the first thing in the picture, say a point, and then to the next thing in the picture, say a line which terminated on the point. The normal macro mentioned in Chapter III for applying functions to all the members of a ring, LGORR, cannot be used, for at the time the next ring member is to be located, both it and the current ring member may be so much meaningless free storage. To delete everything in a picture, Sketchpad again and again deletes the first thing in the picture, thus chewing away until the picture is gone.

The push down list for recursive deletion is formed with the pair of registers which normally indicates what type of thing a block represents. As soon as it is found that a block must be deleted, it is declared "dead" by placing its TYPE pair in a generic ring called DEADS. The first dead thing is then examined to see if it forces other things to be declared dead, which is done until no more dead things are generated by the first dead thing. The first dead thing is then declared "free" and the new first dead thing is examined in exactly the same way until no more dead things exist. The DEADS ring, through registers which normally indicate type, serves as the push down list.

RECURSIVE MERGING

The single most powerful tool for constructing drawings, when combined with the definition copying function described in Chapter VII, is the ability to merge picture parts recursively. The recursive merge function makes it possible to make statements such as "this thing is to be related to that thing in such and such a way." The relationship may be treated as applying to things which it relates only indirectly. For example we shall soon see how one line may be made parallel to another even though the parallelism constraint applies only to the locations

Meta Platforms, Inc.
Exhibit 1140
Page 091

of their end points. Similarly, a set of digits can be forced to display the length of a line, even though the constraint involved refers to the end points of the line and the value of the digits rather than to the line or the digits themselves. The recursive merge function makes it meaningful to combine anything with anything else of the same type regardless of whether the things are dependent on other things or depended on by others.

If two things of the same type which are independent are merged, a single thing of that type results, and all things which depended on either of the merged things depend on the result* of the merger. For example, if two points are merged, all lines which previously terminated on either point now terminate on the single resulting point. In Sketchpad, if a thing is being moved with the light pen and the termination flick of the pen is given while aiming at another thing of the same type, the two things will merge. Thus, if one moves a point to another point and terminates, the points will merge, connecting all lines which formerly terminated on either. This makes it possible to draw closed polygons.

If two things of the same type which do depend on other things are merged, the things depended on by one will be forced to merge, respectively, with the things depended on by the other. The result* of merging two dependent things depends respectively on the results* of the mergers it forces. For example, if two lines are merged, the resultant line must refer to only two end points, the results of merging the pairs of end points of the

---

* The "result" of a merger is a single thing of the same type as the merged things.

Meta Platforms, Inc.
Exhibit 1140
Page 092

original lines. All lines which terminated on any of the four original end points now terminate on the appropriate one of the remaining pair. More important and useful, all constraints which applied to any of the four original end points now apply to the appropriate one of the remaining pair. This makes it possible to speak of line segments as being parallel even though (because line segments contain no numerical information to be constrained) the parallelism constraint must apply to their end points and not to the line segments themselves. If we wish to make two lines both parallel and equal in length, the steps outlined in Figure 6.1 make it possible. More obscure relationships between dependent things may as easily be defined and applied. For example, constraint complexes can be defined to make line segments be collinear, to make a line be tangent to a circle, or to make the values represented by two sets of digits be equal.

INSTANCES

The most powerful tool provided in the Sketchpad system for creating large complex drawings quickly and easily is the instance. Instances are recursively expanded so that instances may contain other instances to give an exponential growth of picture produced with respect to effort expended. Instances may have attachment points and therefore may connect points topologically much as line segments do. For example, an instance of a resistor may connect two points both electrically and geometrically on the drawing. An instance also has the properties of a four component variable: numbers are stored in each instance block to indicate where, how big, and in what rotation that instance is to appear on the picture. It took some time to reconcile the topological properties of instances with their properties as variables.

Meta Platforms, Inc.
Exhibit 1140
Page 093

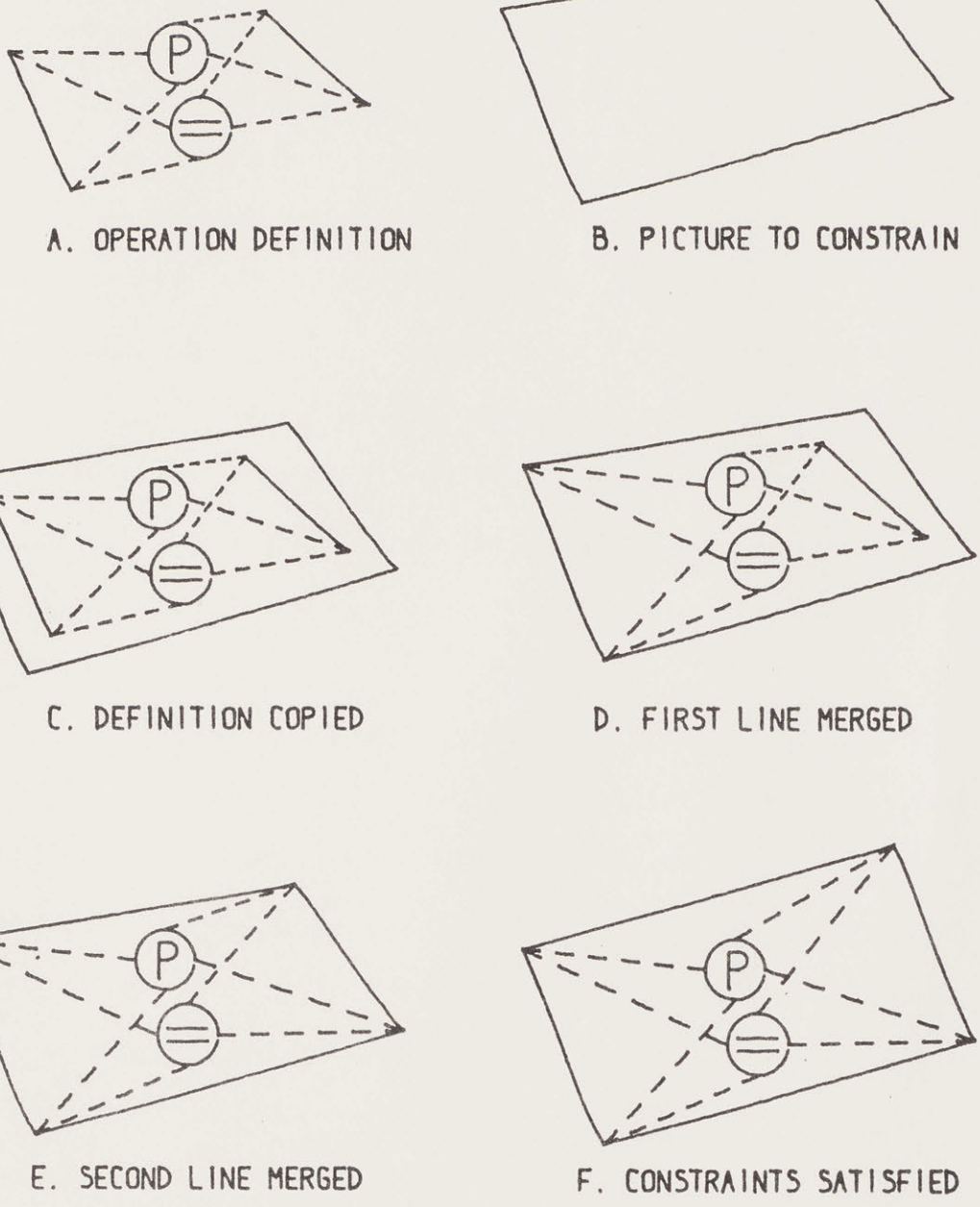


FIGURE 6.1. APPLYING TWO CONSTRAINTS INDIRECTLY TO TWO LINES

P PARALLELISM EQUAL LENGTH

Meta Platforms, Inc.
Exhibit 1140
Page 094

The block of registers which represents an instance is remarkably small considering that it may generate a display of any complexity. For the purposes of display, the instance block makes reference to a picture by means of its chicken in a ring which ties a picture to all its instances. The instance will appear on the display as a figure geometrically similar to the picture of which it is an instance but at a location, size, and rotation indicated by the four numbers which constitute the "value" of the instance. An important omission as this is written is the ability to make mirror images. Right and left handed figures must now be treated separately, whereas the instance should indicate whether a right or left handed version of the master is to be shown.

INSTANCES AS VARIABLES

The four numbers which specify the size, rotation, and location of the instance are considered numerically as a four dimensional vector. In certain computations, the value of a variable is changed "as little as possible" if there is no need to change it further. The distance measured in the case of instances is the square root of the sum of the squares of the four components. For this reason, and for simplicity in the use of the fixed point arithmetic of the TX-2, it is important that the four numbers used to represent the vector be of about the same order of magnitude. The particular numbers chosen are the coordinates of the center of the instance and the actual size of the instance as it appears on the drawing times the sine and cosine of the rotation angle involved. In a typical drawing these four numbers have reasonably similar ranges of variation.

Meta Platforms, Inc.
Exhibit 1140
Page 095

In our early work we attempted to use the position and the sine and cosine of the rotation angle times the reduction in size from the master picture in order to avoid the normalization of master picture size implicit in the above paragraph. This not only prevented having instances larger than their masters because of the fixed point arithmetic, but also made distance in the four dimensional space meaningless. No attempt was ever made to use the size and rotation numbers independently.

The transformations of coordinates represented by the above paragraphs are:

Poor

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} i_1 & i_2 \\ -i_2 & i_1 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix} + \begin{bmatrix} i_3 \\ i_4 \end{bmatrix} \qquad (6\text{-}1)$$

Better

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} i_1 & i_2 \\ -i_2 & i_1 \end{bmatrix} \begin{bmatrix} x_m/s_m \\ y_m/s_m \end{bmatrix} + \begin{bmatrix} i_3 \\ i_4 \end{bmatrix} \qquad (6\text{-}2)$$

where:

x_d, y_d = Display location in page coordinates.

x_m, y_m = Master location in page coordinates.

s_m = Size of master picture in page coordinates.

$i_1 \,..\, i_4$ = 4 vector in instance, $-1 < i_i < +1$.

Meta Platforms, Inc.
Exhibit 1140
Page 096

RECURSIVE DISPLAY OF INSTANCES

In displaying an instance of a picture, reference must be made to the picture itself to find out what picture parts are to be shown. The picture referred to may contain instances, however, requiring further reference, and so on until a picture is found which contains no instances. A recursive program performs this function. At each stage in the recursion, any picture parts displayed must be relocated so that they will appear at the correct position, size and rotation on the display. Thus, at each stage of the recursion, some transformation of the form of Equation (6-2) is applied to all picture parts before displaying them. If an instance is encountered, the transformation represented by its value must be adjoined to the existing transformation for display of parts within it. When the expansion of an instance within an instance is finished, the transformation must be restored for continuation at the higher level.

To avoid the difficulties of taking an inverse transformation, the old transformation is saved in registers provided for that purpose in the picture block of the picture being expanded. Thus, the current transformation is stored in program registers and is being used, whereas the previous transformation is saved in the picture block currently being expanded. The push down list is provided also by indicating in the picture block being expanded the particular instance thereof which is responsible for this expansion of the picture. The first picture to be displayed starts with no transformation at all. Thus, if it contains itself as an instance, one recursion is possible, saving the old transformation in the picture block and saving the address of the instance responsible for the expansion in the picture block as well. Subsequent recursions will be prevented, however, because no instance is expanded

Meta Platforms, Inc.
Exhibit 1140
Page 097

if the picture of which it is an instance already belongs on the push down list. It would be possible to expand such circular instances further by providing some suitable termination condition such as reaching a level too small to show on the display. However, since the instances might get larger rather than smaller, termination conditions are far from simple.

ATTACHERS AND INSTANCES

Many symbols used must be integrated into the rest of the drawing by attaching lines to the symbols at appropriate points, or by attaching the symbols directly to each other as if by zero length lines. For example, circuit symbols must be wired up, geometric patterns made by fitting shapes together, or mechanisms composed of links tied together appropriately. An instance may have any number of tie points, and, conversely, a point may serve as tie for any number of instances.

An "instance-point" constraint block is used to relate an instance to each of its tie points. An instance-point constraint is satisfied only when the point bears the same relationship to the instance that a point in the master picture for that instance bears to the master picture coordinate system. Instance-point constraints are treated as a special case when an instance is moved so that tie points always move with their instance, and lines terminating on the tie points move as well. Each instance-point constraint makes reference to both the instance and its tie points by means of chickens.

To use a point as an attacher of an instance, the point must be designated as an attacher in the master drawing of the instance. For example, when one first draws a resistor, the ends of the resistor must

Meta Platforms, Inc.
Exhibit 1140
Page 098

be designated as attachers if wiring is to be attached. When an instance is created by pressing the "instance" button, toggle switches tell what picture the instance is to refer to. Along with the instance element are created a point and an instance-point constraint for each attacher. These points are bonifide points in the object picture but are not automatically attachers of the object picture. If they are to be used as attachers when the object picture is instanced, they must be designated anew. Thus of the three attachers of a transistor it is possible to select one or two to be the attachers of a flip-flop.

The entire internal structure of the instance is suppressed as far as the light pen is concerned except for the attachers. Thus even on a dense circuit drawing it is possible to connect elements with ease because at the highest level of instance only the designated attachers will hold the attention of the light pen program. Usually there are only a few attachers for each block no matter how complicated internally, and so it is generally obvious which one to use.

RECURSIVE MOVING

At first only variables could be moved. Moving a variable means to change somehow the numbers stored as the components of the variable, usually to make the display for the variable follow light pen motions. A moving point, for example, will be firmly attached to the pseudo pen position, while a moving piece of text faithfully follows light pen displacements so that the part of the text which was under the pen when the "move" button was pressed remains under the pen. For variables with more than two components, moving is partly controlled by the pen and partly by knobs. For example, the moving text can be made larger or

Meta Platforms, Inc.
Exhibit 1140
Page 099

rotated by two of the knobs.

The advent of the recursive merging and the definition copying functions made it clear that one should be able to move anything regardless of whether or not it is variable. To move a non-variable, a recursive process is used to find whatever variables may be basic to the thing being moved. For example, if a line is to be moved, the end points on which it depends must be moved. All objects which are being moved are put in a ring whose hen is in the MOVINGS generic block. The object actually attached to the light pen is first in the ring. Upon termination only this first object in the MOVINGS ring may be merged with other objects.

The numerical operation of moving is accomplished by the standard transformation procedure. The small transformation due to light pen position change and knob rotation since the last program iteration is converted to the form of Equation (6-2) and placed in the standard location. Each object in the MOVINGS ring is transformed by it. The generic block for each type of object, of course, contains the subroutine to apply the transformation to such objects. The generic block for lines, for example, indicates that no transformation need be applied to the line because it contains no numerical values and will automatically be moved when its end points are moved.

