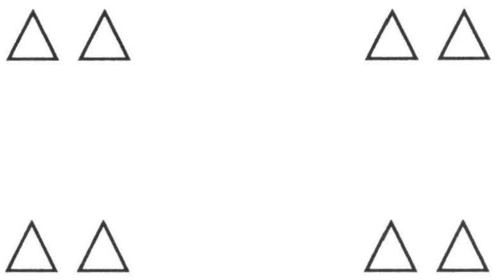
Figure 1.

Although the drawing in Figure 1 is globally perceived as four objects, humans will also perceive each of these objects as two triangles, and in turn each triangle will be perceived as a group of three line segments. This means that this drawing will be mentally represented by a tree. Figure 2 shows this tree for one of the four objects in the drawing. Nodes and leaves correspond to objects humans are able to perceive. This example illustrates that the mental representation of a drawing usually has a *hierarchical structure*, as is often stated in a number of theories of visual perception (Buffart, 1986; Palmer, 1977; Winston, 1975). Each level of the hierarchy will usually consists of only a small number of objects.

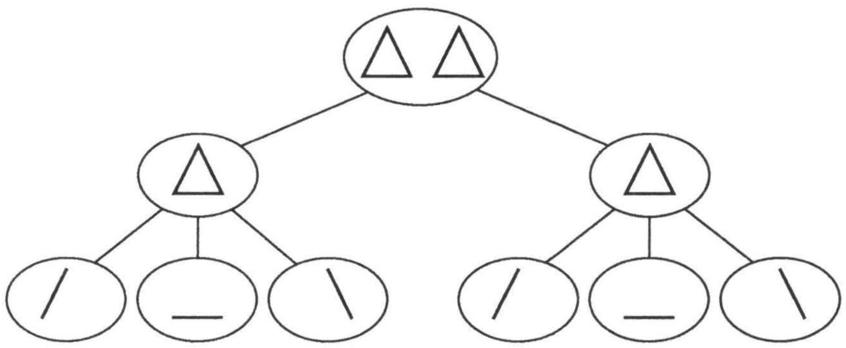
Figure 2.

It might therefore be assumed that editing a complex drawing will be more convenient for humans if they can use their mental representation of this drawing in a drawing program. Although the current interactive drawing programs do allow the user to create a drawing with a hierarchical structure, only the objects at the highest level of the hierarchy can be selected and edited. Due to this limitation the user will only group a number of objects *locally*, e.g. if objects are to be changed by the same operation. This means that the hierarchical structure used in practice does not correspond to the mental representation the user has of the drawing.

This paper will introduce a different approach to editing objects in a hierarchical structured drawing, which is called *structure editing*. The main concept, a *current object*, allows the user to select and edit objects at each level of the hierarchy. This approach not only enables users to exploit a hierarchical structure that corresponds to their mental representation, but also eliminates some inconsistencies found in the current drawing programs.

It is important to note that structure editing is not the same as syntax-directed editing. A syntax-directed graphical editor constrains a user to create and change drawings that can be described by a given syntax. As an example, the syntax-directed graphical editor of Inman (1987) assists the user in editing drawings of graphs or flow-charts with a lay-out which can be described by certain rules. Since there are no restrictions on drawings in this paper, the hierarchical structure of a drawing does not have to satisfy any syntax at all.

The structure of this paper is as follows. Section 2 focuses on the current interactive drawing programs. An overview of the concepts employed in these programs is presented together with an example to illustrate how these programs deal with the hierarchical structure of a drawing. Finally, some inconsistencies common in these programs are described. In Section 3 a different approach to editing objects of a hierarchical structured drawing is introduced. That section presents the concept of a current object, which allows the user to select and edit objects at each level of the hierarchy. It further describes how the inconsistencies mentioned earlier can be eliminated. In Section 4 the current implementation of the structure editor is described , by presenting the objects and operations of the program and by focusing on some user interface aspects. Finally, the implications of structure editing are discussed in Section 5.

2. Drawing programs

Although the physical communication between a human and a computer is not unimportant (Baecker, 1987), the most fundamental design choices for an interactive program are to be made at the *conceptual level* (Foley and Van Dam, 1982; Nielsen, 1986). At this level a computer program is defined by its *objects* and *operations* and what exactly each operation does to each object. How objects and operations are usually defined in current interactive drawing programs like MacDraw (Apple, 1984), Illustrator (Adobe, 1987), Freehand (Aldus, 1987) or Gargoyle (Pier, Bier and Stone, 1988) is described in Section 2.1 (see also: Baudelaire and Gangnet, 1989). Section 2.2 focuses in more detail on how the user has to edit objects of a drawing with a hierarchical structure in these programs. Section 2.3 describes some inconsistencies in these drawing programs which can be eliminated through structure editing.

