LOCATING, REPLACING AND DELETING PATTERNS IN GRAPHICS EDITING OF LINE DRAWINGS

Daniel Thalmann, Louis-Philippe Demers
Départment d'informatique et
recherche opérationnelle
Université de Montréal
Montréal, Canada

Nadia Magnenat-Thalmann
Ecole des Hautes Etudes
Commerciales
Montréal, Canada

ABSTRACT

The operations of locating, deleting or replacing patterns in a given line drawing are very important in graphics editing. However, they are generally not yet implemented in general-purpose graphics editors. We have designed algorithms to implement these operations in an efficient way and we have introduced them into the GRAFEDIT general-purpose graphics editor. This paper presents different aspects of the three operations, and discusses the algorithms and data structures chosen.

key-words graphics editing, pattern recognition

RESUME

Les opérations de localisation, suppression et substitution de formes dans un dessin donné sont très importantes dans l'édition graphique. Cependant, elles ne sont généralement pas implantées dans les éditeurs graphiques d'intérêt général. Nous avons développé des algorithmes pour implanter ces opérations de manière performante et nous les avons introduites dans notre éditeur graphique GRAFEDIT. Cet article présente différents aspects des trois opérations et explique les algorithmes et les structures de données choisies.

mots-clé édition graphique, reconnaissance de formes

1. INTRODUCTION

Almost since the beginning of computer science, text editors [1,2] have been available and they are a standard utility of any operating system from home computers to large computer systems. There are different kinds of text editors, but they all include four fundamental operations: INSERT a string of characters, DELETE a string of characters, LOCATE a string of characters and REPLACE a string of characters by another string.

There has also been rapid development in computer graphics during the last decade. Interactive graphics systems are running in numerous organizations and CAD has become an important area of computer science. A specific kind of interactive graphics system is now popular: the graphics editor [3]. Its role is similar to the role of a text editor, except that drawings are manipulated instead of texts. The user of a graphics editor creates drawings and then manipulates them by rotations, translations and other operations. Drawings can be stored and retrieved exactly in the same way as texts.

However, there is a major difference between text editors and graphics editors: the four fundamental operations of the text editors (INSERT, DELETE, LOCATE and REPLACE) have no equivalent in graphics editors. Of course, these operations can be applied to pictures and are important for pattern recognition and image processing [4,5,6,7], but they are not currently available in graphics editors, except for INSERT which acts to "put a new figure in the drawing".

We have developed a set of procedures to perform the three other operations and we have introduced interactive commands corresponding to these operations into the GRAFEDIT [3] general-purpose interactive graphics editor that we have developed.

2. THE NEW INTERACTIVE COMMANDS

GRAFEDIT is a command-directed graphical editor that can create, modify, transform, copy, store and retrieve almost any kind of two-dimensional diagram on a CRT terminal. The user selects commands from a set of thirty-five different instructions, and types them on the keyboard of a graphics CRT. The system is interactive: prompts, responses, error messages and various kinds of visual and auditory feedbacks assist the user during the editing process. A GRAFEDIT session is a continuous, simple, and constructive dialogue between the user and the system. At the beginning of a session, GRAFEDIT divides the screen in three areas as shown in Fig. 1: the diagram area, the area for the user-definable menu of diagrams and the alphanumerical command/message area.

Five new commands have been introduced in GRAFEDIT to perform the new operations:

1) PREPROCESS [ADD]

As the operations must be applied to drawings created by an existing graphics editor, the graphical objects to which the operations apply must be prepared. For example, suppose we want to locate a graphical object in a given drawing. The object of the search is called a pattern and the given line drawing is the basic figure (for the other operations, the terminology is exactly the same). The basic figure and the pattern must be prepared before any operations. PREPROCESS prepares the figure that the user identifies as the basic figure for pattern handling. If the parameter ADD is present the figure is added to the current basic figure.