Moving objects must be invisible to the light pen. Since the light pen aims at anything within its field of view, it would otherwise aim at a moving object and a jerky motion would result. Motion would only happen when the pen's field of view passed beyond the object being moved. Moreover, the display for moving objects must be recomputed regularly for the benefit of the human user, but the unmoving background need not be recomputed. The display spot coordinates for objects being recomputed

Meta Platforms, Inc.
Exhibit 1140
Page 100

is placed last in the display file, above (in higher numbered registers) the fixed background display so that it may be recomputed without disturbing the rest of the display file. The light pen program rejects any spots seen by the pen which come from these high display file locations. Needless to say, the entire display file must be recomputed once to eliminate the former traces of the newly moving objects.

Meta Platforms, Inc.
Exhibit 1140
Page 101

Chapter VII

BUILDING A DRAWING, THE COPY FUNCTION

As experimentation with drawing systems for the computer progressed, the basic drawing operations evolved into their present form. At the outset, the very general picture and relationship defining capability of the copy and recursive merging functions were unknown and so considerable power had to be built directly into the system. Now, of course, it would be possible to use much simpler atomic operations to draw simple pictures embodying many of the notions now treated as atomic.

DRAWING VS. MOVING

An idea that was difficult for the author to grasp was that there is no state of the system that can be called "drawing." Conventionally, of course, drawing is an active process which leaves a trail of carbon on the paper. With a computer sketch, however, any line segment is straight and can be relocated by moving one or both of its end points. In particular, when the button "draw" is pressed, a new line segment and two new end points are set up in storage, and one of the line's end points is left attached to the light pen so that subsequent pen motions will move the point. The state of the system is then no different from its state whenever a point is being moved.

Similarly, to draw a circle, one creates a center point when the button "circle center" is pressed, and creates in the ring structure a circle block and its start and end points when the button "draw" is pressed with a circle center defined. The end point of the circle arc

Meta Platforms, Inc.
Exhibit 1140
Page 102

is left attached to the light pen to move with subsequent pen motions. Since the start and end points of a circle arc should be equidistant from its center point, an equal distance constraint is created along with the circle but could be subsequently deleted without deleting the circle.

ATOMIC OPERATIONS

In general, when creating new points to serve as the start of line segments and circle arcs or centers for circle arcs, an existing point is used if the pen is aimed at one when the new point would be generated. Thus, if one aims at the end of an existing line segment and presses "draw" the new line segment will use the existing point rather than setting up another point which has the same coordinates. Later motion of this point will move both lines attached to it; the ring structure storage reflects the intended topology of the drawing. Similarly, if one is moving a point and gives a termination signal while aiming at another point, these two points will be merged, again reflecting the intended drawing topology.

We have seen that a constraint is set up to indicate that the start and end points of a circle arc should be equidistant from its center whenever a new circle arc is drawn. Similarly, constraints to indicate that a point should lie on a line or circle are automatically set up if a point is either created while the pen is pointing to the line or circle or moved onto the line or circle. The constraints, of course, do not apply to the line or circle itself but to the points on which it depends. If the light pen is aimed at the intersection of line segments,

Meta Platforms, Inc.
Exhibit 1140
Page 103

two "point-on-line" constraints will be set up for a point created or left there, one for each intersecting line. Three or more line segments may be forced to pass through a single point by moving that point onto them successively to set up the appropriate constraints. Constraint satisfaction will then move the lines so that all of them pass through the point. In order to avoid cluttering up the ring structure with redundant constraints, the point-on-line and point-on-circle constraints are set up only if the point is not already so constrained.

GENERALIZATION OF ATOMIC OPERATIONS

The atomic operations described above make it possible to create in the ring structure new picture components and relate them topologically. The atomic operations are, of course, limited to creating points, lines, circles, point-on-line and point-on-circle constraints. (The point-on-circle constraint is the same type as used to keep the circle's start and end points equidistant from its center.) Since implementation of the copy function it has become possible to create any combination of picture parts and constraints in the ring structure. The recursive merging function makes it possible to relate this set of picture parts to any existing parts. For example, if a line segment and its two end points are copied into the object picture, the action of the "draw" button may be exactly duplicated in every respect. Along with the copied line, however, one might copy as well a constraint to make the line horizontal, or two constraints to make it both horizontal and three inches long, or any other variation one cares to put into the ring structure to be copied.

Meta Platforms, Inc.
Exhibit 1140
Page 104

When one draws a definition picture to be copied, certain portions of it to be used in relating it to other object picture parts are designated as "attachers". Anything at all may be designated: for example, points, lines, circles, text, even constraints! The rules used for combining points when the "draw" button is pressed are generalized so that:

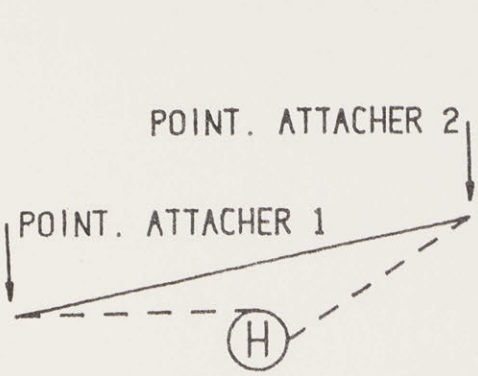
> For copying a picture, the last-designated attacher is left moving with the light pen. The next-to-last-designated attacher is recursively merged with whatever object the pen is aimed at when the copying occurs, if that object is of like type. Previously designated attachers are recursively merged with previously designated object picture parts, if of like type, until either the supply of designated attachers or the supply of designated object picture parts is exhausted. The last-designated attacher may be recursively merged with any other object of like type when the termination flick is given.

Normally only two designated attachers are used because it is hard to keep track of additional ones. The order in which attachers are designated is important because it is in this order that they will be treated. If a mistake is made in ordering the attachers, redesignation of an attacher puts it last in the order. As this is written there is no way to undesignate an attacher, except by deleting it, an oversight which should be corrected.

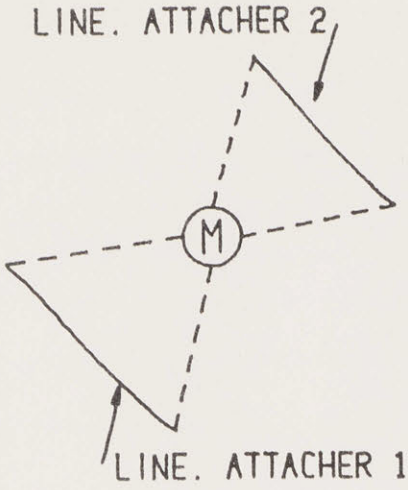
If the definition picture to be copied consists of a line segment with end points as attachers and a horizontal constraint between the end points, as shown in Figure 7.1A, pressing the "copy" button will appear to the user exactly like pressing the "draw" button. One end point of the line will be left behind and one will follow the light pen. Subsequent constraint satisfaction will, however, make the line horizontal.

If the definition picture consists of two line segments, their four end points, and a constraint on the points which makes the lines equal

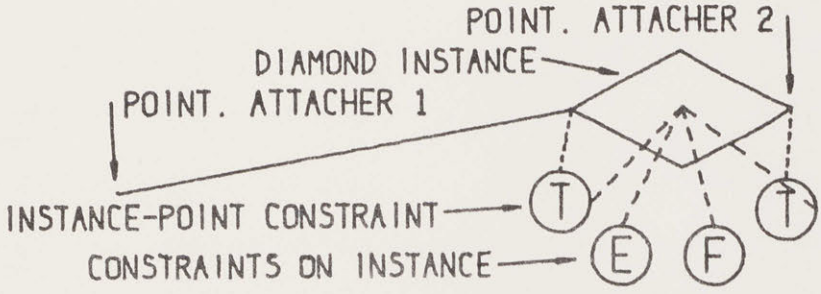
Meta Platforms, Inc.
Exhibit 1140
Page 105



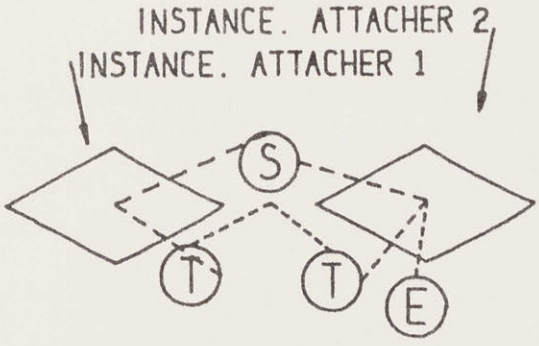
A. HORIZONTAL LINE



B. EQUAL LENGTH LINES



C. PARTLY FLEXIBLE ARROW



D. PRE-JOINED INSTANCES

FIGURE 7.1. DEFINITIONS TO COPY

Meta Platforms, Inc.
Exhibit 1140
Page 106

in length, with the two lines designated as attachers as shown in Figure 7.1B, copying enables the user to make any two lines equal in length. If the pen is aimed at a line when "copy" is pushed, the first of the two copied lines merges with it, (taking its position and never actually being seen). The other copied line is left moving with the light pen and will merge with whatever other line the pen is aimed at when termination occurs. Since merging is recursive, the copied equal-length constraint will apply to the desired pair of object picture lines. If no lines are aimed at, of course, the copied picture parts are seen at once with the scale factor so reduced that the entire copied picture takes up about 1/16 of the display area.

If the picture to be copied consists of the erect constraint and the full size constraint, both applying to a single dummy variable which is the attacher, copying produces a useful constraint complex attached to the pen for subsequent application to any desired instance. With only one attacher, the instance constrained is the one the pen is aimed at when termination occurs.

COPYING INSTANCES

As we saw in Chapter VI the internal structure of an instance is entirely fixed. The internal structure of a copy, however, is entirely variable. An instance always retains its identity as a single part of the drawing; one can only delete an entire instance. Once a definition picture is copied, however, the copy loses all identity as a unit; individual parts of it may be deleted at will.

One might expect that there was intermediate ground between the fixed-internal-structure instance and the loose-internal-structure copy.

Meta Platforms, Inc.
Exhibit 1140
Page 107

One might wish to produce a collection of picture parts, some of which were fixed internally and some of which were not. The entire range of variation between the instance and the copy can be constructed by copying instances.

For example, the arrow shown in Figure 7.1C can be copied into an object picture to result in a fixed-internal-structure diamond arrowhead with a flexible tail. As the definition in Figure 7.1C is set up, drawing diamond-arrowheaded lines is just like drawing ordinary lines. One aims the light pen where the tail is to end, presses "copy" and moves off with an arrowhead following the pen. The diamond arrowhead in this case will remain horizontal.

Copying pre-joined instances can produce vast numbers of joined instances very easily. For example the definition in Figure 7.1D, when repetitively copied, will result in a row of joined, equal size diamonds. In this case the instances themselves are attachers. Although each press of the "copy" button copies two new instances into the object picture, one of these is merged with the last instance in the growing row. In the final row, therefore, each instance carries all the constraints which were applied to either of the instances in the definition. This is why only one of the instances in Figure 7.1D carries the erect constraint. Notice also that although the diamond is normally a two-attacher instance, each of the diamonds in Figure 7.1D carries only one attacher. The other has been deleted so that each instance in the final row of diamonds will obtain only one right and one left attacher, one from each of the copied instances.

Meta Platforms, Inc.
Exhibit 1140
Page 108

THE MECHANICS OF COPYING

Needless to say, when a piece of ring structure is copied the definition picture used is not destroyed; the copying procedure reproduces its ring structure elsewhere in memory. However, the reproduction is not just a duplication of the numbers in some registers. The parts of the definition drawing to be copied may be topologically related, and the parts of the copy must be related to each other in the same way rather than to the parts of the master. Worse yet, some parts of the definition may be related to things which are not being copied. For example, an instance is related to the master picture of which it is an instance, and the copy of the instance must be related to the same master picture, not to a copy of it.

To copy a picture, space to duplicate all the elements of the picture is allocated in the free registers at the end of the ring structure. Each of the new elements is tied into its appropriate generic block ring by its TYPE component. Each new element is placed in this ring adjacent to the element it is a copy of. That is, for each element in the master a duplicate element is set up adjacent to it in the generic ring for that type of element. Appropriate scaled values are given to copied variables. The various references in the definition elements are then examined to see whether they refer to things that have been copied. If they do, the corresponding components of the copied elements are made to refer to the appropriate copied elements. On the other hand, if a definition element refers to something which has not been copied, its copy refers to the same element that its definition does.

When the complete copy has been made, the copies of all but the last-designated of the attachers are recursively merged with the designated

Meta Platforms, Inc.
Exhibit 1140
Page 109

portions of the object picture. The last-designated attacher is fastened to the light pen with the recursive moving function. The last-designated attacher may later on merge with another picture part.

Meta Platforms, Inc.
Exhibit 1140
Page 110

Chapter VIII

CONSTRAINT SATISFACTION

The major feature which distinguishes a Sketchpad drawing from a paper and pencil drawing is the user's ability to specify to Sketchpad mathematical conditions on already drawn parts of his drawing which will be automatically satisfied by the computer to make the drawing take the exact shape desired. For example, to draw a square, any quadralateral is created by sloppy light pen manipulation, closure being assured by the pseudo light pen position and merging of points. The sides of this quadralateral may then be specified to be equal in length and any angle may be required to be a right angle. Given these conditions, the computer will complete a square. Given an additional specification, say the length of one side, the computer will create a square of the desired size.

The process of fixing up a drawing to meet new conditions applied to it after it is already partially complete is very much like the process a designer goes through in turning a basic idea into a finished design. As new requirements on the various parts of the design are thought of, small changes are made to the size or other properties of parts to meet the new conditions. By making Sketchpad able to find new values for variables which satisfy the conditions imposed, it is hoped that designers can be relieved of the need of much mathematical detail. The effort expended in making the definition of constraint types as general as possible was aimed at making design constraints as well as geometric constraints equally easy to add to the system. To date,

Meta Platforms, Inc.
Exhibit 1140
Page 111

however, Sketchpad is more of a model of the design process than a complete designer's aid both because it is limited to two dimensions and because little advanced application has as yet been made of it.

The work on constraint satisfaction has been successful as far as it has been taken. The constraint definition and satisfaction programs generalize easily to three dimensions; in fact, constraint satisfaction for instances is even now treated as a four dimensional problem. The high speed maze solving technique for constraint satisfaction described below works well where constraints have been specified unredundantly. There is much room for improvement in the relaxation process and in making the "intelligent" generalizations that permit humans to capitalize on symmetry and eliminate redundancy.

DEFINITION OF A CONSTRAINT TYPE

Each constraint type is entered into the system as a generic block indicating the various properties of that particular constraint type. Generic blocks for constraints need not be given symbolic programming names since virtually no reference is made to particular constraint types in the program. The generic block tells how many variables are constrained, which of these variables may be changed in order to satisfy the constraint, how many degrees of freedom are removed from the constrained variables, and a code letter for human reference to this constraint type.