2.1. Overview

The *basic object* in a drawing program is the smallest object in a drawing that can be created and edited by the user. This basic object is usually a trajectory or *path*, a series of contiguous line elements, in which each *line element* is specified by a *begin point*, an *end point* and some optional *control points*. Although paths in the PostScript language (Adobe, 1985) may consist of distinct parts by using the "moveto" command, drawing programs usually assume that all line elements in a path should be connected. Each path has its own set of *attributes*, which control rendering aspects of the object, like *line width*, *stroke color* or *stroke pattern*, etc. Paths can be *open* or *closed*, and for a closed path the enclosed area can be filled with a color or pattern. A closed path has therefore some additional attributes, like *fill color* or *fill pattern*. A path can be created with *templates*; the most common templates are: *line segment*, *polyline*, *rectangle* or *oval*.

Objects created by the user can be *selected* in order to change them with one of the *operations*. If more than one object is selected, then this operation will be applied to all selected objects. Examples of operations are *copy*, *cut*, *rotate*, *scale*, *set-attribute*, etc. Selected objects can also be changed without explicitly using an operation, this will be called *shape editing*. Editing the shape of a template will depend on the type of template. The shape of a line segment or polyline can be edited by moving or *dragging* one of the end points or control points of the line elements, while a rectangle or oval can usually be edited by moving one of the corner points of its *bounding box* (the rectangle that encloses the object). By dragging a selected object it is possible to *move* the whole object.

Drawing programs allow the user to create and change drawings with partially hidden objects by rendering each object in a separate *layer*. This implies that objects are *ordered* by their layers. Users do not have to remember how their objects are ordered, because this is always visible when the overlap between objects matters. The order of the layers can be changed by the user with the operations *to-front* or *to-back*, to send a selected object to the front or back layer respectively. Each new object will always be drawn on top of all other objects.

By applying the *group* operation to a number of selected objects it is possible to create a *composite object*. A composite object is not only a group of basic objects, but it can also be a group of composite objects or even a mixed group of basic and composite objects. This composite object or group can be selected as a single object and an operation on this object will be applied to all elements of the object. The shape of a composite object can usually be edited by dragging one of the four corners of the bounding box of the object. If the *ungroup* operation is applied to a selected composite object, then this composite object will be replaced by its elements. With these group and ungroup operations it is possible to create and change a drawing with a *hierarchical structure*.

2.2. Hierarchical structure

The group operation of the current interactive programs allow the user to *create* a drawing with a hierarchical structure. The following example will illustrate how a user has to *edit* objects of such a hierarchical structured drawing in these programs. Suppose someone has drawn a house by creating some basic objects and these objects have been grouped into the hierarchical structure shown by the tree in Figure 3. The leaves in this tree represent the basic objects, while the nodes represent the composite objects of the drawing. If, for some reason, this person wants to delete the roof window, then it appears to be impossible to select this object. The consequence of grouping a number of objects is namely that they are only selectable as one object, so the *selectable* objects of a drawing are always the objects at the highest level of the hierarchy. This means that, in this example, the only selectable object is the whole house. This object has to be ungrouped, then the roof object has to be ungrouped, before finally the roof window can be selected and deleted. However, ungrouping a number of objects in order to select an object also means that part of the hierarchical structure is lost. In order to keep the same hierarchical structure of the drawing, all these ungrouped objects have to be grouped again.

