Behavior Research Methods, Instruments, & Computers
1995, 27 (1), 76–82

Program Abstracts/Algorithms

MICELAB: Spatial processing of mouse movement in Turbo Pascal

THIERRY BACCINO
Université de Nice Sophia-Antipolis, Nice, France

and

ALAN KENNEDY
University of Dundee, Dundee, Scotland

Pointing tasks with a mouse are very useful in psychological experiments concerned with a subject's ability to track a visual object or to move to a target location (e.g., to find a word in text). Such tasks are quite easily performed by subjects, and numerous variables may be obtained from the mouse movement. In this article we describe a set of Turbo Pascal 6.0 units and a program running on the IBM PC family that show how data (x,y coordinates) recorded from a mouse movement can be analyzed to give dispersion, direction, and distance of the movement.

In an earlier paper, Crosbie (1990) showed that the Microsoft mouse available with any microcomputer of the IBM family could be a sophisticated multipurpose response device in psychological experiments. Programming the mouse with the DOS Interrupt 33H, which provides several functions, appears very convenient and avoids inappropriate keyboard responses while reducing the delay until a keypress is detected by a program. However, if the mouse is used in reaction time studies, such as that in Crosbie's article, certain precautions should be taken to improve timing accuracy, such as preventing any mouse movement (Segalowitz & Graves, 1990) and correcting time registration (Beringer, 1992).

Curiously, Crosbie and others do not describe how to get a spatial description of the mouse movement whenever it is used to track a visual object or to find a word position previously stored in memory. Typically, in such an experiment the subject's task is to move the mouse cursor to a target location, given a certain starting point (e.g., pointing task), as is shown in Figure 1.

Numerous applications of a pointing task with a mouse are possible in cognitive psychology, psycholinguistics, and human factors (e.g., work on spatial memory, visual search, word identification, text reading, or information displays on screen). We describe several spatial variables that might be extracted from a pointing task, thereby increasing the set of variables that can be recorded using the mouse. Variables such as dispersion, direction, and distance of the movement are very useful in identifying more precisely the underlying cognitive processes involved in reading or in accessing spatial memory. For example, we have shown that mouse movement is sensitive to the spatial memory for words in text involving different syntactic or referential representations (Baccino,1991; Baccino, 1994; Baccino & Pynte, 1994; Baccino, Pynte, & Kennedy, 1990). Even a subliminal effect, such as the refresh rate of VDU displays, appears to disrupt the mouse trajectory (Kennedy & Baccino, 1993). In the present article we describe a set of Turbo Pascal (Version 6.0) routines (called MICELAB) to be included in the main program (MouseTst) that illustrates how spatial data from mouse movement can be analyzed for experimental purposes. MICELAB programs may be used to analyze any mouse or trackball movements and may be included in your own program or used via the MouseTst software.

TURBO PASCAL UNITS

MICELAB Unit

The unit provides a list of procedures and functions involved in computing the spatial processing of mouse movement during a pointing task. The movement is recorded in an array of integers storing *x* and *y* coordinates of the mouse at a given time. These routines are explained below, according to the three spatial variables studied: trajectory dispersion, trajectory distance, and pointing accuracy. Trajectory dispersion indicates the degree with which the mouse deviates from the optimal trajectory (e.g., straight line). Trajectory distance gives the total distance of the movement. Both dispersion and distance account for uncertainty or error made by a subject during performance of the task. Pointing accuracy reveals the direction taken by the movement, indexing the motor program elaborated.

Trajectory Dispersion Algorithm

The dispersion of the mouse trajectory is calculated as the sum of the areas of the polygons whose boundaries include the line created by the mouse and the line of the optimal trajectory (i.e., from starting point to clicking point), as is shown in Figure 2.

Since the mouse trajectory includes irregular loops or vertical reversals that cut the optimal line, polygons of

Requests for reprints should be sent to the first author at Université de Nice Sophia-Antipolis, Laboratoire de Psychologie Expérimentale, BP 209. 98, Bd E.Herriot, F-06204 Nice Cedex 3, France (e-mail: lapex@frmop22.cnus.fr).