Any number of variables may be related by a constraint, but the display for constraints (see Chapter V) will be ambiguous if more than four variables are indicated, and so no constraints relate more than four variables. Of these variables, some may be referenced only.

Meta Platforms, Inc.
Exhibit 1140
Page 112

The routine which satisfies the constraint by changing the values of some of the variables is forbidden to satisfy the constraint by changing a "for reference only" variable. For example, a constraint could be implemented which would make its first variable equal to its second by changing the first to match the second, but not the reverse. This kind of one-way constraint is useful because it speeds up the relaxation procedure by forcing re-evaluation of variables in a specified order. For example, the constraint which makes the value of a number equal to the change in length of a bridge beam, thus indicating the force carried by the beam, is one way. It would be pointless to have an erroneous value of the indicator affect in any way the relaxation procedure for the bridge. Again, the constraint which relates a point to an instance in such a way that the point maintains the same relationship to the instance that an original point in the master picture had to the master picture, uses the original point "for reference only" to discover just what the correct relationship is. Thus the end terminal on a resistor will always stay at the end of the resistor. It would be out of keeping with the fixed geometry nature of instances to have the internal details of the instance changed to make it fit into some awkward position.

The one-way type constraint, however, can lead to instabilities in the constraint satisfaction procedure. For example, if two scalars were each specified to be twice the value of the other, with reference only made to the smaller,

$$A \rightarrow 2B$$
$$B \rightarrow 2A, \qquad (8\text{-}1)$$

Meta Platforms, Inc.
Exhibit 1140
Page 113

both variables would grow without bound, assuming, each iteration, values four times as big as before. If, however, a similar condition were set up with normal two-way constraints, the values of the variables would approach zero, a correct and stable result. Since the number of one-way constraints is small and they are designed for and used in special applications only, very little instability trouble of this kind has been observed. Future users who add one-way constraints, however, are warned to be cautious of the instabilities which may result.

NUMERICAL DEFINITION OF CONSTRAINTS

After the first stumblings of trying to define a constraint type in terms of the equations of lines along which the constrained variables should lie to satisfy the constraint, the numerical definition of constraints directly in terms of an error was devised. By using an error definition and considering the square of the error as an energy, one not only reflects directly the intent of the relaxation process, but also makes it easy to write the defining subroutines for new constraint types.

The defining subroutine for a constraint type is a subroutine which will compute, for the existing values of the variables of a particular constraint of that type, the error introduced into the system by that particular constraint. For example, the defining subroutine for making points have the same x coordinate (to make a line between them vertical) computes the difference in their x coordinates. What could be simpler? The computed error is a scalar which the constraint satisfaction routine will attempt to reduce to zero by manipulation of the constrained variables. The computation of the error may be non-linear or time dependent, or it may involve parameters not a part of the drawing such

Meta Platforms, Inc.
Exhibit 1140
Page 114

as the setting of toggle switches, etc. The flexibility of computation subroutines for defining constraints is tremendous.

In order to avoid overflow difficulties, the partial derivative of the error with respect to the value of any of the components of a constrained variable must be less than two. In order to make the constraints work well together, it is necessary that they be balanced, that is that the partial derivative of error with respect to displacement be nearly equal for all constraint types. I have arbitrarily tried to make the error subroutines compute an error about proportional to the distance by which a variable is removed from its proper position. In other words, many of the existing constraint computation subroutines make the partial derivative about unity.

LINEARIZATION OF CONSTRAINTS

The method of finding the least mean squares fit to a group of constraints described below requires that a linear equation be given for each constraint. To find the linear equation which best approximates the possibly non-linear constraint for the present values of the variables, the error computed by the subroutine is noted for several slightly different values of the variables. The equation,

$$\sum \frac{\Delta E}{\Delta x_i} (x_i - x_{io}) = -Eo, \qquad (8\text{-}2)$$

where x_i are the components of the variable, E is the computed error, and subscript o denotes intial value, is used as the linear best fit. Actually, the coefficients computed are 1/2 the values shown in equation (8-2) to permit error to be equal to displacement without generating overflow.

Meta Platforms, Inc.
Exhibit 1140
Page 115

Some constraints may remove more than one degree of freedom from the variables constrained. For example, the constraint which locates one thing exactly mid-way between two others removes two degrees of freedom. Such constraints must have as many error computation subroutines as there are degrees of freedom lost, since each subroutine results in a single linear equation. A subroutine which computes the distance from a variable to its correct location without regard to the number of degrees of freedom being removed will cause erratic results. A correct subroutine pair for constraining one thing to lie between two others computes both how far out of line the center thing is and, separately, 1/2 the difference in the distances from the center object to the two outer ones (1/2 is put in to meet the maximum derivative requirement).

THE RELAXATION METHOD

When the one pass method of satisfying constraints to be described later on fails, the Sketchpad system falls back on the reliable but slow method of relaxation to reduce the errors indicated by the various computation subroutines to smaller and smaller values. For simple constructions such as the hexagon illustrated in Figure 1.5, page 15 the relaxation procedure is sufficiently fast to be useful. However, for complex systems of variables, especially directly connected instances, relaxation is unacceptably slow. Fortunately, it is for just such directly connected instances that the one pass method shows the most striking success.

The relaxation method of satisfying conditions is as follows:

> Choose a variable. Re-evaluate it to reduce the total error introduced by all constraints in the system. Choose another variable and repeat.

Meta Platforms, Inc.
Exhibit 1140
Page 116

Note that since each step makes some net reduction of total error, there will be monotonic decrease of error and thus stability is assured. Since re-evaluating a variable will change only the error introduced by the constraints which apply to that variable, only the changes in the errors introduced by these constraints need be considered. Other variables and therefore the errors of constraints applying only to them will remain constant. Sketchpad's ring structure makes it easy to consider all constraints applying to a particular variable since all such constraints are collected together in a ring whose "hen" is in the variable.

It is important in the relaxation method that, at each step, the very latest computed values of all variables be used for error computations. From the point of view of the program, this means that only one value for each variable need be stored, each being updated in turn. Former values not only may, but must be discarded. It is also important that the change in error obtained by completely satisfying a constraint by moving one of its variables be identical to to the change in error to be obtained by completely satisfying it by moving another of its variables. The error computing subroutine definition for a constraint computes the same error for a constraint no matter which of its variables is to be moved. My original instability troubles with constraint satisfaction came from insufficient care in meeting this condition.

LEAST MEAN SQUARES FIT TO LINEARIZED CONSTRAINTS

In implementing the relaxation method above, it is important to be able to find quickly a new value for a variable which reduces the total error introduced by the constraints on that variable. In particular, the linearized form of the constraints results in a set of linear equations

Meta Platforms, Inc.
Exhibit 1140
Page 117

for the variable each of which must be met as closely as possible. Unfortunately, there may be any number of linear equations applying to a particular variable and these may be either independent but incomplete, independent and complete, or redundant and overdefining. A general arithmetic macro, SOLVE, for finding the best value for a set of equations has been devised.

SOLVE converts the given equations into an independent set of equations whose solution will be a point of minimum mean squared error for the original set. It is not always possible to solve the independent set of equations uniquely, and if it is not, SOLVE finds that solution which results in the minimum change from the existing value of the variable. The mathematical discussion pertinent to SOLVE is given in Appendix F. I am indebted to Lawrence G. Roberts for providing me with the basic SOLVE program.

Seen from the outside, then, the linearization program and SOLVE make it possible for Sketchpad to find a new value for any variable to more closely meet the conditions indicated by constraints. Repeated application of these programs to variables, in sequence, implements the relaxation process. Application of these programs to selected variables to detect the number and degree of independence of constraints is used as an important part of the one pass constraint satisfaction method.

ONE PASS METHOD

Sketchpad can often find an order in which the variables of a drawing may be re-evaluated to completely satisfy all the conditions on them in just one pass. For the cases in which the one pass method works, it is far better than relaxation: it gives correct answers at once; relaxation

Meta Platforms, Inc.
Exhibit 1140
Page 118

may not give a correct solution in any finite time. Sketchpad can find an order in which to re-evaluate the variables of a drawing for most of the common geometric constructions. Ordering is also found easily for the mechanical linkages illustrated in the last chapter. Ordering cannot be found for the bridge truss problems illustrated in the last chapter.

The way in which the one pass method works is simple in principle and was easy to implement as soon as the nuances of the ring structure manipulations were understood. To visualize the one pass method, consider the variables of the drawing as places, and the constraints relating variables as passages through which one might pass from one variable to another. Variables are adjacent to each other in the maze formed by the constraints if there is a single constraint which constrains them both. Variables are totally unrelated if there is no path through the constraints by which one could pass from one to the other.

Suppose that some variable can be found which has so few constraints applying to it that it can be re-evaluated to completely satisfy all of them. Such a variable we shall call a "free" variable. As soon as a variable is recognized as free, the constraints which apply to it are removed from further consideration, because the free variable can be used to satisfy them. Removing these constraints, however, may make adjacent variables free. Recognition of these new variables as free removes further constraints from consideration and may make other adjacent variables free, and so on throughout the maze of constraints. The manner in which freedom spreads is much like the method used in Moore's algorithm[7] to find the shortest path through a maze. Having found that a collection of variables is free, Sketchpad will re-evaluate

Meta Platforms, Inc.
Exhibit 1140
Page 119

them in the reverse order, saving the first-found free variable until last. In re-evaluating any particular free variable Sketchpad uses only those constraints which were present when that variable was found to be free.

In the ring structure representation of the drawing all variables found to be free are placed in a special ring called the FREEDOMS ring. (Note that the FREE ring is used for empty spaces in storage and has nothing to do with freedom in the present sense.) Each variable placed on the FREEDOMS ring has associated with it, by extra ties, those constraints which it will be used to satisfy. In what order variables should appear in the FREEDOMS ring need only be computed when the constraint conditions change. For a given set of conditions the same ordering will serve for finding many satisfactory values. For example, as part of a linkage is moved with the light pen, the ordering first set up for the linkage serves until the conditions change.

Meta Platforms, Inc.
Exhibit 1140
Page 120

Chapter IX

EXAMPLES AND CONCLUSIONS

In the first chapter we saw, as an introduction to the system, some simple examples of Sketchpad drawings. In the body of this report we have seen many drawings, all of which, except the drawing of the light pen, Figure 4.2, were drawn with Sketchpad especially to be included here. In this chapter we shall consider a wider variety of examples in somewhat more detail. The examples in this chapter were all taken from the library tape and thus serve to illustrate not only how the Sketchpad system can be used, but also how it actually has been used so far.

We conclude from these examples that Sketchpad drawings can bring invaluable understanding to a user. For drawings where motion of the drawing, or analysis of a drawn problem is of value to the user, Sketchpad excells. For highly repetitive drawings or drawings where accuracy is required, Sketchpad is sufficiently faster than conventional techniques to be worthwhile. For drawings which merely communicate with shops, it is probably better to use conventional paper and pencil.

PATTERNS

The instance facility outlined in Chapter I enables one to draw any symbol and duplicate its appearance anywhere on an object drawing at the push of a button. The symbols drawn can include other symbols and so on to any desired depth. This makes it possible to generate huge numbers of identical shapes; if at each stage two of the previous symbols are combined to double the number of basic shapes present, in twenty steps one million objects are produced.

Meta Platforms, Inc.
Exhibit 1140
Page 121

The hexagonal pattern we saw in Figure 1.1, p. 10, is one example of a highly repetitive drawing. The hexagonal pattern was first drawn in response to a request for hexagonal "graph" paper. About 900 hexagons were plotted on a single 30 x 30 inch plotter page. It took about one half hour to generate the 900 hexagons, including the time taken to figure out how to do it. Plotting them takes about 25 minutes. The drafting department estimated it would take them two days to produce a similar pattern.

The instance facility also made it easy to produce long lengths of the zig-zag pattern shown in Figure 9.1. As the figure shows, a single "zig" was duplicated in multiples of five and three, etc. Five hundred zigs were generated in a single row. Four such rows were plotted one half inch apart to be used for producing a printed circuit delay line. Total time taken was about 45 minutes for constructing the figure and about 15 minutes to plot it.

In both the zig-zag pattern of Figure 9.1 and in the hexagonal pattern of Figure 1.1 the various subpictures were fastened together by attachment points. In the hexagonal pattern, each corner of the basic hexagon was attached to the corners of adjacent hexagons. The position of any hexagon was then completely determined by the position of any other. In the zig-zag pattern of Figure 9.1, however, only a single attachment was made between adjacent zig-zags. Additional constraints were applied to each instance to keep them erect and of the same size.

A somewhat less repetitive pattern to be used for encoding the time in a digital clock is shown in Figure 9.2. Each cross in the figure marks the position of a hole. The holes will be placed so that a binary coded decimal (BCD) number will indicate the time.