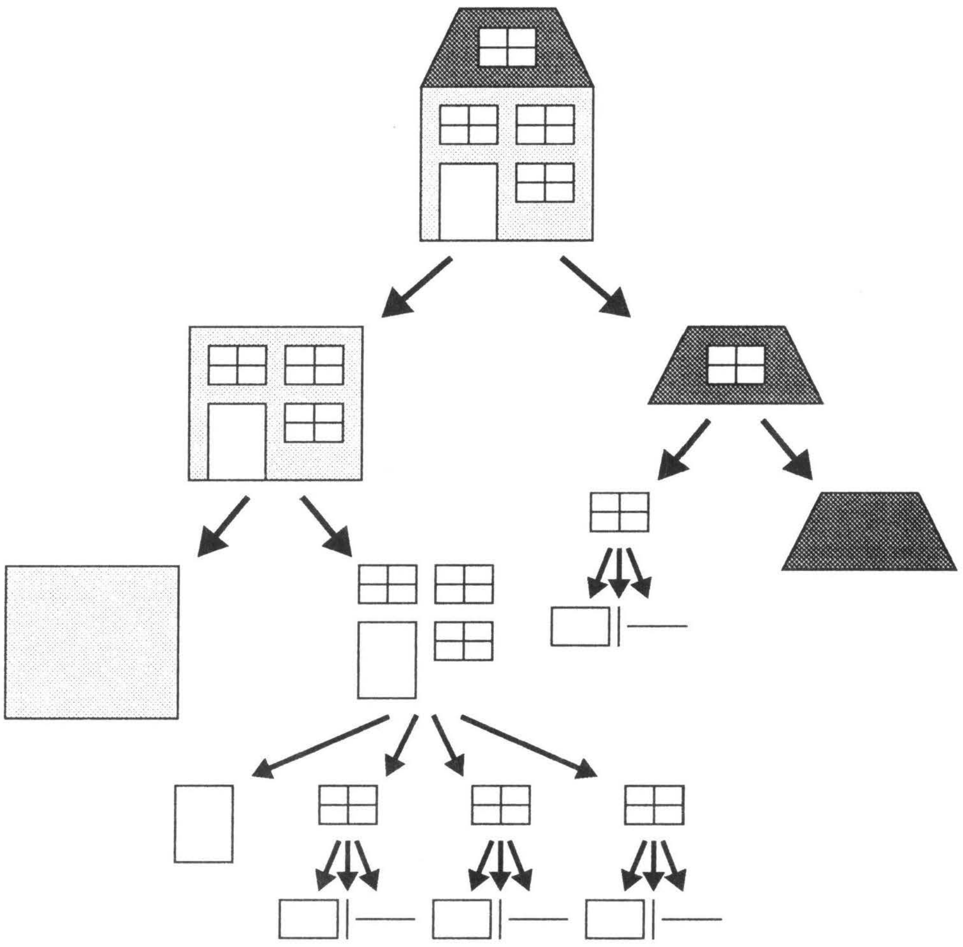

Figure 3.

This implies that the concepts of the current drawing programs do not allow the user to use the hierarchical structure they would prefer. Due to the limitation that only objects at the highest level of the hierarchy can be selected and edited, the user will only *temporarily* group a number of objects, e.g. if objects are to be changed by the same operation. Furthermore, these groups will never have a very deep nesting, because this would require too many (group and ungroup) operations. These *local* groups become rather messy for a drawing with a large number of objects, and this hierarchy usually does not correspond to the user's mental representation of the drawing.

2.3. Inconsistencies

Consistency can intuitively be described as "doing similar things in similar ways". Recently, Reisner (1990) argued that there is a missing assumption in all attempts to define consistency in a more formal way, namely: a program is only consistent if different agents (e.g. the user and the designer of the program) do agree on the classification of "similar things" and "similar ways". One might expect that all humans would normally classify objects with exactly the same visual appearance as similar objects. However, in the current drawing programs these visually similar objects will have a different response to exactly the same operation. Two types of inconsistencies can be found in the current interactive drawing programs, both are caused by the different ways that can be used to create objects with the same visual appearance.

An example of the first inconsistency is the difference between creating a rectangle with four templates of line segments or with the rectangle template. In both cases exactly the same line elements are created, but only in the second case can the rectangle be filled. Even if the four line elements that are created separately are grouped, the enclosed area can not be filled because this object was not created with a single path. This inconsistency corresponds to the dual interpretation of objects (Baudelaire and Gangnet,1989); e.g. a rectangle can be viewed as a set of four lines or as a rectangular area which can colored or filled with a pattern, as shown in Figure 4. It is hard to find this inconsistency in Illustrator (Adobe, 1987), because this program will automatically join a new line element to an existing open path if they are connected. But if a line element appears to be connected with an open path after one of these objects has been edited, then they will not be joined.

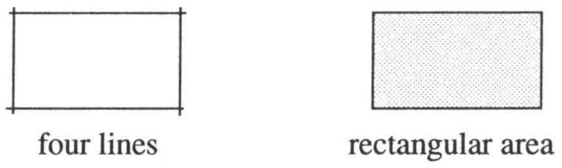

Figure 4.