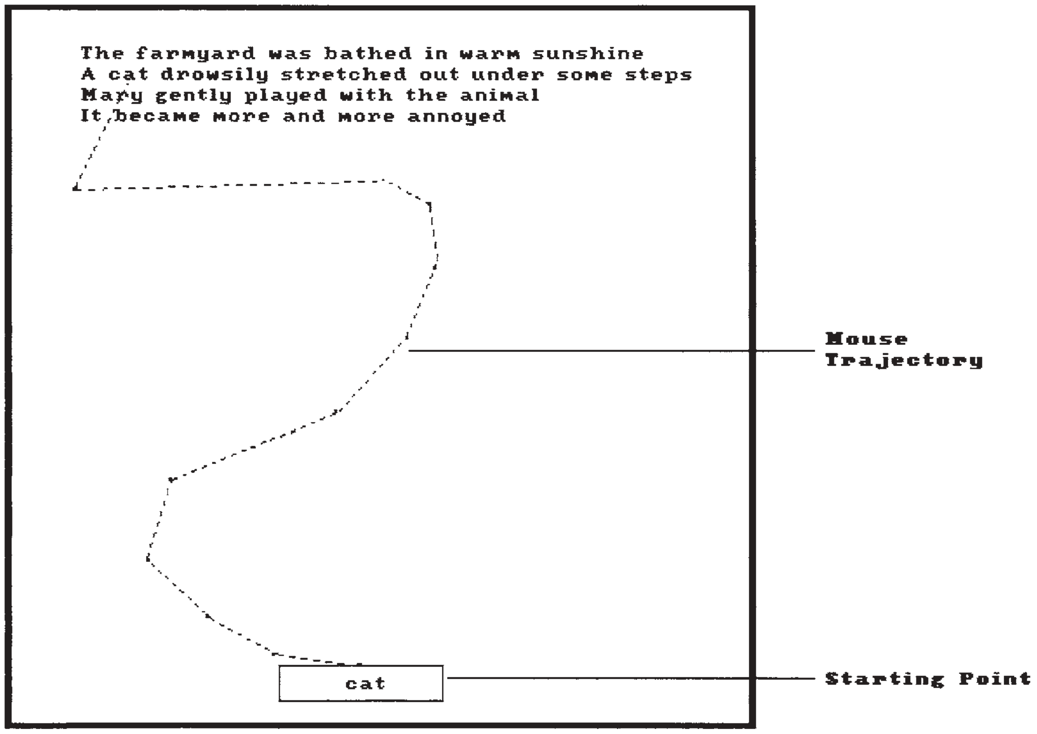
The farmyard was bathed in warm sunshine
A cat drowsily stretched out under some steps
Mary gently played with the animal
It became more and more annoyed
Mouse Trajectory
cat
Starting Point

Figure 1. Schematic diagram showing key elements of the display during a pointing task in which the subject moves the cursor from a starting point to a designated word. The dashed line corresponds to mouse trajectory.

this type are often complexes (because of the lack of convexity). It is necessary to compute the area in multiple stages, depending on whether the optimal trajectory has been crossed. The intersection point is then taken as a pivot vertex in order to calculate the next polygon. The area is computed by the Area function, which calls, respectively, the Intersegt procedure and the PolygonArea function, described below. The Intersegt routine tests whether an intersection occurs between each line segment (from Point I to Point I+1 of the coordinates array) and the optimal trajectory and, if it does, finds their point of intersection. If an intersection is found, the area of the polygon is calculated by calling the PolygonArea function from Point I to Point I+1. The last vertex of the polygon in this case is the intersection point with the optimal line. The same procedure is called for the following points until the end of the movement (clicking point). The dispersion is the sum of all polygon areas. The Area function requires the first and the last coordinate of the array as inputs and the total number of points recorded. The Intersegt procedure and PolygonArea function are described more precisely in the following text.

Intersegt procedure. Given two line segments (endpoints *a* and *b*, *c* and *d*), the procedure determines whether they intersect and, if they do, finds their point of intersection. The method first finds where the "parent" lines (P1, P2) of each line segment intersect (the parent line is the infinite line, of which the segment is a part) and then checks whether the intersection is within both line segments, as is shown in Figure 3. In mouse movement terms, the two line segments are drawn (1) from Point I to Point I+1, and (2) from starting point to clicking point.