2) LOCATE [SIZE]

This command searches for the pattern indicated by the user in the basic figure. When this is found, the rectangle that surrounds it is displayed in dashed lines. If the parameter SIZE is present, the pattern must have the specified size. Each time this command is typed a new occurrence is sought.

3) DELETE [LAST] [,SIZE]

This command deletes the pattern indicated by the user or the last pattern found by the LOCATE command (if LAST is used). SIZE has the same meaning as in LOCATE.

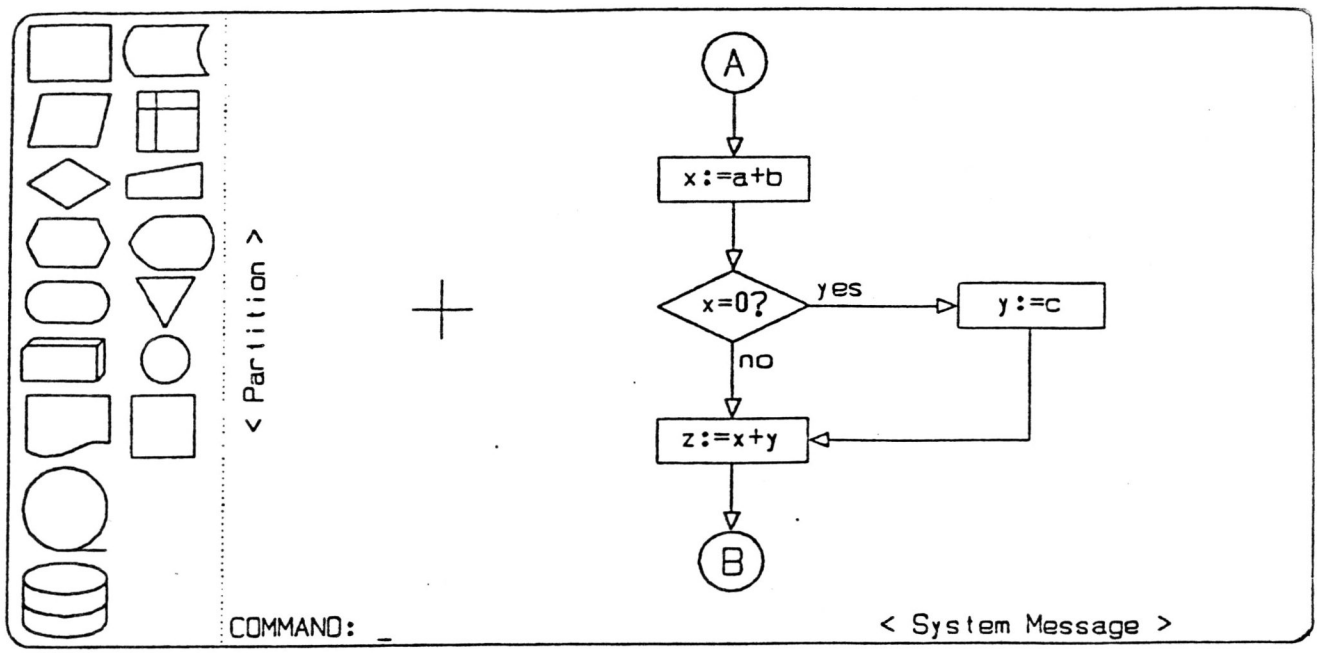

Fig. 1 The GRAFEDIT screen

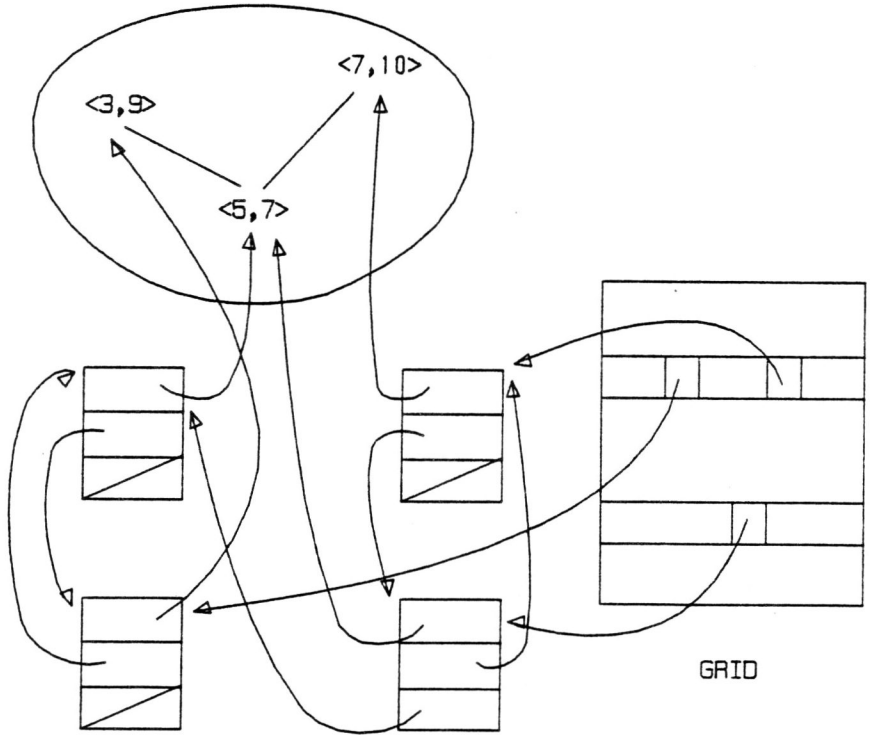

Fig. 2 Data structure (an example)

4) REPLACE [LAST] [,SIZE]

This command replaces the pattern indicated by the user or the last pattern found by the LOCATE command by a new figure indicated by the user. SIZE has the same meaning as in LOCATE.

5) ENDPATTERN

This command destroys the information concerning the basic figure and the last pattern used.

3. ALGORITHMS AND DATA STRUCTURES

The basic data structure used in the algorithms is an information on the basic figure. This grid is a two-dimensional array of pointers to blocks that contain a pointer to the corresponding point of the basic figure, a pointer to the other end of the line segment arriving at this point and a pointer to another line segment starting from this point. Fig.2 gives an example of the data structure for a basic figure of only 3 points. Data structures that contain informations about the basic figure and the pattern are listed in the PASCAL definition presented in Fig.3.