The second inconsistency is caused by the different template types that can be used to create paths. It is possible to create exactly the same path with different templates. For example, a rectangular path can be created by a rectangle template or a polyline template. Although the paths are exactly the same, editing the shape of one of these paths will depend on the template type that was used to create the path. Figure 5 shows that the result of editing the shape of a rectangle template is always a rectangle again, while editing the shape of a polyline will change the rectangle into a quadrilateral.

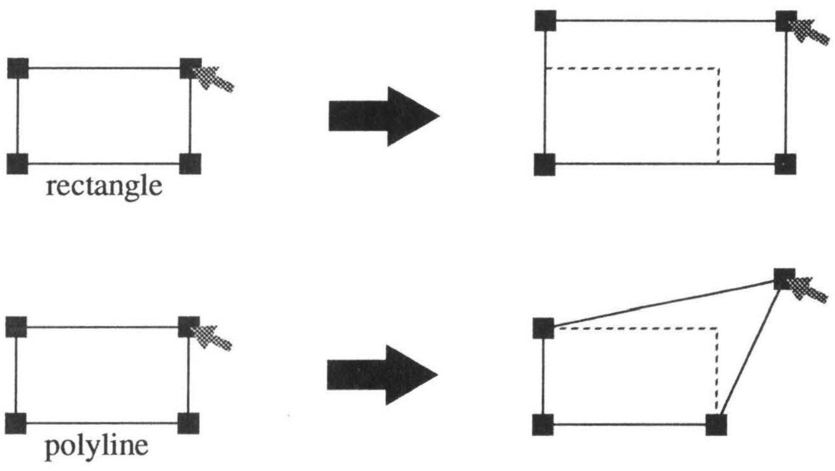

Figure 5.

The drawing programs Illustrator (Adobe, 1987) and Freehand (Aldus, 1987) avoid this second inconsistency by regarding the templates for a rectangle and oval as a grouped polyline. Editing the shape of a group is usually a scaling operation and that is exactly what should happen for these templates, as the scale operation leaves the characteristic shape of the object invariant. By ungrouping the template of a rectangle or oval the user gets the corresponding polyline, and the shape of the path can be changed in all end points and control points of this polyline. However, it seems illogical to regard exactly the same path in some cases as a basic object and in other cases as a group.

3. Structure editing

This section presents a different approach to editing objects of a hierarchical structured drawing, which is called *structure editing*. Section 3.1 shows how the user is able to select each object of the drawing, while maintaining the hierarchical structure. The main concept, a *current object*, allows the user to travel through the hierarchical structure of the drawing in order to select objects at each level of the hierarchy. Section 3.2 describes how structure editing enables the elimination of the inconsistencies in drawing programs.

3.1. Current object

As argued before, it would be much more convenient if each object of a drawing could be selected while retaining the hierarchical structure. This is why the concept of a *current object* will be introduced. By default this current object is the whole drawing, a group which contains all objects of the drawing. This group is called the *home* object and it is automatically created by the editor. The *selectable* objects are always the immediate sub-objects of the current object. The set of selectable objects will change by changing the current object. This enables the user to travel through the drawing along the hierarchical structure in order to select and edit objects.

Three new operations have to be defined to change the current object, namely *open*, *close* and *home*. The open operation replaces the current object by the currently selected object. If the selected object is a basic object, then the current object will not change since the current object should always be a composite object. If multiple objects are selected, the open operation will be applied to the last selected object, as there can only be one current object. With the close operation the current object is replaced by its parent object, unless the current object was already equal to the home object. The home operation replaces the current object by the home object. Note that the open operation has to be applied to a selected object, while the close and home operation are completely independent of the selected objects.

The following example will illustrate how multiple objects can be selected. Suppose someone wants to select all windows of the same house as in Figure 3. From the default current object, which is the whole house, it is possible to select and open the roof object in order to select the roof window. After that the home operation can be applied to return to the default current object. The roof window remains selected as the home operation does not change the objects which are selected. It is now possible to select and open the "house with no roof" object without deselecting the roof window. In the same way the object with the front parts can be selected and opened, and finally the three other windows can be selected. In other words, it is possible to select multiple objects, even if they belong to different groups at different levels of the hierarchy. There is a restriction though and it states that the selected objects should be *disjoint*, that is, they may not share elements. For instance, it is not possible to select the roof object as well as the roof window, since the roof window is an element of the roof object. If the last selected object is not disjoint from one of the selected objects, then this previously selected object will be deselected.