Meta Platforms, Inc.
Exhibit 1140
Page 122

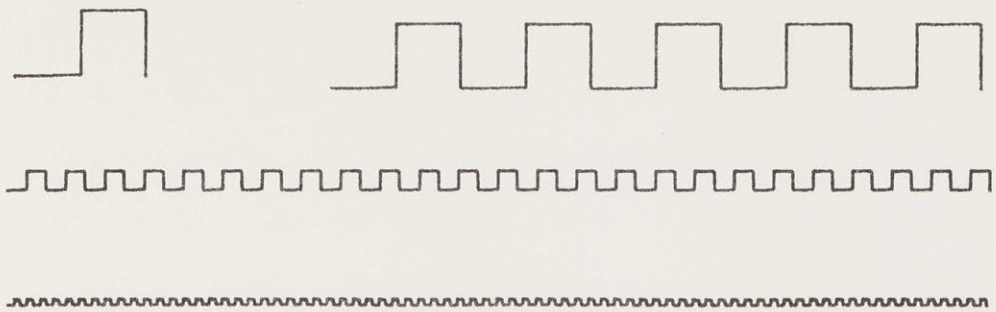


FIGURE 9.1.
ZIG-ZAG FOR DELAY LINE

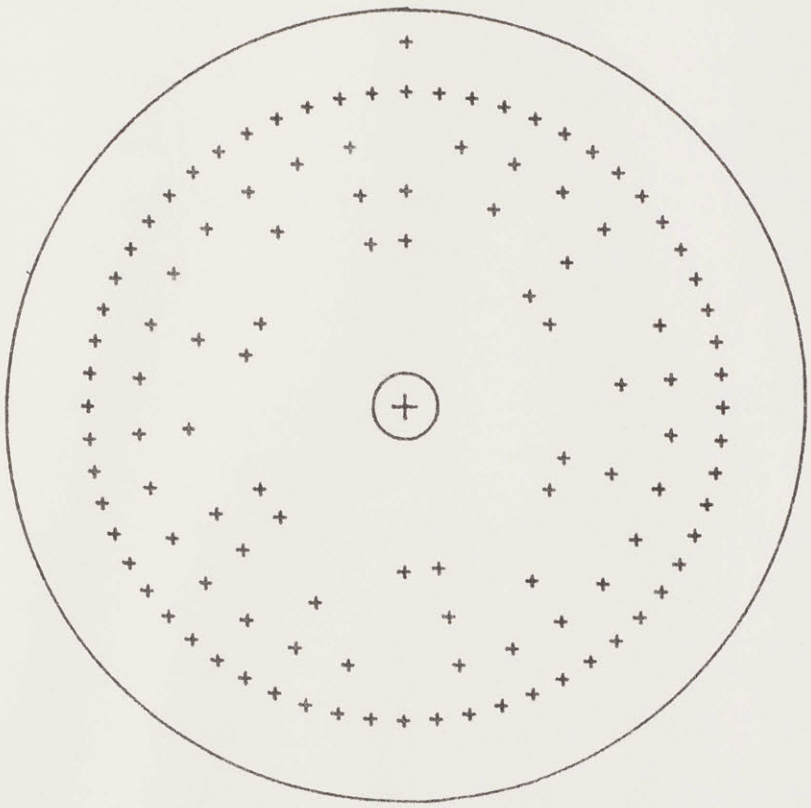


FIGURE 9.2.
BCD ENCODER FOR CLOCK

Meta Platforms, Inc.
Exhibit 1140
Page 123

Sketchpad was first used in the BCD clock project to produce 60 radial lines at equal 6° spacing. To do this a single 6° wedge was produced by first trisecting a right angle to obtain a 30° wedge and then cutting the 30° wedge into five parts. The relaxation procedure was used in each case to make three or five sketched-in chords equal in length. Making the 6° wedge took a brand new user less than one half hour including instruction time. The author has constructed other wedges as small as 1/128 of a circle in five minutes. All such wedges become a part of the library.

The 6° wedge has three attachment points. By attaching five of the wedges together, and then attaching three groups of five, a quadrant is constructed. Fitting together four quadrants gives a complete circle based entirely on the single 6° wedge. The advantage of constructing a full circle composed of 60 wedges is that any lines drawn in the original 6° wedge will appear 60 times around the circle with no further effort on the part of the user. Sixty radial lines were produced in this way.

Using the sixty radial lines plotted for him the BCD clock designer then marked with pencil approximately where the crosses should be placed to obtain BCD coding. Returning to Sketchpad we put a pattern of dots in the 6° wedge so that in the full circle, rings of dots appeared which could be aimed at with the light pen. It was then an easy matter to place a cross exactly on each of the desired dots. Total time for placing crosses was 20 minutes, most of which was spent trying to interpret the sketch.

Meta Platforms, Inc.
Exhibit 1140
Page 124

LINKAGES

By far the most interesting application of Sketchpad so far has been drawing and moving linkages. We saw in Chapter I the straight line linkage of Peaucellier, Figure 1.6, p.20. The ability to draw and then move linkages opens up a new field of graphical manipulation that has never before been available. It is remarkable how even a simple linkage can generate complex motions. For example, the linkage of Figure 9.3 has only three moving parts. In this linkage a central ⊥ link is suspended between two links of different lengths. As the shorter link rotates, the longer one oscillates as can be seen in the multiple exposure. The ⊥ link is not shown in Figure 9.3 so that the motion of four points on the upright part of the ⊥ may be seen. These are the four curves at the top of the figure.

To make the three bar linkage, an instance shaped like the ⊥ was drawn and given 6 attachers, two at its joints with the other links and four at the places whose paths were to be observed. Connecting the ⊥ shaped subpicture onto a linkage composed of three lines with fixed length created the picture shown. The driving link was rotated by turning a knob below the scope. Total time to construct the linkage was five minutes, but over an hour was spent playing with it.

Sketchpad can make linkages that one would hardly think of constructing out of actual links and pins. For example, a Sketchpad sliding joint is ideal, whereas to actually build a sliding joint is relatively difficult. Again, it is possible to make two widely separated links be of equal length by applying an appropriate constraint, but to build such a linkage would be impossible.

Meta Platforms, Inc.
Exhibit 1140
Page 125

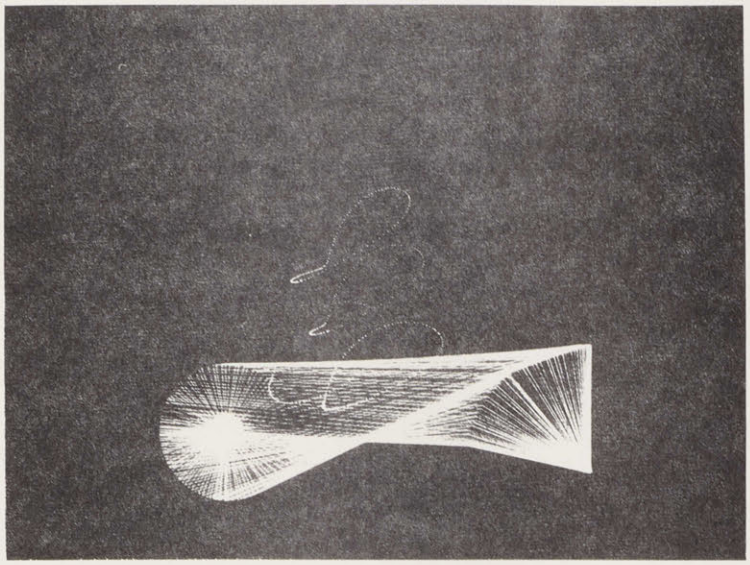


FIGURE 9.3. THREE BAR LINKAGE
The paths of four points on the central link are traced. This is a 15 second time exposure of a moving Sketchpad drawing.

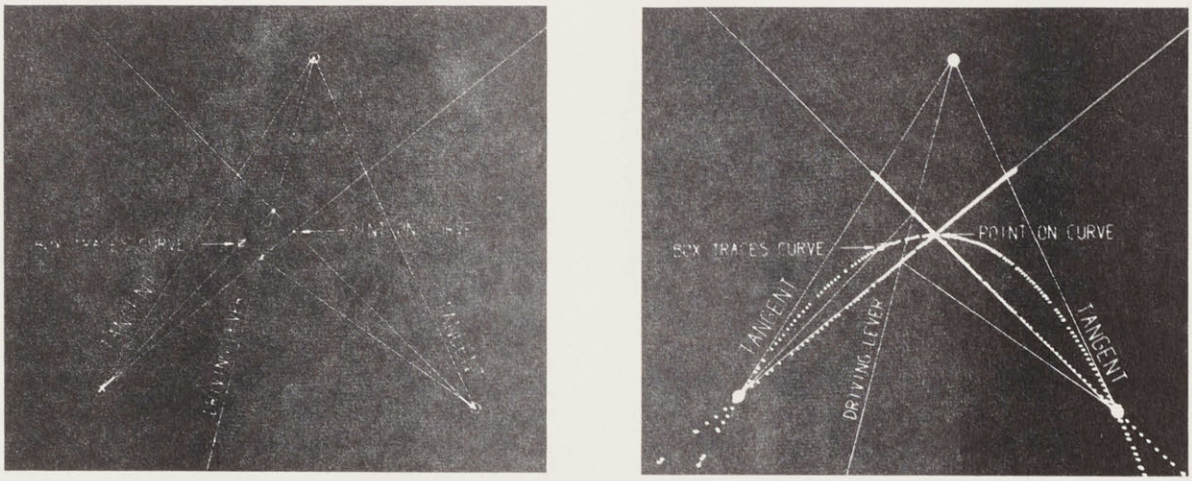


FIGURE 9.4. CONIC DRAWING LINKAGE
As the "driving lever" is moved, the point shown with a box around it traces a conic section. This conic can be seen in the time exposure.

Meta Platforms, Inc.
Exhibit 1140
Page 126

A linkage that would be difficult to build physically is shown in Figure 9.4. This linkage is based on the complete quadrilateral. The three circled points and the two lines which extend out of the top of the picture to the right and left are fixed. Two moving lines are drawn from the lower circled points to the intersections of the long fixed lines with the driving lever. The intersection of these two moving lines (one must be extended) has a box around it. It can be shown theoretically that this linkage produces a conic section which passes through the place labeled "point on curve" and is tangent to the two lines marked "tangent." Figure 9.4B shows a time exposure of the moving point in many positions. The straight dotted lines are the paths of other, less interesting points. **At first, this linkage was** drawn and **working in fifteen minutes.** Since then we have rebuilt it time and again until now we can produce it from scratch in about three minutes.

DIMENSIONING OF DRAWINGS

It is important that a Sketchpad drawing be made in the correct size for many applications. For example, the BCD clock pattern shown in Figure 9.2 was plotted exactly 12 inches in diameter for the actual application. In fact, the precision of the plotter is such that its plotted output can be used directly as a layout in many cases. But the size of a drawing as seen on the computer display is variable. To make it possible to have an absolute scale in drawings, a constraint is provided which forces the value displayed by a set of digits to indicate the distance between two points on the drawing. The distance is indicated in thousandths of an inch for "full size" plotted output.

Meta Platforms, Inc.
Exhibit 1140
Page 127

This distance indicating constraint is used to make the number in a dimension line. Many other constraints are used to make the arrowheads at the end of the line be "parallel" to the dimension line and to make enough space in the line for the dimension number. In some sense the dimension line is a complicated linkage; like a linkage it can be moved around while retaining its properties. For example, the arrowheads stay the same size even when the dimension line is made longer. A dimension line with small arrowheads is a part of the library. This line is suitable for dimensions of the order of a few inches. A three-four-five triangle dimensioned with this line is shown in Figure 9.5.

To produce the three-four-five triangle of Figure 9.5, three vertical and four horizontal line segments were made to be the same length. After erasing these lines, the three correctly positioned corners of the triangle were dimensioned. Putting in a dimension line is as easy as drawing any other line. One points to where one end is to be left, copies the definition of the dimension line by pressing the "copy" button, and then moves the light pen to where the other end of the dimension line is to be. The size of the three-four-five triangle was adjusted so that even dimensions appeared. At other sizes, of course, the ratio of the dimensions was correct but not so easy to recognize at a glance. Total time to produce dimensioned three-four-five triangle was three minutes, exclusive of time taken to produce the library version of the dimension line. The first dimension line took about fifteen minutes to construct, but that need never be repeated.

Meta Platforms, Inc.
Exhibit 1140
Page 128

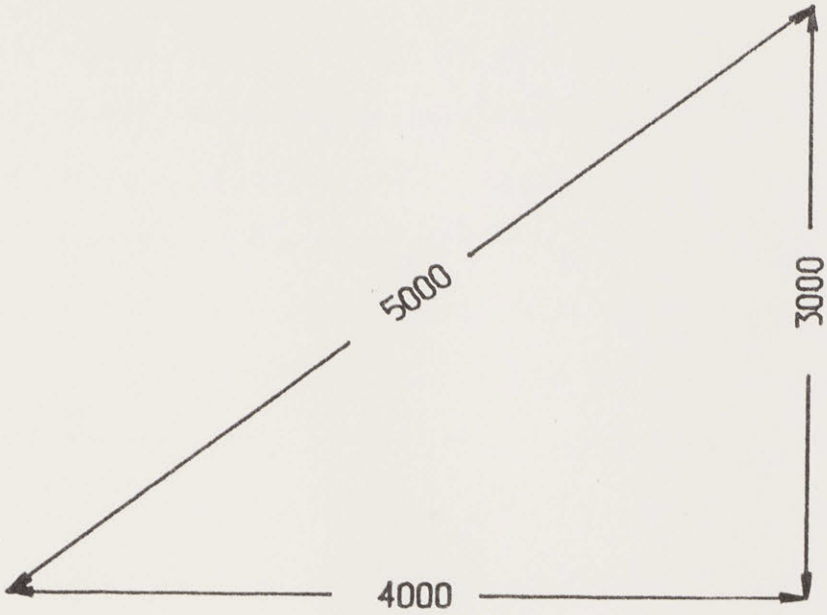


FIGURE 9.5. DIMENSION LINES

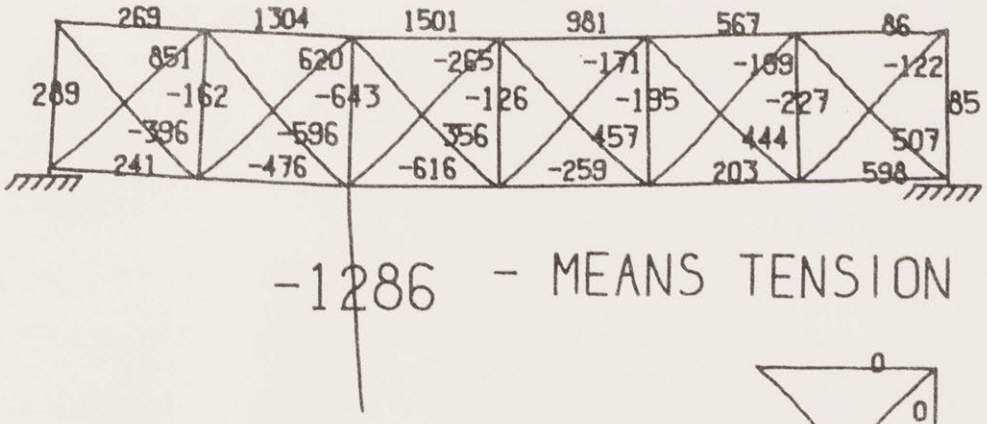


FIGURE 9.6.
TRUSS UNDER LOAD

Meta Platforms, Inc.
Exhibit 1140
Page 129

BRIDGES

One of the largest untapped fields for application of Sketchpad is as an input program for other computation programs. The ability to place lines and circles graphically, when coupled with the ability to get accurately computed results pictorially displayed, should bring about a revolution in computer application. With Sketchpad we have a powerful graphical input tool. It happened that the relaxation analysis built into Sketchpad is exactly the kind of analysis used for many engineering problems. By using Sketchpad's relaxation procedure we were able to demonstrate analysis of the force distribution in the members of a pin connected truss. We do not claim that the analysis represented in the next series of illustrations is accurate to the last significant digit. What we do claim is that a graphical input coupled to some kind of computation which is in turn coupled to graphical output is a truly powerful tool for education and design.

In Figure 9.6 is shown a truss bridge supported at each end and loaded in the center. To draw this figure, one bay of the truss (shown below the bridge) was first drawn with enough constraints to make it geometrically accurate. These constraints were then deleted and each member was made to behave like a bridge beam. A bridge beam is constrained to maintain constant length, but any change in length is indicated by an associated number. Under the assumption that each bridge beam has a cross-sectional area proportional to its length, the numbers represent the forces in the beams. The basic bridge beam definition (consisting of two constraints and a number) may be copied and applied to any desired line in a bridge picture. Each desired bridge member was

Meta Platforms, Inc.
Exhibit 1140
Page 130

changed from a line into a full bridge beam by pointing to it and pressing the "copy" button.