For the parent lines to intersect, we need to solve the equation

$$D = (b_x - a_x)(d_y - c_y) - (b_y - a_y)(d_x - c_x),$$

where *a* and *b* represent the endpoints of Line Segment 1, and *c* and *d* denote the endpoints of Line Segment 2. If *D* is equal to zero, the parent lines are parallel. If *D* is *not* equal to zero, t_0 and u_0 are computed, where

$$t_0 = (c_x - a_x)(d_y - c_y) - (c_y - a_y)(d_x - c_x)/D$$

$$u_o = a_x + (b_x - a_x)\,t_0 - c_x/(d_x - c_x).$$

If t_0 lies outside the interval [0,1], there will be no intersection; the segment does not reach the other line. Otherwise, an intersection may exist, and so we compute u_0. If u_0 lies between 0 and 1, the segments will intersect

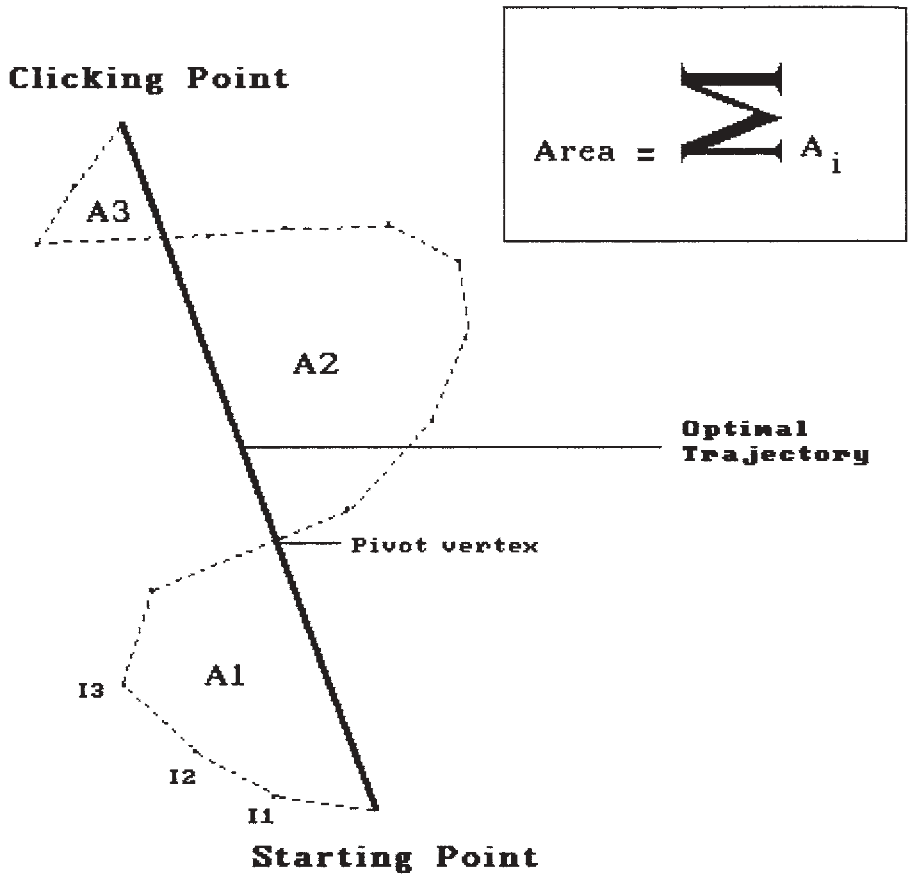

Figure 2. Analysis of the mouse movement from Figure 1. The movement is decomposed into multiple polygons (A1, A2, A3) whose boundaries include the mouse line and the line of the optimal trajectory.

and the point of intersection (I_x, I_y) is found according to the equations

$$I_x = a_x + (b_x - a_x)\, t_0$$

$$I_y = a_y + (b_y - a_y)\, t_0.$$

If D is equal to zero, it means the slopes are equal and the parent lines are parallel. But the segments might still overlap; we test this by the equation

$$\text{Lhs} = (b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x).$$

If Lhs (left-hand side) is sufficiently close to zero (such as 10^{-5}—an arbitrary value in the routine), the parent lines coincide and we find the two values t_c and t_d, where

$$t_c = (c_x - a_x)/(b_x - a_x)$$

$$t_d = (d_x - a_x)/(b_x - a_x).$$

There is an overlap, unless both t_c and t_d are less than 0 or greater than 1.