Users will not constantly be aware of the current hierarchical structure of their drawing. It is therefore essential that there is a good visual feedback of the current object or the currently selectable objects. This feedback should be distinguishable from the normal visual feedback of the objects which are selected. This problem and other user interface aspects will be further discussed in Section 4.2.

3.2. Paths

In this section it will be shown how structure editing enables the elimination of the inconsistencies in the drawing programs mentioned earlier. Recall that the first inconsistency was caused by the difference between a set of line elements that are created separately and a path with exactly the same line elements. The second inconsistency was caused by the different template types that can be used to create exactly the same path. Since both inconsistencies are dealing with paths, a number of changes to this concept are proposed.

Structure editing enables the user to edit objects at each level of the hierarchy. This means that the user can select and edit a whole path as well as each line element of a path separately, if a path is always regarded as a composite object of line elements instead of a basic object. As it is very time consuming for a user to create only line elements, it is still possible to create objects with templates. However, invoking a template of a rectangle, oval or polyline now adds a group of line elements to the drawing instead of adding a basic object. This group can be edited in exactly the same way as groups that are explicitly created with the group operation. Nevertheless a group of line elements that form a closed path should be a special composite object, because it can be filled with a color or pattern. In order to assign these extra path attributes, the editor should therefore recognize a group of connected line elements as a path even if all line elements were created separately. This should not be done only when new line elements have been created but also every time that line elements have been edited. The algorithm that is used consists of a simple test to check whether all end points of the elements of the group have an even degree, that is, if each end point appears exactly twice or four times, etc. If this is the case, then the line elements should be rearranged in a path-like way. The area that has to be filled will be independent of the path that is arbitrarily chosen by the editor. In this way the first inconsistency is eliminated; whenever there is a group of line elements which forms a closed path it will be recognized by the editor, even if all line segments were created separately. This does not mean that each closed path would always be filled; the user can still set the fill attribute to a transparent color or pattern.

Eliminating the first inconsistency is not the only reason for regarding a path as a composite object. The example in Figure 2 already illustrated that although humans perceive a triangle as a single object, they also perceive it as a group of three line elements. This implies that humans also regard a path as a group of line elements.

The second type of inconsistency was caused by the different template types that can be used to create exactly the same path. Editing the shape of a path therefore depends on the template type that created the path. In order to eliminate this inconsistency the following changes for shape editing are proposed. First, the bounding box of an object should no longer be used to edit the shape of an object, because these changes can also be done with the scale operation. (However, one could imagine that a bounding box is used to perform the scale operation, but that is something different.) Furthermore, structure editing enables the user to differentiate between editing the shape of a basic object or the shape of a composite object. Dragging an end point of a group is now defined as editing the shape of the elements of the group that share this end point. So if the user drags an end point of a path then only the line elements of this path that share this end point will change accordingly. Figure 5 shows the results of moving an end point of a path or of a line element. If the user moves an end point of a line element of a path, it might happen that this path is no longer closed. If this is the case the path can not be filled.

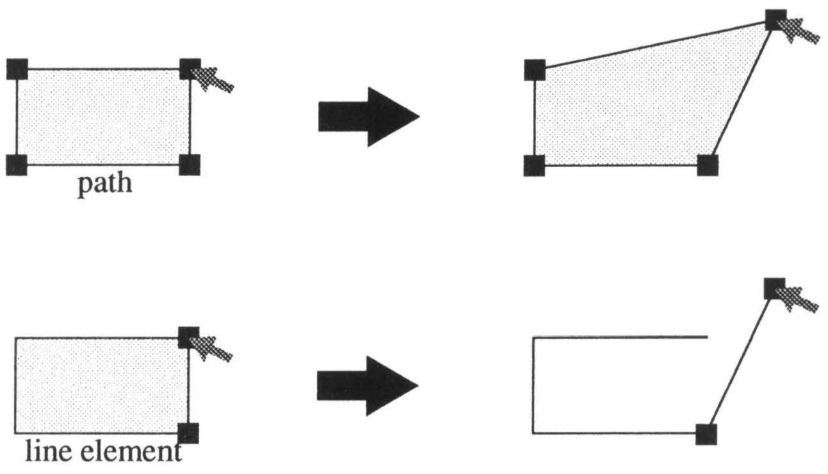

Figure 6.

4. Implementation

The proposed way of structure editing has been implemented in a drawing program. It runs on a Sun SPARC station on top of HyperNeWS, a interactive user interface development system developed by the Turing Institute in Glasgow. Section 4.1 presents some of the consequences for the objects and operations in this program as a result of introducing the concept of a current object, while Section 4.2 focuses on some user interface aspects.