Using the bridge bay six times we construct the complete bridge. The loading line and the one missing end member are put in separately. The six-bay unloaded truss bridge is part of the library. It took less than ten minutes to draw completely. Applying a load where desired and attaching supports, one can observe the forces in the various members. It takes about 30 seconds for new force values to be computed. The bridge shown in Figure 9.6 has both outside lower corners fixed in position. Normally, of course, a bridge would be fixed only at one end and free to move sideways at the other end.

Having drawn a basic bridge shape, one can experiment with various loading conditions and supports to see what the effect of making minor modifications is. For example, an arch bridge is shown in Figure 9.7 supported both as a three hinged arch (two supports) and as a cantilever (four supports). For nearly identical loading conditions the distribution of forces is markedly different in these two cases.

ARTISTIC DRAWINGS

Sketchpad need not be applied only to engineering drawings. The ability to put motion into the drawings suggests that it would be exciting to try making cartoons. The capability of Sketchpad to store previously drawn information on magnetic tape means that every cartoon component ever drawn is available for future use. If the almost identical but slightly different frames that are required for making a motion picture cartoon could be produced semi-automatically, the entire Sketchpad system could justify itself economically in yet another way.

Meta Platforms, Inc.
Exhibit 1140
Page 131

FIGURE 9.7. CANTILEVER AND ARCH BRIDGES

Meta Platforms, Inc.
Exhibit 1140
Page 132

One way of cartooning is by substitution. For example, the girl "Nefertite" shown in Figure 9.8 can be made to wink by changing which of the three types of eyes is placed in position on her otherwise eyeless face. Doing this on the computer display has amused many visitors.

A second method of cartooning is by motion. A stick figure could be made to pedal a bicycle by appropriate application of constraints. Similarly, Nefertite's hair could be made to swing. This is the more usual form of cartooning seen in movies.

Aside from its economics as a teaching or amusement device, cartooning can bring the insights which are the prime value of Sketchpad drawings. The girl seen in Figure 9.9 was traced from a photograph into the Sketchpad system. The photograph was read into the computer by a facsimile machine used in another project[8] and shown in outline on the computer display. This outline was then traced with wax pencil on the display face. Later, with Sketchpad in the computer, the outline was made into a Sketchpad drawing by tracing the wax line with the light pen.

Once having the tracing on magnetic tape many things can be done with it. In particular, the eyes and mouth were erased to leave the featureless face which may also be seen in Figure 9.9. Returning to the tracing and erasing everything except the mouth and then everything except an eye we obtained features. In refitting the features to the blank face we discovered that, although the original girl was a sweet looking miss, an entirely different character appears if her mouth is made larger as in Figure 9.10. Using a computer to partially automate an artistic process has brought me, a non-artist, some understanding of

Meta Platforms, Inc.
Exhibit 1140
Page 133

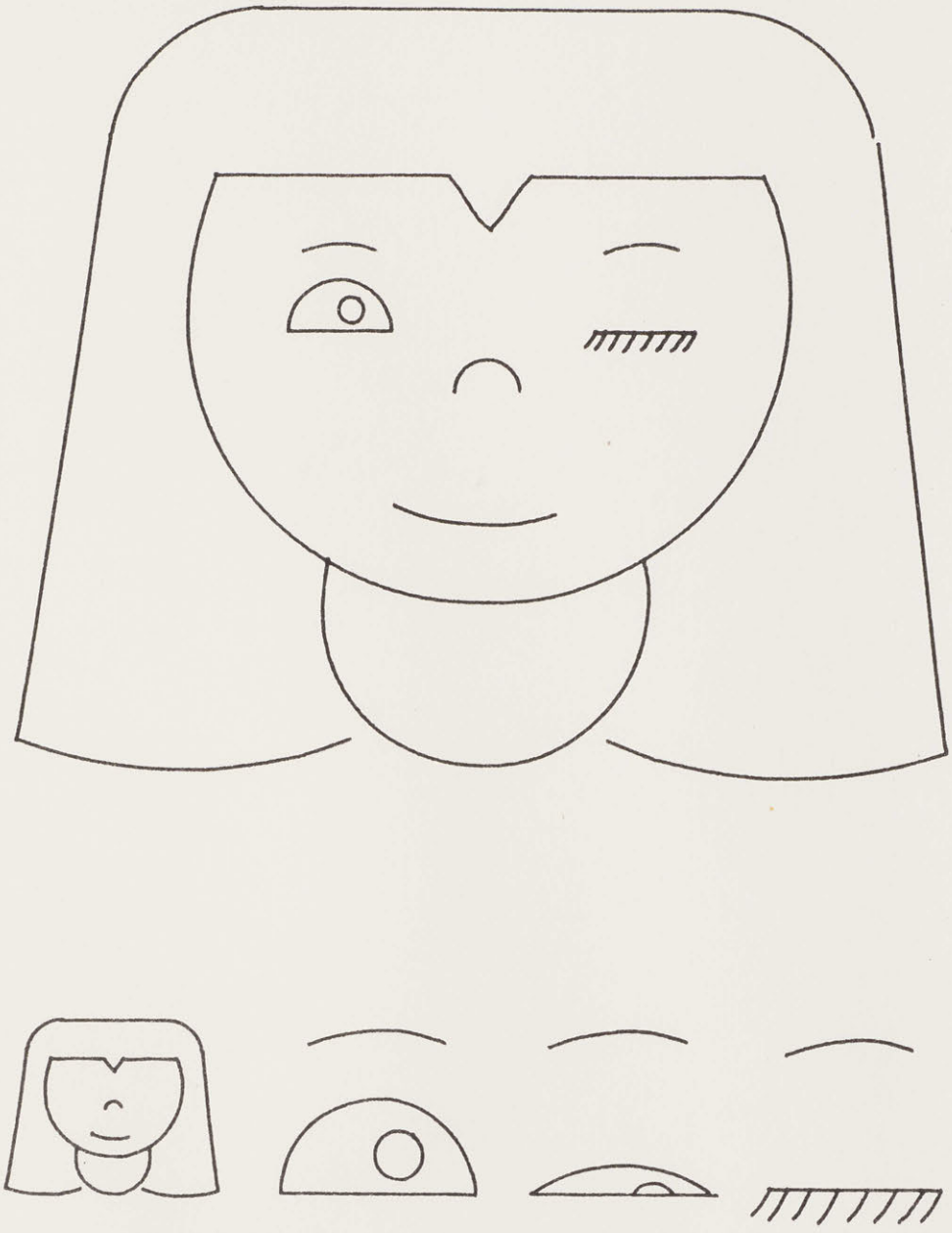


FIGURE 9.8.

WINKING GIRL AND COMPONENTS

Meta Platforms, Inc.
Exhibit 1140
Page 134



FIGURE 9.9.
GIRL TRACED FROM PHOTOGRAPH



FIGURE 9.10.
GIRL WITH FEATURES CHANGED

Meta Platforms, Inc.
Exhibit 1140
Page 136

the effect of certain features on the appearance of a face. It is the understanding that can be gained from computer drawings that is more valuable than mere production of a drawing for shop use.

ELECTRICAL CIRCUIT DIAGRAMS

Electrical engineers are, of course, interested in making circuit diagrams. It is not surprising that Sketchpad should be applied to this task. Unfortunately, electrical circuits require a great many symbols which have not yet been drawn properly with Sketchpad and are not therefore in the library. After some time is spent working on the basic electrical symbols it may be easier to draw circuits. So far, however, circuit drawing has been a big flop.

The circuits of Figure 9.11 are parts of an analog switching scheme. You can see in the figure that the more complicated circuits are made up of simpler symbols and circuits. It is very difficult, however, to plan far enough ahead to know what composits of circuit symbols will be useful as subpictures of the final circuit. The simple circuits shown in Figure 9.11 were compounded into a big circuit involving about 40 transistors. Including much trial and error, the time taken by a new user (for the big circuit not shown) was ten hours. At the end of that time the circuit was still not complete in every detail and he decided it would be better to draw it by hand after all.

CONCLUSIONS

The circuit experience points out the most important fact about computer drawings. It is only worthwhile to make drawings on the

Meta Platforms, Inc.
Exhibit 1140
Page 137

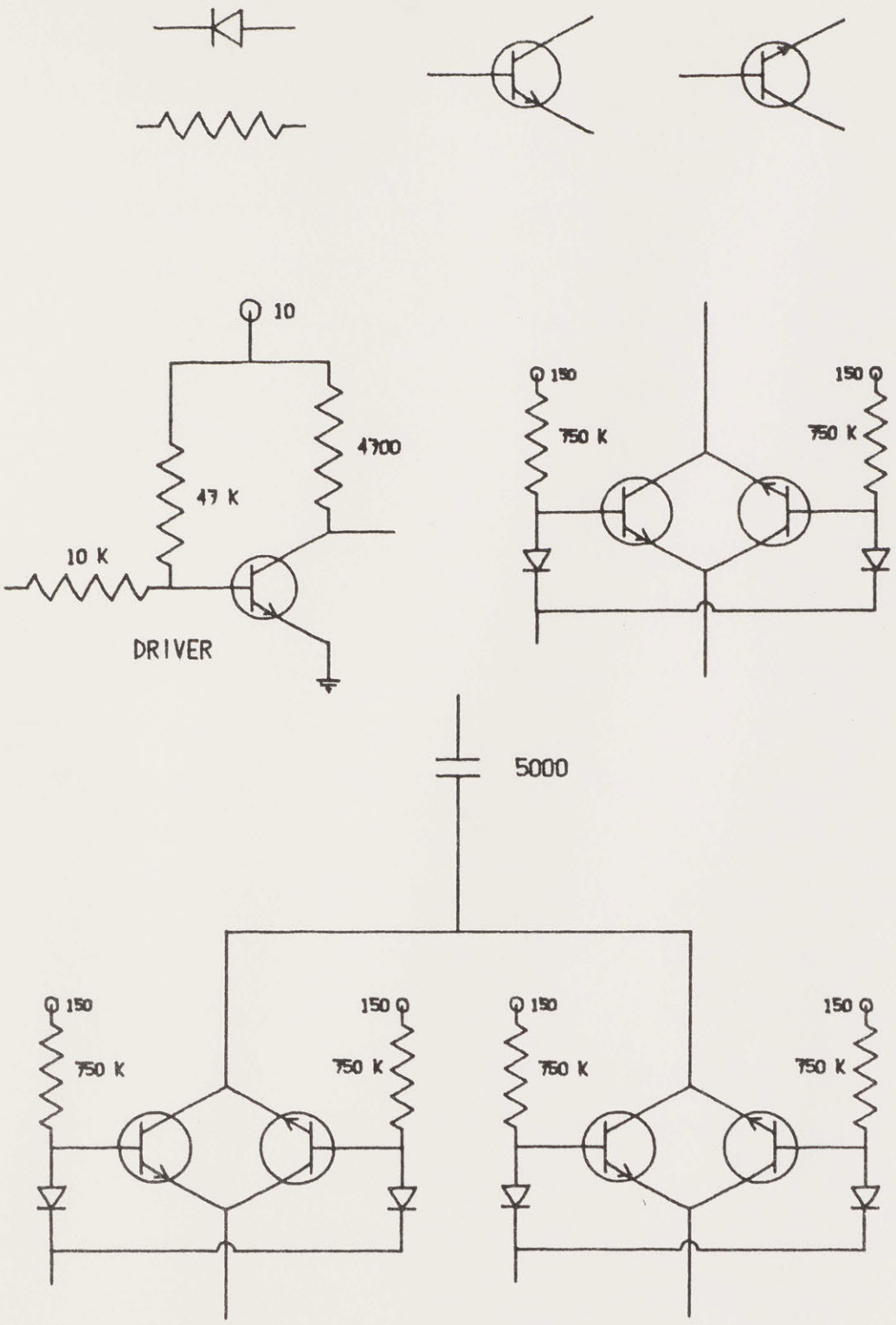


FIGURE 9.11.
CIRCUIT DIAGRAMS

Meta Platforms, Inc.
Exhibit 1140
Page 138

computer if you get something more out of the drawing than just a drawing. In the repetitive patterns we saw in the first examples, precision and ease of constructing great numbers of parts were valuable. In the linkage examples, we were able to gain an understanding of the behavior of a linkage as well as its appearance. In the bridge examples we got design answers which were worth far more than the computer time put into them. If we had had a circuit simulation program connected to Sketchpad so that we would have known whether the circuit we drew worked, it would have been worth our while to use the computer to draw it. We are as yet a long way from being able to produce routine drawings with the computer.

FUTURE WORK

The methods outlined in this report generalize nicely to three dimensional drawing. In fact, work has already been begun to make a complete "Sketchpad Three" which will let the user communicate solid objects to the computer. A forthcoming thesis by Timothy Johnson of the Mechanical Engineering Department will describe this work. When Johnson is finished it should be possible to aim at a particular place in the three dimensional drawing through two dimensional, perspective views presented on the display. Johnson is completely bypassing the problem of converting several two dimensional drawings into a three dimensional shape. Drawing will be directly in three dimensions from the start. No two dimensional representation will ever be stored.

Work is also proceeding on direct conversion of photographs into line drawings. Roberts reports a computer program[8] able to recognize

Meta Platforms, Inc.
Exhibit 1140
Page 139

simple objects in photographs well enough to produce three dimensional line drawings for them. Roberts is storing his drawings in the ring structure described in Chapter III so that his results will be compatible with the three dimensional version of Sketchpad.

Much room is left in Sketchpad itself for improvements. Some improvements are minor, such as including mirror image subpictures. Some improvements should be made to suit Sketchpad to particular uses that come up. For example, it is so interesting to study the path of particular points on a linkage that Sketchpad should be able to store and later display the path of chosen points.

More major improvements of the same order and power as the existing definition copying capability can be forseen. At present Sketchpad is able to add defined relationships to an existing object drawing. A method should be devised for defining and applying changes which involve removing some parts of the object drawing as well as adding new ones. Such a capability would permit one to define what rounding off a corner means. Then, by pointing at any corner and applying that definition, one could round off any corner. Sketchpad cannot now do this because rounding off a corner involves disconnecting the two lines which form the corner from the corner point and then putting a small circular arc between them.

HARDWARE

Sketchpad has pointed out some weaknesses in present computer hardware. A proposal for a line drawing display which would greatly surpass the capability of the spot display now in use is given in

Meta Platforms, Inc.
Exhibit 1140
Page 140

Appendix E. Such a display would not only provide flicker free display to the user, but also would relieve the computer of the burden it now carries in computing successive spots in the display.