PolygonArea function. The area is computed by decomposing the polygon into a collection of triangles and summing the areas of these triangles, as is shown in Figure 4. The area of the triangle is easily found by calculating the cross product ($^1/_2\, x_i \times y_i$), where

$$x_i \times y_i = (x_1 y_2 - x_2 y_1) + (x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3).$$

It is necessary to keep the sign of the area in order to solve polygons that are not convex. The area of the polygon is determined by summing all the positive and negative triangle areas. Likewise, the overall area of the mouse trajectory is given by summing the absolute values of every polygon area.

Algorithmically, the polygon area (Area function) requires two steps:

1. Find the last vertex of the polygon by calling Intersegt (if no intersection is found, the last vertex is the clicking point).

2. Calculate the area of the polygon by summing the signed areas of the successive triangles (using the

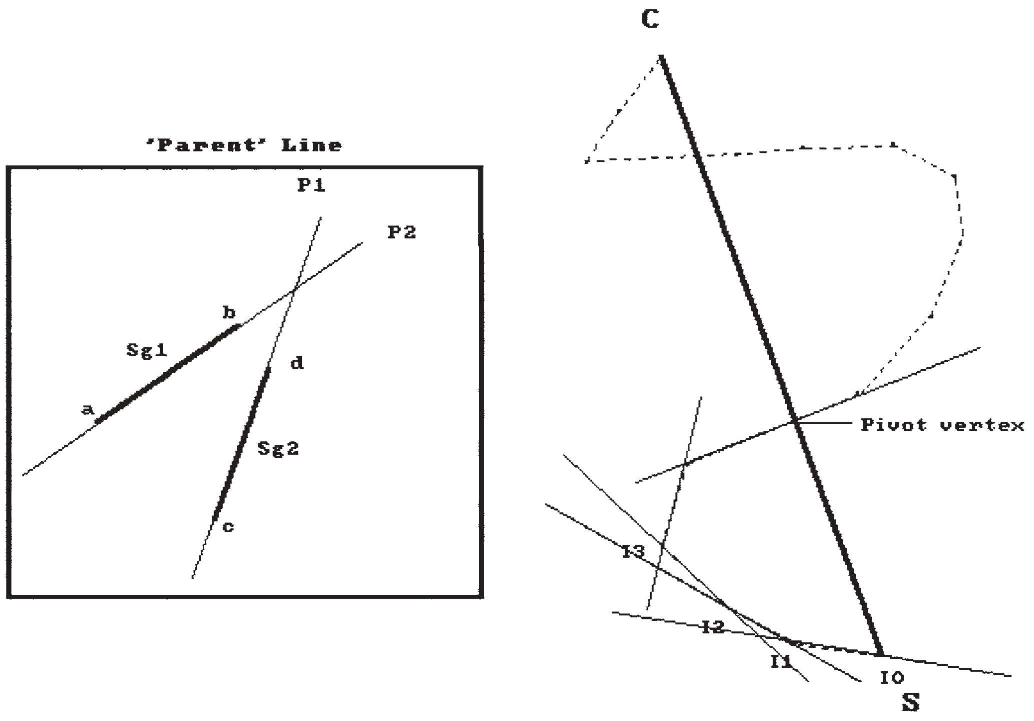

Figure 3. Representation of the "parent" line notion and Intersegt algorithm. The intersection between each line segment (from Point I to I+1) and the optimal line (SC) is tested until the end of the movement.