4.1. Objects and operations

In order to define only one type of basic object, it is assumed that each line element is a *Bézier* curve, that is, a line element with a begin point, an end point and exactly two control points. Some examples of Bézier curves are shown in Figure 7. Black squares in this figure represent the end points of a line element, while the control points are represented by open circles. The first line element in Figure 7 is an example of a straight line segment, a Bézier curve where the control points are exactly in the same position as the end points. The user can change a straight line segment into a curve and vice versa simply by dragging a control point of the line element. The drawing programs Illustrator (Adobe, 1987) and Freehand (Aldus, 1987) also use Bézier curves as line elements in their paths.

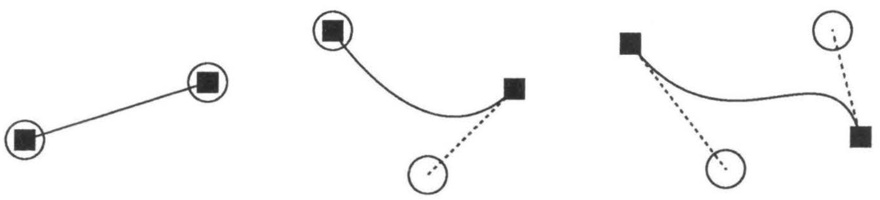

Figure 7.

Although the current object is mainly used for object selection, it also has a role when new objects are added to the drawing. As with the current drawing programs, there is a drawing mode for each of the template types which allows the user to create new objects. Currently, there are drawing modes for line segments, rectangles, ovals and polylines. In the drawing mode for line segments the user adds a single line element to the drawing, while in the other drawing modes a group of line elements will be added to the drawing. However, the new object will now be added to the current object, that is, it becomes a new member of this group.

The current object also has some consequences for the operations in the structure editor. First of all, it requires the new operations *open*, *close* and *home*. All other operations will be similar to the operations in the current drawing programs, though some of them have to be defined differently. For example, if objects are deleted, then the current object might become empty. If this is the case, the current object is replaced by its parent object and the empty group is deleted automatically from the drawing. Furthermore, the group operation now adds a new object to the current object. The members of this new group are deleted from the composite objects to which they belong. If, through this operation, one of these composite objects becomes empty, it is also deleted automatically from the drawing.

Editing the shape of an object will depend on the selected object. As mentioned before, structure editing enables the user to differentiate between editing the shape of a composite or of a basic object. Figure 6 has already shown what happens if the user drags an end point of a path and of a line element of a path. What happens in the same situations when the user moves a control point or the whole object is shown in Figure 8 and Figure 9 respectively. It might happen that a path is split into two parts.

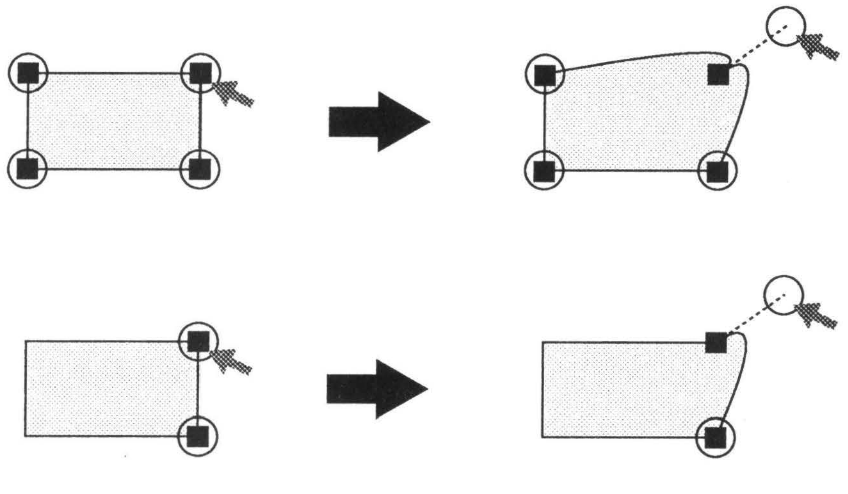
Figure 8.