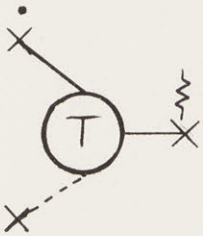
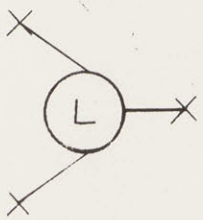
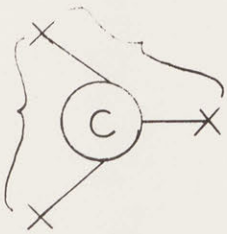
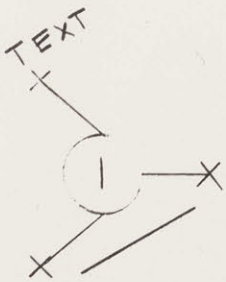
There are two conflicting demands made by Sketchpad on the light pen. On the one hand, the pen must have a fairly large field of view for ease of tracking. On the other hand, it should have a small field of view for aiming at objects. It should be possible to build a pen with two concentric fields of view which would report to the computer separately.

The arithmetic element of the computer is not used in doing the ring structure processing which forms a large part of Sketchpad. On the other hand, the index registers and their associated arithmetic are extensively used. This suggests that several users could share an arithmetic element if sufficiently powerful index arithmetic were made available to each of them.

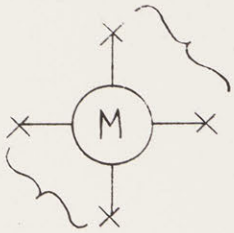
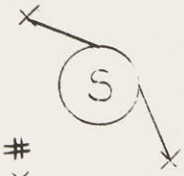
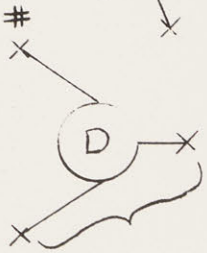
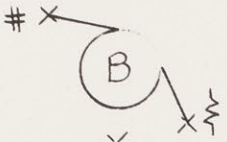
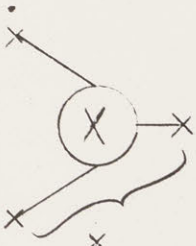
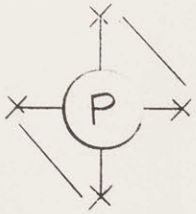
Meta Platforms, Inc.
Exhibit 1140
Page 141

Appendix A

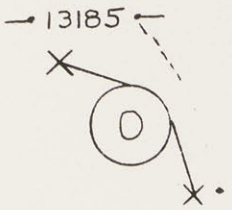
CONSTRAINT DESCRIPTIONS

	code	variable types	description
	43 T	point instance (point)	Point bears same relation to instance that (point) bears to its picture. GENERATED AUTOMATICALLY WITH INSTANCES
	33 L	p thing p thing p thing	Three things are collinear. Note: no distinction made about ordering of variables. GENERATED AUTOMATICALLY WHEN POINTS ARE CREATED ON LINES
	22 C	p thing p thing p thing	Distance from first to second is equal to distance from first to third. (First is circle center.) GENERATED AUTOMATICALLY WHEN POINTS ARE CREATED ON CIRCLES
	24 E	4 thing	Thing is erect or on its side. 
	27 H	p thing p thing	First thing is directly above or below, or directly beside second thing. (Horizontal or vertical line.) GENERATED AUTOMATICALLY FOR ANY LINE BY HORV BUTTON
	30 I	4 thing p thing p thing	4 thing is "parallel" to line between p things. Parallel to horizontal line means upright. (To set angle of text.)

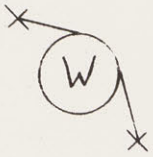
Meta Platforms, Inc.
Exhibit 1140
Page 142

	code	variable types	description
	34 M	p thing p thing p thing p thing	Distance from first thing to second is 1/3,1/2,1,2,3, times distance from third to fourth.
	42 S	4 thing 4 thing	First thing is 1/3,1/2,1,2,3 times size of second thing.
	23 D	scalar p thing p thing	Value of scalar equals distance between things in inches.
	21 B	scalar 4 thing	Value of scalar equals size of thing in inches.
	25 F	instance	Instance is full size, i.e. the same size as its master picture.
	47 X	p thing p thing p thing	First thing is at mid point of other two, e.g. dimension in dimension line is at center of line.
	06 6	4 thing	Thing is 1/32,1/16,1/8,1/4,1/2 or 1 inch in overall size.
	37 P	p thing p thing p thing p thing	Line from first to second would be parallel or perpendicular to line from third to fourth. (Lines need not be there.)

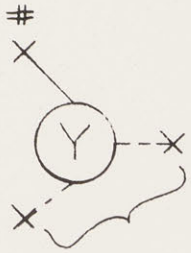
Meta Platforms, Inc.
Exhibit 1140
Page 143



code	variable types	description
36	4 thing	p thing will be next to 4 thing with enough space for 5 digit number, e.g. to create space in dimension line.
O	p thing	



46	p thing	Distance between things is maintained what it was last time meta of tog 22 was down. USES META OF TOG 22. e.g. for bridges and linkages.
W	p thing	



50	scalar	Value of scalar is equal to change in distance between p things since meta of tog 22 was down, sign considered. e.g. to display forces in beams. USES META OF TOG 22.
Y	(p thing)	
	(p thing)	

Meta Platforms, Inc.
Exhibit 1140
Page 144

Appendix B

PUSH BUTTON CONTROLS

BUTTON NAME	BIT NUMBER	FUNCTION
Draw	1.8	Create a new straight line segment or circle arc. End of line or arc left attached to light pen.
Circle center	1.7	Center of circle is left where pen is pointing. Next thing drawn will be circle arc.
Move	2.1	Object pointed at moves with light pen.
Delete	1.3	Object pointed at removed from drawing.
Instance	2.4	Instance of picture whose number is in toggle register 25 is created.
Copy 20 Copy 21 Copy 22 Copy 23	3.6 3.1 2.5 1.9	Four buttons. Copy definition picture indicated in toggle registers 20 to 23 respectively. These buttons can be set up to create equal length lines, dimension lines, etc. Any four functions can be available at once.
Stop	1.6	Leave moving object wherever it is. Merge moving object if aiming at object of like type. Same as termination flick of the pen.
Text	4.3	Create line of text consisting only of the letter X. Typing while a piece of text is moving adds to the text displayed.
Number	3.7	Create a new set of digits and a scalar which is its value. Digits left moving.
Hold	4.9	Following pen flick not to be taken as termination signal. Used to set pen aside for typing text.
Garbage	1.1	If pen is tracking, recenter picture so that place pen is pointing at will be in the center. If pen not tracking, compact ring structure by removing garbage.

Meta Platforms, Inc.
Exhibit 1140
Page 145

BUTTON NAME	BIT NUMBER	FUNCTION
Constraint	2.8	Create a new constraint of the type numbered in toggle register 25. Dummy variables are created. Constraint left moving.
Horv	2.9	Apply horizontal or vertical constraint to line aimed at. Choice is based on 45° cutoff.
Designate	2.2	Designate object. For copying a definition picture with three or more ties.
Tie	2.6	Object pointed at is an attacher of this picture.
Fix	3.3	This object must not move during constraint satisfaction. Moving an object with the light pen unfixes it.
Unfix	2.7	All fixed and designated objects unfixed and undesignated.
IBM	4.3	Read tape record. Number of record on tape given in toggle 26. Typewriter confirms successful reading or writing.
Library	3.9	Read a record from the TX-2 library tape. Address of record given in toggle register 27. Typewriter confirms.
Library write	Special start point	Write a record on library tape. Typewriter confirms.
Change instance	2.3	Moving instance or instance pointed at is changed to type indicated in Toggle register 25. Can change resistor into diode, etc.
Dismember	4.4	Instance pointed at is reduced one level, i.e., its internal structure on the next level becomes usable.
Order	4.6	Lines are put in better order for plotting.
Disorder	4.5	Lines are put in worst order for plotting.

Meta Platforms, Inc.
Exhibit 1140
Page 146

BUTTON NAME	BIT NUMBER	FUNCTION
Punch	4.7	Punch plotter tape for object picture.
Plot	4.8	Plot object picture.

The following dangerous functions only operate if "meta" button (4.10) is pressed as well.

Delete constraints	1.2	All constraints in object picture are deleted.
Delete points	1.4	All unattached points in object picture are deleted.
Delete picture	1.5	Entire object picture is deleted.
IBM	4.3	Write IBM tape record. Typewriter confirms.

Meta Platforms, Inc.
Exhibit 1140
Page 147

STRUCTURE OF STORAGE BLOCKS

(C) = Chicken (S) = Start of subroutine . = Spare register
(H) = Hen - = Ring part of component {} = Data part of block

TYPE OF BLOCK	STRUCTURE		REMARKS
Universe Variables Holders Constraints Topos Frees Deads Movings Curpics Freedoms Fixeds Desigs Mergers Works	TYPE - SPECB - {NAME .}	(C) (H)	All these short generic blocks use the same format. TYPE is a chicken (C) which connects the block to its next higher level in the generic structure, see Figure 3.8. SPECB is the hen (H) collecting the TYPE blocks in the next lower level. TYPE and SPECB serve this purpose in all blocks where they appear. NAME contains a four letter typewriter code name for each generic block. Counting lines, one finds that TYPE = 0, SPECB = 2, and NAME = 4.
Lines Circles Pictures	TYPE - SPECB -	(C) (H)	(Generic blocks for lines, circles and picture blocks.)
	NAME		
	DISPLAY	(S)	Display subroutine.
	HOWBIG	(S)	Fit scope around this thing.
	.		
	MOVIT	(S)	Apply transformation to this thing (Degenerate)
	SIZE		Length of line, circle and picture blocks.
	.		
	KIND		Put these in PPART or PICBLKS of a picture block.
Scalars Points Instances Texts Digits Dummies	TYPE - SPECB -	(C) (H)	(Generic blocks for various kinds of variables.)
	NAME		
	DISPLAY	(S)	
	HOWBIG	(S)	
	.		
	MOVIT	(S)	Apply transformation to this thing.
	SIZE		
	WHERE	(S)	Find position of thing on display.
	KIND		
	TUPLE		Number components in vector.
	VARLOC		Location of first vector component in block.

Meta Platforms, Inc.
Exhibit 1140
Page 148

Hov	TYPE	(C)	Generic blocks for various constraint
Porp	-		types.
etc.	SPECB	(H)	
etc.	-		
	NAME		
	DISPLAY	(S)	
	HOWBIG	(S)	Degenerate. (Does nothing.)
	.		
	MOVIT	(S)	Degenerate.
	SIZE		
	CONLET		Letter to appear in display.
	KIND		
	.		
	.		
	COMP	(S)	Error computing subroutine.
	NCON		Number degrees of freedom removed.
	CHVAR		Number of changeable variables.
Picture	TYPE	(C)	(Specific picture block.)
	-		
	PICBLKS	(H)	Abstractions in picture. KIND of generic
	-		block tells if a thing is an abstraction.
	PPART	(H)	Picture parts. Lines, Circles, Instances,
	-		Texts, and Digits in picture.
	PWHOS	(C)	Put into SPECB of Curpics ring if this
	-		is current picture.
	PPARTM	(H)	Moving parts of picture.
	-		
	PATAP	(H)	Attachers of this picture.
	-		
	PINS	(H)	Instances of this picture.
	-		
⎛	PSIZE	⎞	Overall size of this picture.
⎜	PNAME	⎟	36 bit "name" for this picture.
⎜	PSAVE	⎟	Space to save transformation when recursively
⎜	"	⎟	expanding instances.
⎜	"	⎟	
⎜	"	⎟	
⎜	"	⎟	
⎝	.	⎠	
Line	TYPE	(C)	(Specific line block.)
	-		
	ATATAP	(C)	Put into PATAP of picture if this line
	-		is an attacher.
	BWHOS	(C)	Which picture this thing belongs to.
	-		
	VORD	(C)	Put into SPECB of Movings if this line
	-		is moving.
	LSP	(C)	Start point of line. Goes into PLS ring
	-		of point.
	LEP	(C)	End point of line.
	-		

Meta Platforms, Inc.
Exhibit 1140
Page 149

Circle	TYPE	(C)	(Specific circle block.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	CSP	(C)	Start point of circle arc.
	-		
	CEP	(C)	End point of circle arc.
	-		
	CIRCEN	(C)	Center point of circle.
	-		
	CVAL		Angle of circle arc (to avoid ambiguity).
	"		Radius of Circle (to save recomputation).
Point	TYPE	(C)	(Specific point block.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	Put in SPECB of Freedoms during maze-solving constraint satisfaction.
	-		
	VFLW	(H)	Constraints which this variable will be used to satisfy.
	-		
	VCON	(H)	Constraints on this variable.
	-		
	PLS	(H)	Lines and Circles on this point
	-		
	IPCOTP	(H)	Instance-point constraints which use this point for reference only.
	-		
	PVAL		X coordinate of point.
	"		Y coordinate of point.
Instance	TYPE	(C)	Specific instance block. Size of instance is half size of enclosing box.
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	VFLW	(H)	
	-		
	VCON	(H)	
	-		
	IWHAT	(C)	What picture this is an instance of.
	-		
	IVAL		Size times cosine of rotation.
	"		Size times sine of rotation.
	"		X coordinate.
	"		Y coordinate.

Meta Platforms, Inc.
Exhibit 1140
Page 150

Text	TYPE	(C)	Particular lines of text. Size of text is half height of letters. Position of text is center of first letter in the line.
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	VFLW	(H)	
	-		
	VCON	(H)	
	-		
	TVAL		Size times cosine of rotation.
	"		Size times sine of rotation.
	"		X coordinate.
	"		Y coordinate.
	TXTS		Text to be shown, four letters per register, typewriter codes.
	"		
	"		
	"		
	"		
	"		
	"		
	"		
	"		
	"		

Dummy	TYPE	(C)	(Particular dummy variable.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	VFLW	(H)	
	-		
	VCON	(H)	
	-		
	TPVAL		X coordinate.
	"		Y coordinate.