$$\text{Polygon Area} = \left| \sum_{i=1}^{M-2} SA_i \right|$$

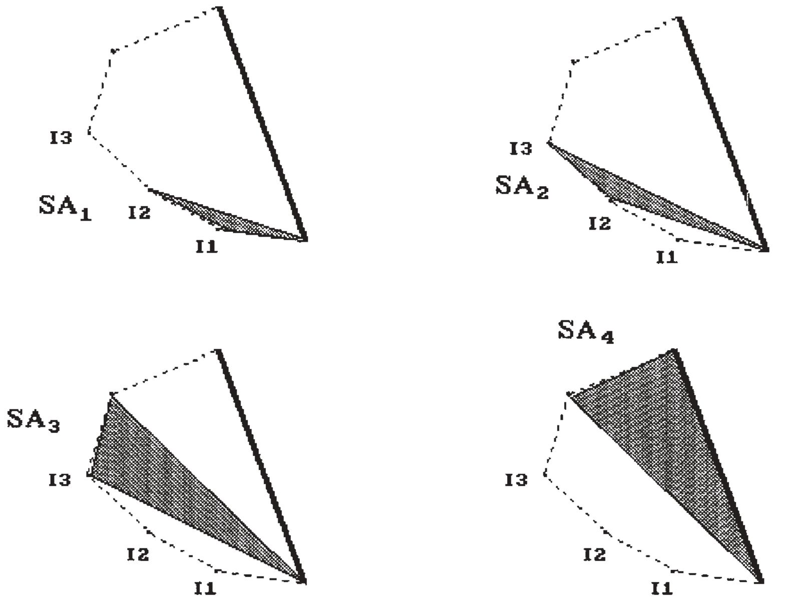

Figure 4. Representation of the PolygonArea algorithm. Each polygon area is found by decomposing it into a series of triangles and summing the signed areas (SA_i) of these triangles (e.g., $SA_1+SA_2+SA_3+SA_4$).

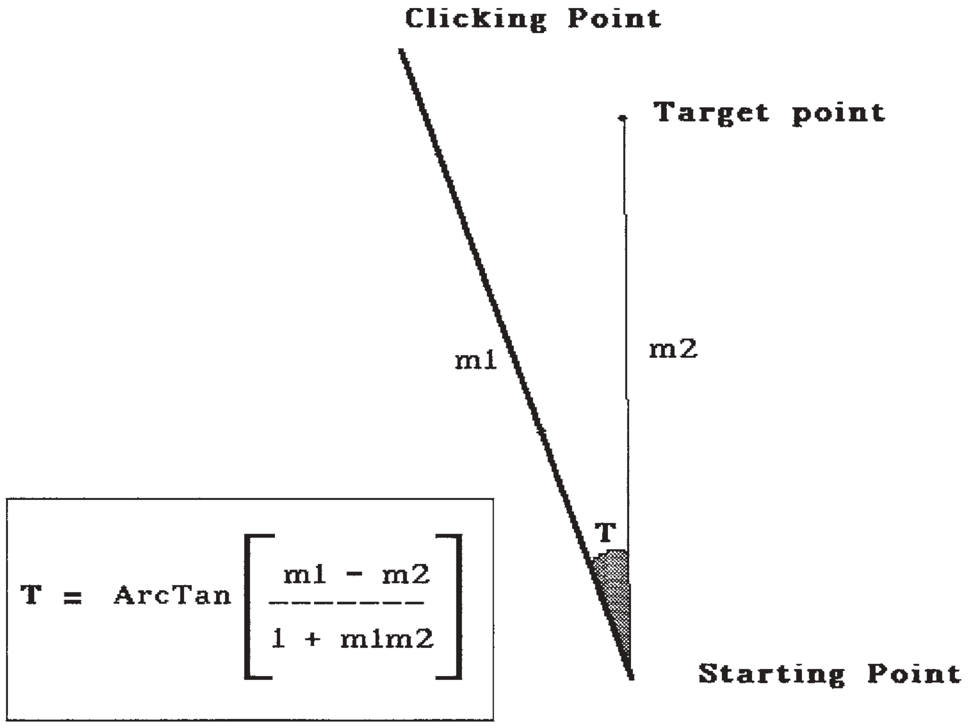

Figure 5. Angle of the pointing (clicking point, starting point, target point) corresponds to the angle between two lines given from their slopes (m_1, m_2).

