THE "HIGHLY INTELLIGENT" TABLET
AS AN EFFICIENT POINTING
DEVICE FOR INTERACTIVE GRAPHICS
(Preliminary Report)

Henry Fuchs
Zvi H. Kedem

University of Texas at Dallas
Richardson, Texas 75080

Described is a simple, efficient algorithm for determining the
nearest displayed point on a screen to an arbitrary cursor position.
The algorithm seems particularly appropriate for interactive systems
using a data tablet with a "smart" controller. The algorithm is
based on partitioning the screen among the currently displayed
points and minimizing this structure as points are added and
deleted. Finding the nearest point for cursor position consists
then of moving through this partitioning structure until the region
is determined. A divide-and-conquer method is used for both
inclusion testing in a particular region and also for speeding the
search for the proper nearest point.

Key Words: Cursor control; interactive computer graphics; nearest
neighbor.

1. Introduction

A basic, often encountered, problem
in interactive computer graphics concerns
the determination of the proper picture
point "pointed to" by the user-
controlled cursor. This seemingly simple
problem becomes increasingly burdensome
to the system as the scene complexity grows.
Naive solutions to this problem cause
significant time delays in the system
response. For users of these kinds of
systems delays of even a few seconds are
distracting to the overall design task.

Formally this problem can be stated
simply: Given a set of points on the
screen P = \{P_1, P_2, ..., P_n\} where
P_i = (x_i, y_i) and the cursor position,
Q = (x, y), find \( i_0 \) for which
d(Q, P_i) = \min_{i=1,2,...,n} d(Q, P_i)
where d(Q, P_i) is the distance from Q to P_i.

It may be important to observe certain characteristics of the
applications in which this problem is
encountered:

1. The computation of \( i_0 \) has to
be executed frequently for
changing values of Q with a
relatively stable set P.

2. The distance between two
consecutive values of Q for
which the problem is to be solved is
generally small relative to the
size of the screen.

3. Changes in P while relatively
infrequent, with respect to
changes in Q often occur
incrementally, i.e., certain P_i's
are deleted or moved, or new P_i's
are added.

4. Changes in P often occur in
clusters of points as an entire
graphical object may be moved,
deleted or inserted.

A straightforward approach requires
1(n) operations. Newman and Sproll
(1973), in their popular text, describe
a technique which utilizes a small
"window" around each P_i and then checks
whether Q is inside specific windows.
(Alternately a window can be constructed
around a cursor position.) This approach
suffers from serious limitations, e.g.,
the cursor may not lie in any square, or
if squares overlap it may lie in more than
one, thus requiring further operations to
find a solution. (Similarly with the
window drawn around the cursor point,
there may be none or many screen points
which lie inside it.)

Another approach currently used
involves marking selected data points as
the only ones which can be "touched" and
thus reducing the total number of points
to be considered.

*Present address of Henry Fuchs: Dept. of Computer Science,
University of North Carolina, Chapel Hill, NC 27514
This research was supported in part by NSF grant MCS-77-03905.

© 1978 ACM 0-89791-000-1/78/0012/0765 $00.75

This problem is related to one described by Knuth (1973) as the "post office problem." It involves preprocessing $P_i$'s in such a way that given $Q$, an appropriate $i$ is found efficiently. Solutions to this problem were given by Stamos (1975) and Lipton and Tarjan (1977). These solutions did not deal with the characteristics of our problem described above and thus more appropriate solutions may be devised to deal with it.

2. Outline of Solution

Our solution utilizes a structure in a n-dimensional Euclidean space introduced by Voronoi, described in Rogers (1964). This structure partitions the space into convex polyhedra. In our two-dimensional case it is a planar graph whose regions are convex polygons each containing exactly one of the $P_i$'s and containing exactly the points of the screen which are closer to this $P_i$ which is inside the polygon than to any other $P_i$. (See fig. 1.) (Stamos (1975) used the Voronoi structure to solve several related problems.)