The algorithm for preprocessing of the basic figure can be summarized as follows:

1. if the basic figure is a new one or
   the extrema are modified then
   compute the extrema and the center;
   compute the scaling factor which makes the figure unitary;
   compute the translation that places the figure at the origin;
2. for each stroke of the figure do
   search for the subscripts of the grid at both ends;
   update the data structure in the grid;

The preprocessing algorithm for the pattern is as follows:

1. define a reference point for the pattern;
2. compute the extrema, the center and the distance of the center from the reference point;
3. rotate the pattern so that the first stroke is on the X-axis
4. translate the reference point to the origin
5. make the first stroke unitary using a scaling factor
6. for each stroke of the pattern do
   count it;
   describe its position relative to the reference point

Step 6 is included to provide a faster access within the grid; it avoids certain calculations during the LOCATE operation.

LOCATE operation

The algorithm for the LOCATE operation can now be described in pseudo-PASCAL.

```
repeat
  if this is the first trial for a given
     stroke in the basic figure then
     given a reference point in the basic
     figure, fetch a first stroke corresponding to this in the pattern;
     check if the other strokes also match
  else (* second trial: the pattern can be
     found for the stroke
       in two ways: PT1-PT2 or PT2-PT1 *)
     reverse the reference point;
     check if the other strokes also match
until a match occurs or the complete search
     is unsuccessful
```