CrossProduct function). Specifically, for the *i*th triangle, the signed area, SA_i, is

$$SA_i = 1/2\, \mathbf{a}_i \times \mathbf{a}_i + 1,$$

where $\mathbf{a}_i$ and $\mathbf{a}_i+1$ represent the vectors of two sides of the triangle and the area of the polygon is:

$$\text{area} = \left| \sum_{i=1}^{m-2} SA_i \right|.$$

The area is given in square pixels, and the dispersion of the movement is equal to the absolute values of all polygon areas.

Total Distance of the Trajectory

The total distance is the sum of the algebraic distances from each point (i) to the next ($i+1$). The algebraic distance (Dist_Alg function) between two points is given by the equation

$$d = \sqrt{(X_2 - X_1)2 + (Y_2 - Y_1)2}.$$

The Distance function requires the first and the last point (in pixels) from which the distance will be calculated. Since the system uses a VGA resolution, pixels have the same vertical and horizontal dimensions (i.e., a diameter of 0.28 mm). This precaution (same dimensions in width and height for a pixel) is meaningful whenever distances are used for comparative purposes. Similarly, good resolution allows for the detection of very small mouse movement, such as tremor.

Pointing Accuracy Algorithm

The pointing accuracy is calculated by the angle drawn between the target point and the clicking point from the starting point, as is shown in Figure 5. This variable, which gives the direction of the movement, is especially useful when the target is not visible (e.g., in memory experiments). For example, we have tested the reader's ability to locate with the mouse the position of a target word in previously read text. The direction of the movement is influenced by perceptual as well as linguistic factors of the text (Baccino & Pynte, 1994).

Angle function. First we find the slopes of each side of the triangle STC (starting point, target point, and clicking point). Given the endpoints of each side, the slope (Slope function) is easily calculated by the equation

$$m = b_y - b_x / a_y - a_x.$$

Then, the angle between two lines (AngInter function) is given from their slopes (m_1 and m_2) by using the equation

$$\tan\theta = \frac{m_1 - m_2}{1 + m_1 m_2},$$

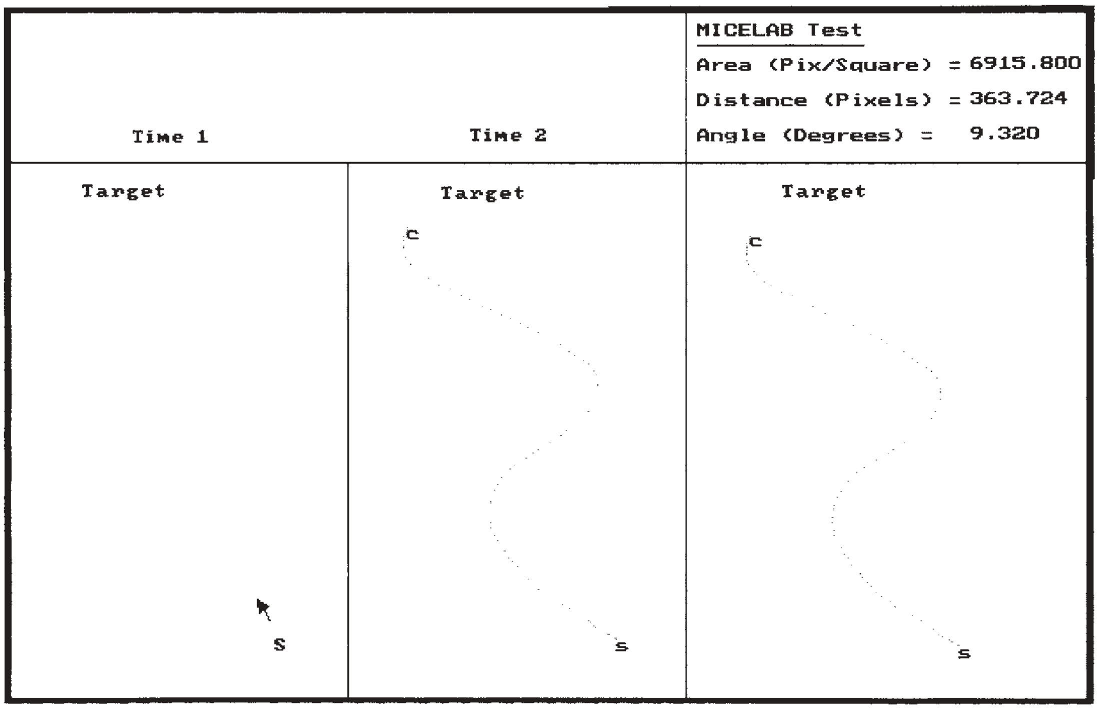
```
MICELAB Test
Area (Pix/Square) = 6915.800
Distance (Pixels) = 363.724
Angle (Degrees) =   9.320
```