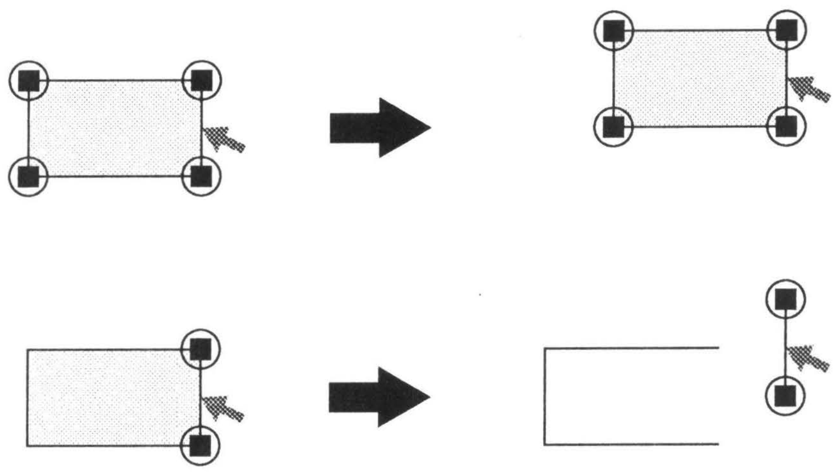

Figure 9.

4.2. User interface

The user is either in one of the drawing modes or in the selection mode. Depending on the mode, the user can select or draw objects with the left or *select* button on the mouse. With the middle or *edit* button on the mouse the user can always edit the shape of a selected object, independent of the current mode. The right or *menu* button on the mouse gives the user a pop-up menu, from which it is possible to select the operations. For each menu option a corresponding key-board accelerator is available.

Beside the feedback of the currently selected objects, there should be a good feedback of the current object or the currently selectable objects. As the current object is mainly used for object selection, a feedback of the currently selectable objects seems more appropriate than a feedback of the current object. I have been experimenting with a number of different feedback techniques of these objects. For example, one could temporarily hide the objects that are currently not selectable, but the missing context will hinder the user in the editing process. An alternative is to let the selectable objects blink or highlight, but the problem here is that this feedback is often confused with the feedback of the currently selected objects. Finally, it has been decided that the objects that are temporarily not selectable get a gray color, in analogy with menu options that are temporarily not selectable. With this feedback the context is still available and the selectable objects are not confused with the selected objects, as their appearance remains the same. Nevertheless the selectable objects will be more or less highlighted since all other objects are gray. This feedback will only give problems if the user makes a drawing with exclusively gray objects.

In using this structure editor I discovered the need for accelerators or short-cuts of the operations that change the current object. The following extensions have therefore been implemented. Clicking with the select button of the mouse on a object that is already selected, will directly open this object. Since the first click will select an object, a double click (but without time limits) will directly open an object, in analogy with the Macintosh style. Clicking on a gray object will directly select this object at the level where this object and the current object have a common parent. This common parent will then be the new current object. Clicking in an empty area will directly close the current object. The home object can thus be reached by multiple clicks in an empty area, depending on the level of the current object.

5. Discussion

Unfortunately, the structure editor has not been used by real users yet, so I can only describe some of my own experiences with this program. First of all, I found it very frustrating to use a traditional drawing program again after working for a while with the structure editor. The drawings in this paper were all created with the drawing tool of FrameMaker, a tool that is comparable with the drawing programs described in Section 2.1. I frequently found myself clicking at composite objects in order to open them, before I realized that I first have to ungroup the object. Second, I very much liked the flexibility with paths, e.g. that each line element in a path can have different attributes, as one can edit them separately. Furthermore, one never has to remember how a path was created. A rectangle created with four templates of line segments or created with a rectangle or polyline template will always have the same behaviour. A disadvantage of the structure editor is that one has to realize what the current object is when new objects are added to the drawing. As the current object is mainly used for selecting objects, I sometimes forgot that a newly created object will be added as a member of the current object. A possible solution for this problem (which would mean a slight change on what has been said before) is to add new objects by default at the home object, and only by request as a member of the current object.

This paper assumed that editing a complex drawing will be easier if people can use their mental representation of the drawing as the hierarchical structure in a drawing program. Although there are drawings in which humans could perceive more than one hierarchical structure, a number of theories (Buffart and Leeuwenberg, 1983; Hatfield and Epstein, 1985; Koffka, 1935) state that humans will always have a preference for a specific structure. It is therefore not very likely that a user has more than one mental representation for the same drawing, although this representation can be different for different people. Structure editing enables users to edit drawings easily with the hierarchical structure they prefer to use. Whether the mental representation will indeed be used as the hierarchical structure of the drawing, still depends on the user. Users are not forced to use a hierarchy, they are still able to edit a drawing with a *'flat'* structure, i.e. a drawing with only a number of basic objects.