Fig. 4 shows an example of patterns using both trials.

```
const
  MAXGRID=50; (*grid dimension*)
  MAXPATTERN=500;(*maximum number of points in the pattern to be found*)
type
  PTGRID=^PIXEL;
  TGRID=packed array [0..MAXGRID,0..MAXGRID] of PTGRID;
  PIXEL=packed record
                PT1:FIG;(*pointer to the basic figure*)
                BROTHER, (*pointer to the other end of the
                          line segment arriving at this point*)
                NEXT:PTGRID; (*pointer to another line segment
                              starting from this point*)
               end;
  ENVIRONMENT=record
              (*information concerning the basic figure*)
                BASE:FIG; (*basic figure*)
                TRANSX,TRANSY (*to translate the basic figure to origin*)
                SCALEBASE:REAL; (*scale factor to make the basic figure
                                  unitary*)
                STROKE:FIG; (*current stroke*)
                GRID:TGRID; (*information grid*)
                MIN,MAX,CENTERBASE:VECTOR; (*extrema and center of the
                                            figure*)
              (*information concerning the pattern*)
                PMAX,PMIN:VECTOR; (*extrema of the pattern*)
                PATTERN:FIG;
                PSCALE, (*scale factor*)
                FANGLE, (*angle*)
                DISPX,DISPY, (*pattern center*)
                DISTSTROKE:REAL; (*current stroke size*)
                NBPATTERN, (*number of vertices*)
                NBSTROKE: 0..MAXPATTERN; (*number of strokes*)
                DESCRIPTX,DESCRIPTY:array [0..MAXPATTERN] of REAL;
                              (*relative position of the points*)
                NEXTOK:packed array [0..MAXPATTERN] of BOOLEAN;
                              (*tests if the points are connected*)
                STACK:packed array [0..MAXPATTERN] of PTGRID;
                              (*set of pattern blocks in the basic figure*)
              (*search status*)
                STATE:(FIRST,SECOND); (*first or second trial*)
                PATFOUND:BOOLEAN; (*pattern found?*)
              (*description of the pattern to be found*)
                FSCALE,FANGLE (*scale and angle*)
                X,Y:REAL (*reference point*)
                REF1,REF2:VECTOR; (*current line from REF2 to REF1 or
                                    from REF1 to REF2*)
              end;
```

Fig. 3 Data structure in PASCAL

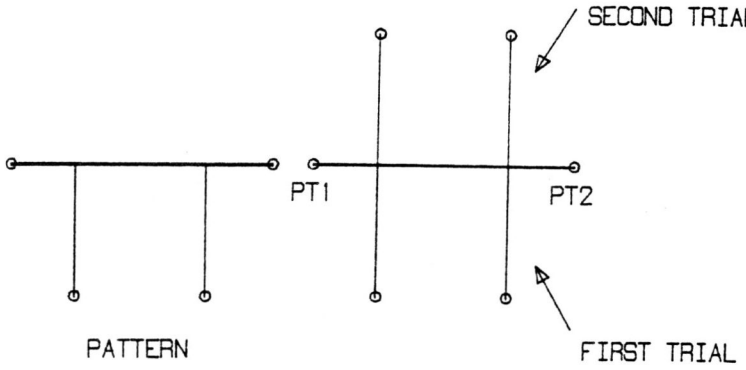

Fig. 4 Two trials for a pattern

The stage of searching for the first stroke corresponding to the first stroke of the pattern is based on simple comparisons. Checking whether other strokes match is more difficult. The pattern has to be compared point by point with the basic figure. This is performed by a loop on every stroke. Strokes are pushed onto a stack during this process.

Scaling, rotation and translation allows the transformation of two ends of a stroke so that they can be compared with the points stored in the grid. The LOCATE operation gives the following information:

- lower right and upper left corners of the rectangle that surrounds the pattern
- pointers in the basic figure
- F and a used in the REPLACE operation

DELETE operation

This operation is implemented by the following algorithm:

```
if not already located then locate
the pattern
if OK then
   for every stroke do
        make the stroke invisible if it
        must be deleted
```

To make the strokes invisible is a rather easy operation because as has already been mentioned, the strokes of the pattern where pushed onto a stack during the LOCATE operation.