Figure 6. Schematic diagram of the MouseTst program, illustrating a mouse-pointing task with the spatial analysis of the movement. Time 1, mouse cursor (in S) must be moved up to the target; Time 2, typical mouse movement (from s to c clicking point); Time 3, data analysis and final results on spatial variables.

and obviously the angle is equal to the arc tangent of θ (using the ArcTan built-in function). The angle initially calculated in radians is given in degrees by the following transformation (RadDeg function):

$$\text{AngleDegree} = (\text{AngleRadian} * 180.0)/\text{Pi}\,.$$

The FindSign procedure has been given in order to check the sign of every angle of the triangle STC. Angles of the same triangle must have the same sign, and their sum must equal 180º. If one angle out of three has a different sign, it must be subtracted from 180º if it is positive, or added to 180º if it is negative.

Mouse Unit

This unit (MouseU.pas) provides a set of procedures by which the mouse records buttonpresses, mouse movement, and mouse cursor locations similar to those previously described by Crosbie (1990). These routines are available in programming the Interrupt 33H. Initialization sets the mouse driver, and Hide_Curs and Show_Curs manipulate the cursor display. Cursor position is defined by Def_Position and returned by Ret_Position. Horiz_Bound and Vert_Bound for x,y axes, respectively, define the boundaries of the cursor movement on the screen. Since a graphics mouse has been used in this program, positions or boundaries are given in pixels.

MouseTst Program

The program MOUSETST.PAS uses elements of both of the algorithms described above to measure trajectory dispersion, trajectory distance, and pointing accuracy of the mouse movement. Init_Ecr_Graph_Auto is called to automatically set the graphics screen. The program displays a target (X) at a random position on the screen, and the task consists of moving the mouse cursor to the target. After the left button of the mouse has been pressed, the analysis of the pointing movement is given. MOUSETST.PAS is a very simple program, but it illustrates how the MICELAB unit can be used (see Figure 6).

Availability

The complete units (including source code) and an executable version of the demonstration program may be obtained from the first author by sending a self-addressed mailer and formatted floppy disk (any format).

REFERENCES

Baccino, T. (1991). *Spatial coding and text reading on VDU display*. Unpublished doctoral dissertation, University of Provence, Aix-en-Provence, France.

Baccino, T. (1994). Mouse movement control and spatial coding. *L'Année Psychologique,* **94**, 11-24.

Baccino, T., & Pynte, J. (1994). Spatial coding and discourse models during text reading. *Language & Cognitive Processes,* **9**, 143-155.

Baccino, T., Pynte, J., & Kennedy, A. (1990). *Spatial coding of word position in text during mouse operations*. Paper presented at the Fourth European Cognitive Psychology Conference, Como, Italy.

Beringer, J. (1992). Timing accuracy of mouse response registration on the IBM microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **24**, 486-490.

Crosbie, J. (1990). The Microsoft Mouse as a multipurpose response device for the IBM PC/XT/AT. *Behavior Research Methods, Instruments, & Computers*, **22**, 305-316.

Kennedy, A., & Baccino, T. (1993). *The effects of screen refresh rate on editing operations using a computer mouse pointing device*. Paper presented at the Seventh European Conference on Eye Movements, Durham, England.

Segalowitz, S. J., & Graves, R. E. (1990). Suitability of the IBM XT, AT, and PS/2 keyboard, mouse, and game port as response devices in reaction time paradigms. *Behavior Research Methods, Instruments, & Computers*, **22**, 283-289.

(Manuscript received August 30, 1993;
revision accepted for publication February 11, 1994.)