How the hierarchy will be used in practice is still an open question as there are no experiences of real users yet, but some possible fields of further research and development are already visible. One of these is as follows. In the current drawing programs users seem to prefer to group a number of objects that are similar in some respect, because they often have to be changed in the same way. In the structure editor I can also imagine that users prefer to group similar objects if they have to be changed more than once. For example, selecting all windows of the house in Figure 3 takes quite a lot of time as they belong to different groups at different levels of the hierarchy. Consequently, users would have to switch several times between different hierarchies. Furthermore, these hierarchies have nothing to do with the user's mental representation. Similar objects do not have to be grouped in order to select them quickly, if these objects can be selected with a *search* operation. Graphical search and replace (Kurlander and Bier, 1988) enables the user to select and change similar objects with only one operation. This technique allows the user to specify in a search window the properties (shape, color, etc.) of the objects to be searched for. The matched objects will then be replaced by the properties specified in the replace window. One plan for the future is to extend the structure editor with this powerful technique, in order to eliminate the need to group similar objects. This prevents the user from switching several times among different hierarchies and enables them to constantly use their mental representation as the main hierarchical structure of the drawing.

The plans for the future include to further test and evaluate this structure editor. Although the first impressions of this implementation seems promising, it is necessary to get more practical experiences of real users of this program.

6. References

Adobe (1985). *PostScript Language Reference Manual.* Adobe System Inc., Palo Alto.

Adobe (1987). *Adobe Illustrator User's Manual.* Adobe System Inc., Palo Alto.

Aldus (1987). *Freehand User's Manual.* Aldus Corp., Seattle.

Apple (1984). *MacDraw Manual.* Apple Computer Inc., Cupertino.

Baecker, R. (1987). Towards an Effective Characterization of Graphical Interaction. In R.M. Baecker and W.A.S. Buxton (Eds.), *Readings in Human-Computer Interaction: a Multidisciplinary Approach.* California: Morgan Kaufmann Publishers, Inc.

Baudelaire, P., and Gangnet, M. (1989). Planar Maps: an Interaction Paradigm for Graphic Design. In K. Bice and C. Lewis (Eds.), *CHI'89 Conference Proceedings.* Addison-Wesley Publishing Company, Inc.

Buffart, H. (1986). Gestalt Qualities, Memory Structure and Minimum Principles. In F. Klix and H. Hagendorf (Eds.), *Human Memory and Cognitive Capabilities.* Amsterdam, NL: North-Holland.

Buffart, H, and Leeuwenberg, E. (1983). Structural Information Theory. In H.-G. Geissler, H.F.J.M. Buffart, E.L.J. Leeuwenberg and V. Sarris (Eds.), *Modern Issues in Perception.* Amsterdam, NL: North-Holland.

Foley, J.D., and Van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics.* Addison-Wesley Publishing Company, Inc.

Hatfield, G.C., and Epstein, W. (1985). The status of the minimum principle in the theoretical analysis of visual perception. *Psychological Bulletin,* 97, 155-186.

Inman, E. (1987). A Syntax-Directed Graphics Editor. In H.-J. Bullinger and B. Shackel (Eds.), *Human-Computer Interaction – INTERACT'87.* Elsevier Science Publishers B.V., North-Holland.

Kurlander, D., and Bier, E.A. (1988). Graphical Search and Replace. *Computer Graphics, 22,* 113-120.

Miller, G.A. (1956). The magical number seven, plus or minus two. *Psychological Review,* 63, 81-97.

Nielsen, J. (1986). A Virtual Protocol Model for Computer-Human Interaction. *Man-Machine Studies,* 24, 301-312.

Palmer, S.E. (1977). Hierarchical Structure in Perceptual Representation. *Cognitive Psychology,* 9, 441-474.

Pier, K., Bier, E., and Stone, M. (1988). An Introduction to Gargoyle: an Interactive Illustration Tool. In J.C. van Vliet (Ed.), *Proceedings of the International Conference on Electronic Publishing. Document Manipulation and Typography (EP88),* Cambridge University Press.

Reisner, P. (1990). What is Inconsistency? In D. Diaper et al. (Eds.), *Human-Computer Interaction – INTERACT'90.* Elsevier Science Publishers B.V., North-Holland.

Winston, P.H. (1975). Learning Structural Descriptions from Examples. In P.H. Winston (Ed.), *The Psychology of Computer Vision,* New York: McGraw-Hill.