![Voronoi structure](image)

**Figure 1. Voronoi structure**

Our solution involves three components:

a. determining closest point:
   - given a Voronoi structure for points on the screen $V(P_1,P_2,\ldots,P_n)$ and a cursor position $Q$, find the closest $P$;

b. adding a point:

given $V(P_1,P_2,\ldots,P_n)$ and a new point $P_{n+1}$, construct the new $V(P_1,P_2,\ldots,P_{n+1})$;

c. deleting a point:
   - given $V(P_1,P_2,\ldots,P_n)$ and an $i$, $1<i<n$, (a point to be deleted) construct $V(P_1,P_2,\ldots,P_{i-1},P_{i+1},\ldots,P_n)$.

We shall describe each of these parts in turn:

a. Determining closest point:

We assume here that the structure $V(P_1,P_2,\ldots,P_n)$ has already been created. (This can be accomplished either by iteratively applying part b) and thus creating $V(P_1),V(P_1,P_2),\ldots,V(P_1,\ldots,P_n)$ or applying an off-line algorithm such as the one described in Stamos (1975).) Our approach to determine closest point is to initially check whether the solution to the just-previous cursor position is still valid -- if not, interactively moving in toward the new closest $P_i$.

Specifically, the old closest point $P_i$ is still the closest point if and only if $Q$ is contained in $P_i$'s associated Voronoi polygon. (See fig. 2.)

![Voronoi structure with screen points and successive cursor positions](image)

**Figure 2. Voronoi structure with screen points and successive cursor positions**

To efficiently perform this surroundedness test we utilize a new algorithm for determining inclusion of a point in a convex polygon. (We shall from here on refer to this as an "inclusion test.") The algorithm is described more fully elsewhere (Kedem and Pach (1978));
we describe it here only as it relates to this problem.

The Voronoi polygon associated with a point \( P_k \) can be defined by a sequence (say, counter-clockwise) of vertices \( V_k = (v_1, v_2, \ldots, v_m) \). A straightforward method, as described in Sutherland, Sproull, and Shumacher (1974), to test for inclusion would involve testing \( Q \) against each line segment \( V_{i+1}V_i, i = 1, \ldots, m-1 \); and \( V_mV_1 \). \( Q \) would be inside \( P_k \) if and only if \( Q \) were on the left side of all these lines.

The number of tests, \( m \), can be significantly reduced by the following procedure: First test \( Q \) against line \( V_1V_2, h = (1+1)/2 \). (See fig. 3.) If \( Q \) is to the right of this line, then if it is inside \( P_k \) it must be inside the polygon defined by \( (v_1, v_2, \ldots, v_h) \). On the other hand, if \( Q \) is to the left of this line, then if it is inside \( P_k \) it must be inside the polygon defined by \( (v_1, v_{h+1}, \ldots, v_m) \).

\[ \text{Figure 3. Testing for inclusion in a Voronoi polygon (first step)} \]

Thus, in one step the problem is reduced to one which is approximately half the size of the original problem. The same divide-and-conquer approach is repeated. \( Q \) is tested against either \( V_1V_2, r = (1+2)/2 \) or \( V_{h+1}V_h, s = (h+2)/2 \). This process is repeated until only a "slice" of the original polygon \( V \) remains -- \( (v_1, v_{h+1}) \). This will take \( \log m+1 \) tests. At this point we know that \( Q \) is to the left of \( V \) and to the right of \( V_{h+1} \). (See Fig. 4.) A single test against \( V_{h+1} \) determines whether \( Q \) is indeed inside or outside the polygon \( V_k \). It is easy to see that such an approach is advantageous not only for large \( m \) (for \( m = 100 \), the number of tests is reduced to 7% of the original), but even for \( m \) as small as 4 -- in which case the number of tests is already reduced by 25%.

\[ \text{Figure 4. Last step in a inclusion test} \]

\[ \text{Figure 5. Determining next polygon for inclusion test} \]
Thus we can very efficiently determine whether or not the previous P is still the closest point. If it is not, then the inclusion test fails, but still yields a very important result—namely, that Q is inside the semi-infinite truncated wedge $<v_i, v_i+1>$ (See fig. 5.) A slightly different (untruncated) wedge results in the special case where Q lies either to the left or right of $v_i v_i+1$.