Meta Platforms, Inc.
Exhibit 1140
Page 151

Digits	TYPE	(C)	(A particular set of digits. Size of digits is half height of figures.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	VFLW	(H)	
	-		
	VCON	(H)	
	-		
	NTOSHOW	(C)	Scalar whose value is to be shown.
	-		
	NVAL		Size times cosine of rotation.
	"		Size times sine of rotation.
	"		X position.
	"		Y position.
Scalar	TYPE		(A particular scalar block.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	VORD	(C)	
	-		
	VFLW	(H)	
	-		
	VCON	(H)	
	-		
	SSHOW	(H)	Digits showing this scalar's value.
	-		
	SVAL		Value of scalar.
	.		
Constraint	TYPE	(C)	(All constraint blocks have same format. If fewer than four variables, block will be shorter and VARIATION will be moved up.)
	-		
	ATATAP	(C)	
	-		
	BWHOS	(C)	
	-		
	CVTS,VORD	(C)	Variable used to satisfy this constraint in maze-solving method.
	-		
	VAR1		First constrained variable.
	-		
	VAR2		Second constrained variable.
	-		
	VAR3		
	-		
	VAR4		
	-		
	VARIATION		Code for variations within a constraint type. e.g., horizontal or vertical.
	-		

Meta Platforms, Inc.
Exhibit 1140
Page 152

Appendix D

RING OPERATION MACRO INSTRUCTIONS

The macro instructions listed in this appendix are used to implement the basic ring operations listed in Chapter III. Only the format is given here since to list the machine instructions generated would be of value only to persons familiar with the TX-2 instruction code. In each case the macro name is followed by dummy variables separated by non-alphabetic symbols. The dummy variables XR and XR2 refer to index registers which contain the address of the block which contains the ring element being worked on. The terms N of XR or N×XR mean the Nth element of the block pointed to by index register XR, for example, the LSP (line start point) register of the line block pointed to by index register α.

LTAKE≡N×XR

Take N of XR out of whatever ring it is in. The ring is reclosed. If N of XR is not in a ring, LTAKE does nothing. N of XR must not be a hen with chickens.

PUTL≡N×XR→M×XR2

PUTR≡N×XR→M×XR2

Put N of XR into the ring of which M of XR2 is a member. N of XR is placed to the left (PUTL) or right (PUTR) of M of XR2. M of XR2 may be either a hen or a chicken. N of XR must not already belong to a ring.

Meta Platforms, Inc.
Exhibit 1140
Page 153

MOVEL≡N×XR→M×XR2

MOVER≡N×XR→M×XR2

Combination of LTAKE and PUTL (PUTR). Assumes that both N of XR and M of XR2 are in the same ring. Intended for reordering a ring.

CHGRL≡N×XR→M×XR2

CHGRR≡N×XR→M×XR2

Combination of LTAKE and PUTL (PUTR). N of XR and M of XR2 may be in different rings.

LGORR≡N×XR=XR2→SUBR→LEXIT

LGORL≡N×XR=XR2→SUBR→LEXIT

Go around the ring of which N of XR is the hen. Exit to subroutine SUBR once for each ring member. The address of the top of the block to which each ring member belongs is put in XR before starting the subroutine. XR2 is used as a working index register. The subroutine may destroy the contents of both XR and XR2. The subroutine may delete individual members of the ring provided recursive deletion does not delete additional ring members. The subroutine must not generate new ring members. Jump to LEXIT when finished with the ring. Go around the ring to the right (LGORR) or left (LGORL).

LGORRI≡N×XR=XR2→SUBR→LEXIT

LGORLI≡N×XR=XR2→SUBR→LEXIT

Same as LGORR except that the subroutine may generate new members in the ring. The subroutine must not delete the current member of the ring. New members will be visited if they are put in the ring later in sequence.

COMBHR≡N×XR→M×XR2

COMBHL≡N×XR→M×XR2

The members of the ring whose hen is at N of XR are placed in the ring of which M of XR2 is a member. N of XR must not be empty. The new members are placed to the right (COMBHR) or left (COMBHL) of M of XR2. M of XR2 may be either a hen or a chicken. N of XR is left empty.

Meta Platforms, Inc.
Exhibit 1140
Page 154

Appendix E

PROPOSAL FOR AN INCREMENTAL CURVE DRAWING DISPLAY

In the course of the work with Sketchpad it has become all too clear that the spot-by-spot display now in use too slow for comfortable observation of reasonable size drawings. Moreover, having the central machine compute and store all the spots for the display is a waste of general purpose capacity that might better be applied to other jobs. As a solution to these difficulties I propose that a special purpose incremental computer be used to generate the successive spots of the display at high speed. The central machine would provide only a minimum of information about each curve to be drawn; e.g., end points of lines; start, center and arc length of circle arcs.

The technology of incremental computers is well developed, but so far as I know, no one has yet applied them directly to the problem of computer display systems. Basically the incremental computer works by adding one register to another successively and detecting any overflows or underflows which may be generated. Certain registers are incremented conditionally on the result of overflow or underflow generation.

In the system of Figure E.1, the x and y increment registers are added to the x and y remainder registers and overflows or underflows (dotted lines) are used to increment the beam position of the display. A counter (not shown) is provided to limit the length of the straight line generated. The unit would request more information from the computer after the appropriate number of additions. For drawing straight lines on a 1024 x 1024 raster display, the increment registers should

Meta Platforms, Inc.
Exhibit 1140
Page 155

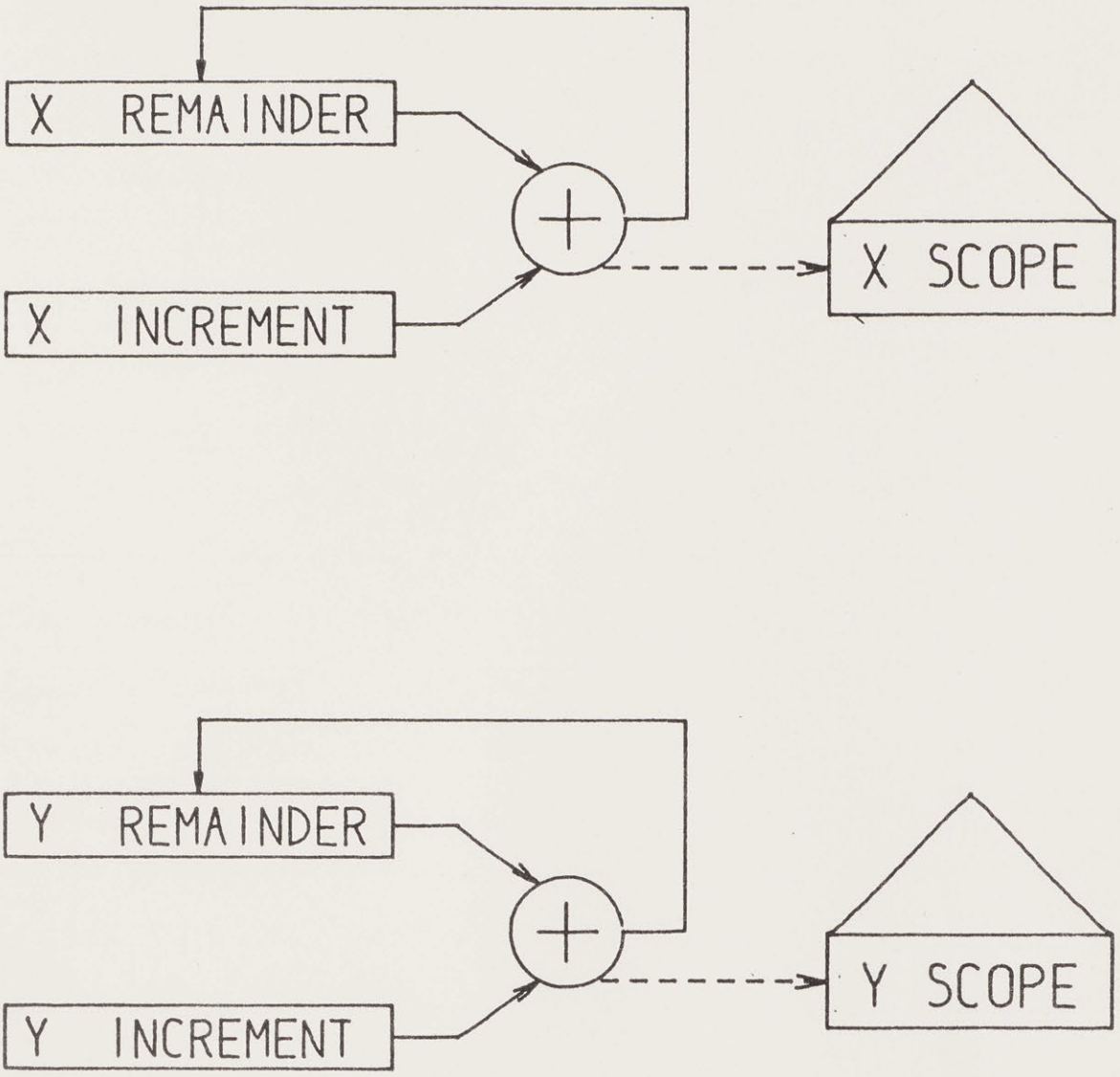


FIGURE E.1.
DDA FOR DRAWING LINES

Meta Platforms, Inc.
Exhibit 1140
Page 156

contain 10 bits plus sign, 11 bits in all each; the remainder registers should contain 10 bits with no sign; and the counter should contain 10 bits.

To understand how the system of Figure E.1 operates consider that its x increment register contains the largest possible positive number and that its y increment contains one half that value. The x addition would result in overflow nearly every iteration, whereas the y addition would result in overflow only on alternate additions, and so a line would be drawn up and to the right with a slope of 1/2.

The usual practice in incremental computers is to be able to step the increment registers by a single unit up or down according as overflow or underflow is produced in another addition. In the system of Figure E.2, the (+?) is an adder-subtractor which can increase or decrease the increment register by the amount stored in the curvature register. The (+?) adds or subtracts if overflow or underflow is generated in the other addition. Overflow or underflow is signalled to the (+?) adder along the dotted paths in Figure E.2.

Use of the conditional adder permits a curvature to be specified so that curves can be drawn. The system of Figure E.2 will draw straight lines if the numbers in the curvature registers are zero, circles if the numbers are equal and opposite in sign, ellipses if the numbers are unequal and unlike in sign, and hyperbolas if the numbers are like in sign. The ellipses and hyperbolas are generated, however, with axes parallel to the coordinate axes of the display.

Theory and simulation show that just as in the incremental equation used for generating circles (see Chapter V), the latest value of increment must be used if the curve is to close. Therefore, the additions

Meta Platforms, Inc.
Exhibit 1140
Page 157

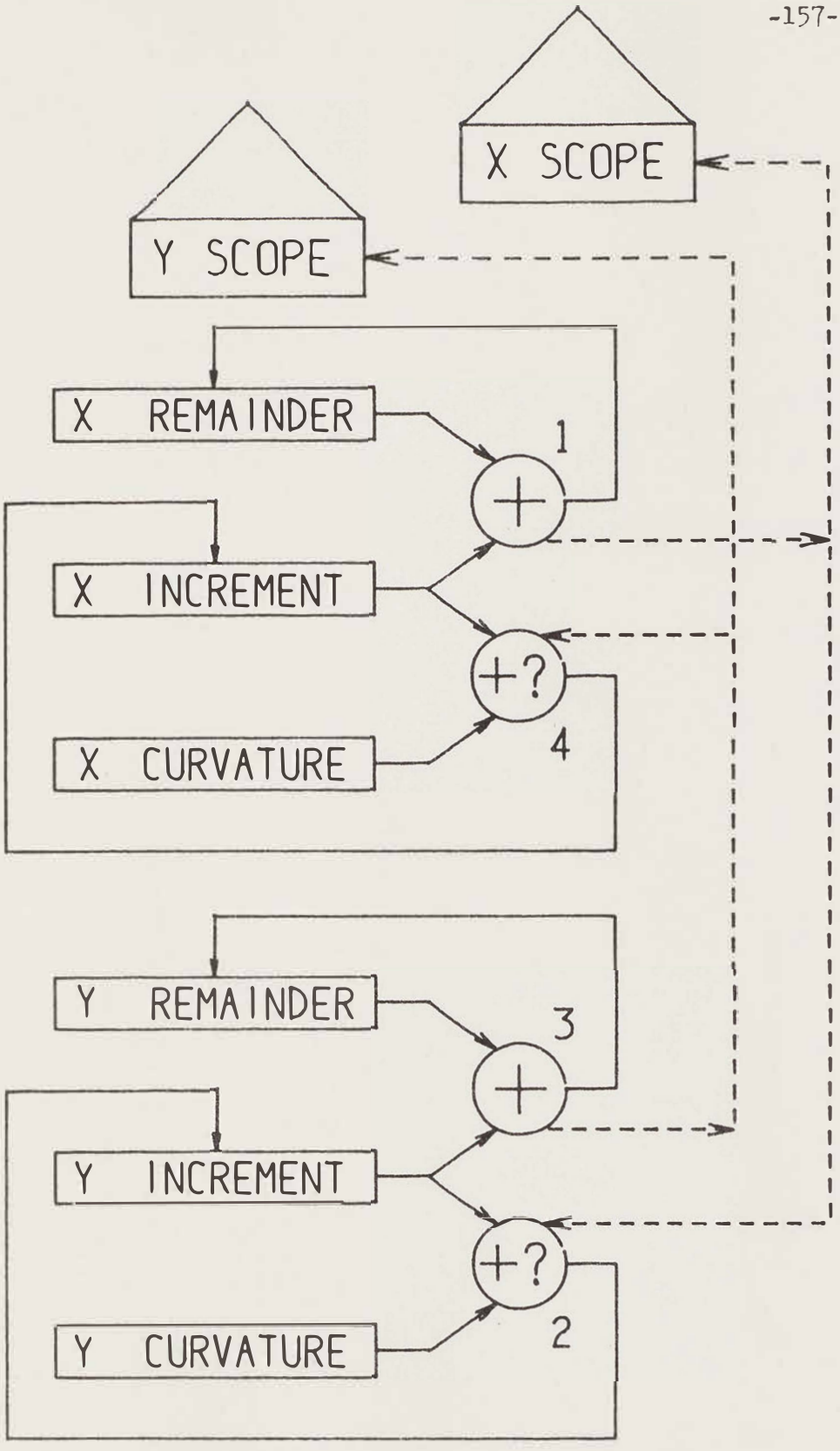


FIGURE E.2.
DDA FOR UPRIGHT CONICS

Meta Platforms, Inc.
Exhibit 1140
Page 158

cannot all occur at once; the order shown in Figure E.2 by the numbers 1-4 next to the adders makes the circles and ellipses close. In a serial device it is possible to do the four additions in just two add times by having only a one bit time delay between the two additions for each coordinate, i.e., (+?) just before (+).

Circles can be drawn with radii from about one scope unit to a straight line according to the numbers put in the curvature registers. Simulation shows that if the increment and curvature registers contain 17 bits plus sign, 18 bits each in all, and the remainder contains 17 bits without sign, the largest radius circle that can be drawn is just noticeably different from a straight line after having passed fully across a 1024 x 1024 raster display. The simulation program for this test is less than 100 instructions long and requires, of course, no multiply or divide. Simulation of larger incremental computers on small general purpose digital computers should be a powerful way to get complex numerical answers quickly and easily.