REPLACE operation

The algorithm for this operation can be summarized as follows:

```
if not already located then locate
the pattern

if OK then
   compute a scale factor F;
   compute a rotation angle a;
   determine the center CT of the
   new figure NEWF;
   rotate NEWF around CT by the angle
   a;
   translate NEWF to the origin;
   scale NEWF by the factor F;
   translate NEWF to the reference
   point of the pattern to replace;
   add NEWF to the basic figure
```

4. CONCLUSION

GRAFEDIT is an interactive program, presently implemented on a D.E.C. VAX 11 and a CDC Cyber 173 with the following terminals: HP2648A, TEKTRONIX 4027 and D.E.C. GIGI. The program is written in MIRA-2D [8,9], a graphical PASCAL [10] extension based on abstract graphical data types [11].

To facilitate the introduction of the new commands, corresponding procedures have been implemented in PASCAL and the data structures are completely compatible with MIRA-2D. This also means that the procedures can be called in any MIRA-2D program. New commands have been only implemented on the D.E.C. VAX 11.

The operations of locating, deleting and replacing patterns in a given figure are very important to perform graphics editing. However they are generally not implemented in general-purpose graphics editors. We have designed algorithms to implement these operations in an efficient way and we have introduced them into the GRAFEDIT graphics editor. Because of the structured approach used to develop GRAFEDIT and the implementation language MIRA-2D, the new commands have been introduced in a natural way and the results are quite satisfactory. In the near future, we expect to augment the tolerance, although that it should cause an

important performance degradation.

ACKNOWLEDGEMENT

The authors are grateful to Ann Laporte who has revised the English text. This work was sponsored by the Natural Sciences and Engineering Council of Canada.

REFERENCES

[1] Meyrowitz, N. and van Dam, A. "Interactive Editing Systems", Document Preparation Systems, North-Holland, 1982, pp.21-132.

[2] Furuta, R.; Scofield, J. and Shaw, A. "Document Formatting Systems: Survey, Concepts and Issues", Document Preparation Systems, North-Holland, 1982, pp. 133-210.

[3] Magnenat-Thalmann, N.; Thalmann, D.; Larouche, A. and Lorrain, L. "GRAFEDIT: an Interactive General-purpose Graphics Editor", Comput. and Graphics, Pergamon Press, Vol. 6, No. 1, 1982, pp.41-46.

[4] Pavlidis, T. "A Review of Algorithms for Shape Analysis", Comp. Graphics and Image Processing, Vol. 7, 1978, pp. 243-258.

[5] Pavlidis, T. "Algorithms for Shape Analysis of Contours and Waveforms", IEEE Trans. Pattern Analysis and Machine Intelligence, 2, 1980, pp.301-312.

[6] Pavlidis, T. Algorithms for Graphics and Image Processing, Springer-Verlag, 1982.

[7] Rosenfeld, A. and Kak, A.C. Digital Picture Processing, Academic Press, 1976.

[8] Magnenat-Thalmann, N. and Thalmann, D. "A Graphical PASCAL Extension Based on Graphical Types", Software-Practice and Experience, Vol. 11, Nr. 1, 1981, pp.53-62.

[9] Magnenat-Thalmann, N. and Thalmann, D. "Some Unusual Primitives in the MIRA Graphical PASCAL Extension", Comput. and Graphics, Pergamon Press, Vol. 6, No. 3, 1982, pp. 127-139.

[10] Jensen, K. and Wirth, N. PASCAL User Manual and Report, Springer-Verlag, 1974.

[11] Thalmann, D. and Magnenat-Thalmann, N. "Design and Implementation of Abstract Graphical Data Types", Proc. COMPSAC'79, Chicago IEEE Press, 1979, pp. 519-524.