This indicates the direction of the polygon in which Q lies. Thus the next polygon tested is the other polygon which contains the edge $v_i v_{i+1}$. (Further, one edge of this new polygon has already been tested — $v_i v_{i+1}$). The remaining tests are performed as before. In this way the procedure "homes in" on the proper P. (See fig. 5.)

b. Adding a point:

Adding a new point, $P_{n+1}$, into the Voronoi structure is accomplished by first determining the closest point of Pj according to the just-described method (consider $Q = P_{n+1}$). The new Voronoi polygon is "carved out" around this new point Pj by the following sequence of operations:

Figure 6. Adding a point (first step)

The polygon $V_j$ is partitioned into two polygons by the perpendicular bisector of $P_j$ and $P_{n+1}$. The region in which P lies is the new Voronoi polygon for $P_j$. The other region, in which $P_{n+1}$ lies is the beginning of the Voronoi polygon for $P_{n+1}$. The rest of the polygon is constructed by combining regions acquired in a "traversal" of perpendicular bisectors around $P_{n+1}$ by the following method. Consider the part of the perpendicular bisector which lies within $V_j$. For a counter-clockwise traversal consider the orientation on this bisector line segment such the $P_{n+1}$ is on the left and $P_j$ is on the right. (See fig. 6.) The head of this arc intersects (touches, actually) an existing segment, $v_i v_{i+1}$, of the polygon of $P_j$. Call this point of intersection $v_i$ and let $P_k$ be the point associated with the polygon bounding of $v_i v_{i+1}$ (See again fig. 6.) It is easy to see that $v_i$ is equidistant from the three points $P_j$, $P_{n+1}$, and $P_k$. The next stage of the "carving out" of a polygon for $P_{n+1}$ consists of using the perpendicular bisector between $P_{n+1}$ and $P_k$ to divide the old polygon of $P_k$ between $P_{n+1}$ and $P_k$. The bisector will start at $v_i$, the end of the just previous perpendicular bisector. This procedure continues until the path of the perpendicular bisectors completes its circuit around $P_{n+1}$ and returns to the old polygon of $P_j$. (See fig. 7.)

Figure 7. Formation of Voronoi polygon for a new point

c. Deleting a point:

Deleting a point, say $P_d$, from a Voronoi structure is simply the reverse of adding one; a new Voronoi structure is constructed inside the polygon $V_d$, using only those points which share a common boundary with this polygon. The region acquired by each of these points is then appended to its previous polygon. (See fig. 6.)

We note that the algorithms to test for inclusion in a convex polygon and to
construct a new Voronoi polygon are optimal. The proof for the first one appears in Redem and Fuchs (1978) and the number of elementary steps in the latter is equal to the number of line segments in the generated Voronoi polygon.

3. Extensions

Extensions of the algorithms for 3D design have not yet been explored, but the generalization seems promising. The individual polygons would generalize to convex polyhedra, the perpendicular bisectors generalize to planar tiles and the simple traversals around a point to obtain a new polygon would generalize to an appropriate "non-deterministic flooding" of the adjacent polyhedra.

Further extensions are needed to take advantage of the fourth characteristic mentioned in the beginning of this paper; namely, that changes to the set of displayed points often occur in clusters as an entire graphical object is inserted, moved, or removed. Perhaps the Voronoi structure for the cluster of points could be constructed and then efficiently merged or extracted from the larger, full-screen structure.

4. Acknowledgements

Our initial interest in the topic of this paper was stimulated by discussions between one of us (RF) and Fred Parke of Case Western Reserve University.

References