If the system of Figure E.2 is duplicated twice as shown in Figure E.3, a general Conic Section drawing capability is obtained. I am indebted to Larry M. Delfs for pointing out that the display incrementing outputs of the two systems should be added together. The full system of Figure E.3 can draw not only arbitrary conic sections but a host of interesting cycloidal curves. For drawing the simple straight lines and circles, the two halves of the system would be loaded with identical numbers to gain a two-fold speed advantage.

A trial design using 20 megacycle serial logic and 36 bit delay lines available commercially showed that the full system would be able

Meta Platforms, Inc.
Exhibit 1140
Page 159

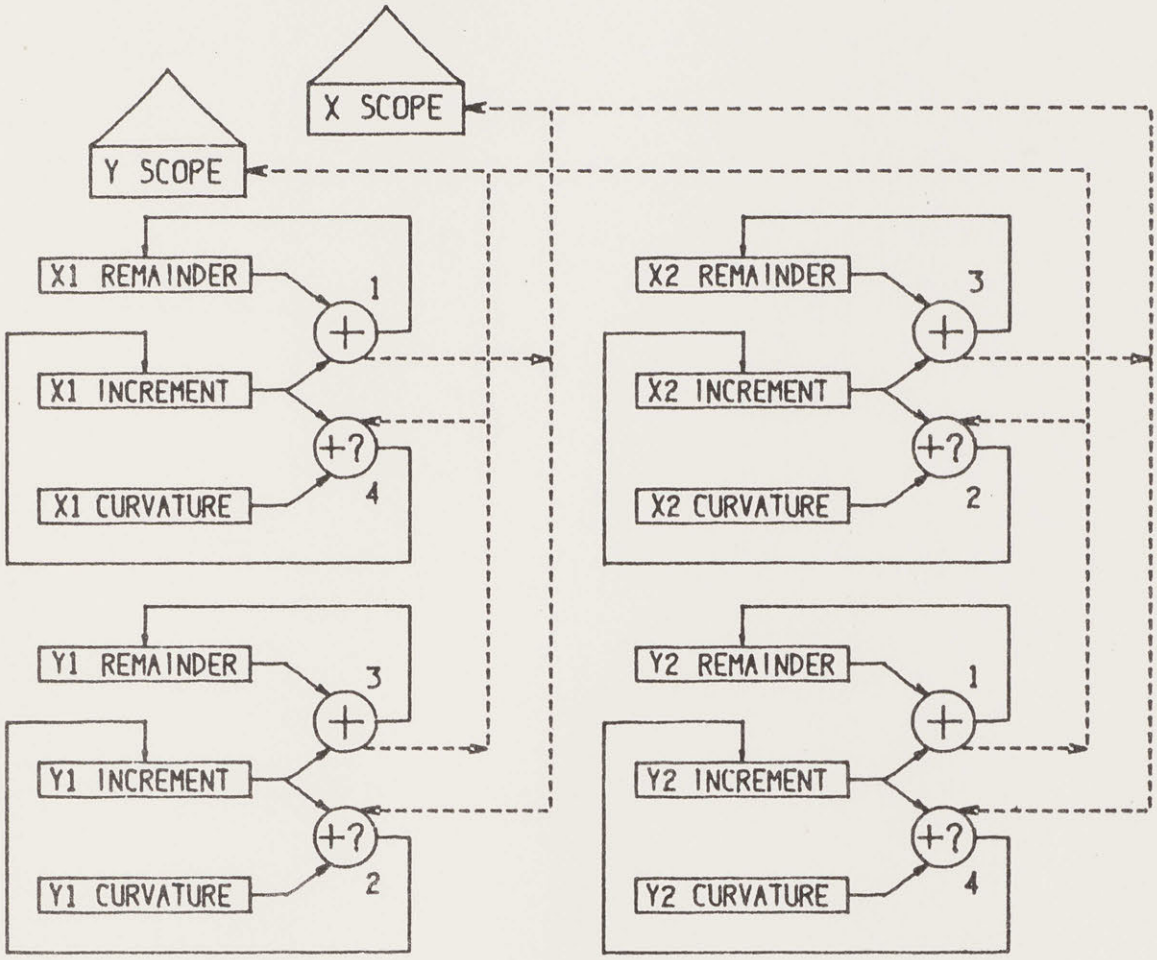


FIGURE E.3.
DDA FOR THE GENERAL CONIC

Meta Platforms, Inc.
Exhibit 1140
Page 160

to generate new display points at 0.9 microseconds each for lines and circles and slightly slower (but not half speed) for complicated conics. This corresponds to a writing rate of about 10,000 inches per second. Some saving in cost could be expected if longer delay lines were used and a correspondingly slower operation speed were tolerated. It appears possible to get similar performance from a parallel scheme.

Meta Platforms, Inc.
Exhibit 1140
Page 161

Appendix F

MATHEMATICS OF LEAST MEAN SQUARE FIT

The result quoted in this appendix is well known and is repeated here only for reference.

Suppose we have P equations in N unknowns:

$$\sum_{j=1}^{N} a_{ij}\, x_j = c_i \quad 1 \leq i \leq P; \qquad \text{or} \qquad AX = C\,. \tag{F-1}$$

If P is larger than N there would in general be no exact solution. We wish to find the values for the unknowns which minimize the sum of the squared errors of the equations. The error in the i^{th} equality is given by:

$$E_i = \sum_{j=1}^{N} (a_{ij}\, x_j - c_i), \tag{F-2}$$

and the total squared error,

$$E_t = \sum_{i=1}^{P} \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right]^2. \tag{F-3}$$

We wish to minimize E_t, and so we take partials with respect to each x_j and set all these equal to zero. For a particular x_j called x_k,

$$\frac{\partial E_t}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{P} \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right]^2. \tag{F-4}$$

Since the partial of a sum is equal to the sum of the partials,

Meta Platforms, Inc.
Exhibit 1140
Page 162

$$\frac{\partial E_t}{\partial x_k} = \sum_{i=1}^{P} \frac{\partial}{\partial x_k} \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right]^2, \qquad \text{(F-5)}$$

or since

$$\frac{\partial}{\partial x} (Q)^2 = 2Q \frac{\partial}{\partial x} Q,$$

$$\frac{\partial E^2}{\partial x_k} = \sum_{i=1}^{P} 2 \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right] \frac{\partial}{\partial x_k} \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right]. \qquad \text{(F-6)}$$

Now the last part of (F-6) is a sum of terms like $a_{12}x_2$ ... only one of which involves x_k at all, namely $a_{ik}x_k$. Therefore,

$$\frac{\partial E^2}{\partial x_k} = \sum_{i=1}^{P} 2 \left[\sum_{j=1}^{N} (a_{ij}\, x_j) - c_i \right] (a_{ik}), \qquad \text{(F-7)}$$

which, when set equal to zero gives:

$$0 = \sum_{i=1}^{P} \left[\sum_{j=1}^{N} (a_{ik}\, a_{ij}\, x_j) - a_{ik}\, c_i \right], \qquad \text{(F-8)}$$

or

$$\sum_{i=1}^{P} \sum_{j=1}^{N} a_{ik}\, a_{ij}\, x_j = \sum_{i=1}^{P} a_{ik}\, c_i. \qquad \text{(F-9)}$$

Changing the order os summation,

$$\sum_{j=1}^{N} \left(\sum_{i=1}^{P} a_{ik}\, a_{ij} \right) x_j = \left(\sum_{i=1}^{P} a_{ik}\, c_i \right), \qquad \text{(F-10)}$$

which in matrix notation becomes:

$$A^T A X = A^T C. \qquad \text{(F-11)}$$

Meta Platforms, Inc.
Exhibit 1140
Page 163

A^TA is a square matrix of order N. Thus a system of any number of linear equations can be reduced to a simpler system whose solution is the value of the variables for least square fit to the original set of equations.

If the original equations are equations in two unknowns, a plot of (F-2) with error squared in the upward direction is a parabolic valley. Since any vertical section of a parabolic valley will be a parabola, and the sum of any two parabolas in likewise a parabola, a plot of (F-3) can at most be an eliptic paraboloid. The Equations (F-10) and (F-11) resulting from the method described here represent the locus of locations where contour lines of the eliptic paraboloid are parallel to the axes. The intersection of these loci, the solution of (F-11), is the lowest point in the eliptic paraboloid, the least mean squares fit to (F-1).

Meta Platforms, Inc.
Exhibit 1140
Page 164

Appendix G

A BRIEF DESCRIPTION OF TX-2*

At first glance, TX-2 is an ordinary single-address, binary digital computer with an unusually large memory. It is an experimental machine—many of its in-out devices are not commercially available. On closer inspection, one finds it has some important innovations—at least they were innovations at the time TX-2 was built (1956).

The distinctive features of TX-2 are:

1. Simultaneous use of in-out machines through interleaved programs.
2. Flexible, "configured" data processing.

Some other virtues include:

1. Automatic memory and arithmetic overlap.
2. A "bit" sensing instruction (i.e., the operand is one bit!).
3. Addressable arithmetic element registers.
4. Especially flexible in-out.
5. 64 index registers.
6. Indirect—i.e. deferred addressing.
7. Magnetic Tape Auxiliary Storage

IN-OUT

The phrase "simultaneous use of in-out machines" should be taken quite literally. It does not mean simultaneous control. Each unit has

---

* By Alexander Vanderburgh

Meta Platforms, Inc.
Exhibit 1140
Page 165

its own buffer register and only one of these can be processed by TX-2 at any given instant. It is the relative speed that is important. For example, the in-out instruction that "fills" the display scope buffer takes no more than 10 microseconds, but the display itself takes from 20 to 100 microseconds, i.e., up to ten times as long. While the display is busy, the computer can compute the next datum of course, but it can also initiate other in-out transfers. In practice, since most in-out units are much slower than their associated programs, the computer spends a significant percentage of the time just waiting (in "Limbo"), even when several devices are in use. Interleaved initiation of in-out data transfers is partly automatic and partly program controlled. Each in-out routine is independently coded and is operated by TX-2 according to its "priority." Each unit has a "Flag Flip-Flop" to indicate to control that it is ready for further attention. When a unit is ready for further attention its routine will be operated unless another unit of higher priority also needs attention. An index register is reserved for each in-out unit and is used as a "place-keeper" when its routine is not being operated. The sharing among in-out routines of storage, index memory, and the arithmetic element is the programmer's responsibility.

"CONFIGURED" DATA PROCESSING

The "normal" word length for TX-2 is 36 bits. For many applications 18 or 9 bits would suffice, and in some cases each piece of data requires the same processing. Configuration control permits "fracture" of the normal word into two 18 bit pieces, four 9 bit pieces, or one 27

Meta Platforms, Inc.
Exhibit 1140
Page 166

bit and one 9 bit. These "subwords" are completely independent—for example, there are separate overflow indicators. In addition to "fracture" there is "activity" and "quarter permutation". Any quarter word can be made "inactive" i.e., inoperative. The 9 bit quarters of a datum from memory may be rearranged (permuted) before use. There are 8 standard permutations—for example, the right half of memory can be used with the left half of the arithmetic element. Nine bits are required for complete configuration specification. Since only 5 bits are available for bit thin film memory is addressed by each instruction word, a special 32 word, 9 bit thin film memory is addressed by each instruction that processes data directly. A complete change to any of 32 configurations is therefore possible from instruction to instruction.

THE SMALLER VIRTUES

Overlap: TX-2 has two core memories—"S" memory, a vacuum tube driven 65,536 word core memory, and "T" memory, a transistor driven 4096 word core memory about 20% faster. Instruction readout can be done concurrently with the previous data readout if program and data are in separate memories.

The use of the arithmetic element is also overlapped. Instructions that follow a multiply or divide operation will be done during the arithmetic time if they make no reference to the arithmetic element. The overlap is entirely automatic and may be ignored if the programmer chooses. A careful programmer can gain speed by doing indexing after multiply or divide and by putting program and data in separate memories.

Meta Platforms, Inc.
Exhibit 1140
Page 167

Bit Sensing Instruction: One instruction--SKM--uses a single bit of any memory word as its operand. Control bits provide 32 variations of skipping setting, clearing, and/or complementing the selected bit. This instruction can also cycle the whole word right one place if desired.

Addressable Arithmetic Element: Seventeen bits of the TX-2 instruction word are reserved for addressing an operand. This would allow a 131,072 word memory. TX-2 has only 69,632 registers of core storage. The toggle switch and plugboard memories, the real time clock register, the knob register (shaft encoder), and the arithmetic element registers use 55 of the remaining addressing capability. The arithmetic element registers are therefore part of the memory system and can be addressed, e.g., one can add the accumulator to itself.

Flexible In-Out: The TX-2 user must program each and every datum transfer. The lack of complex automatic in-out controls may seem to be a burden, but the simplicity of the system gives the programmer much more precise and variable control than automatic systems provide. For example, coordination of separate in-out units such as display and light pen is possible. Moreover, it is relatively easy to attach new in-out machines as they become available.

Index Memory and Indirect Addressing: Of the 64 index registers, one must devote a few to each in-out unit's program. With all 21 in-out devices concurrently in use, each program would have two index registers for normal programming use. In practice, one seldom uses more than half a dozen in-out units, and each routine would then have 9—clearly a luxury. Indirect addressing provides a means for indexing normally nonindexable instructions, or for double indexing normal instructions.

Meta Platforms, Inc.
Exhibit 1140
Page 168

Magnetic Tape Auxiliary Storage: Each TX-2 magnetic tape unit stores about 70 million bits, 34 times the capacity of the core memory system. Like a magnetic drum the tape is addressable. It can be read in either direction at any speed from 60 to 600 ips, and can be searched at a maximum of 1200 ips. It is used at present primarily for program storage. "Turn around time"—i.e. the time required to save one program and read-in a different one is seldom more than 2 minutes and often less than 30 seconds. (The read-in time, once the desired section of the tape is found, is about 12 seconds for 69,632 words.) A standard IBM 729 tape unit is also available.

SUMMARY OF VITAL STATISTICS--TX-2—DECEMBER 1962

Word Length:	36 bits, plus parity bit, plus debugging tag bit		
Memory:	256 x 256 core	65,536 words	6.0 μsec cycle time
	64 x 64 core	4,096 words	4.4 μsec cycle time
	Toggle switch	16 words	
	Plugboard	32 words	
Auxiliary Memory:	Magnetic Tape 2+ million words, 70+ million bits per unit (2 units in use, total of 10 planned)		
Tape Speeds:	selectable 60-300 inches/sec, search at 1000 inches/sec (i.e. about 1600 to 8000 36 bit words/sec)		

IN-OUT EQUIPMENT

Input:

Paper Tape Reader: 400-2000 6 bit lines/sec

2 keyboards—Lincoln writer 6 bit codes

Meta Platforms, Inc.
Exhibit 1140
Page 169

Input:

Random number generator--average 57.6 μsec per 9 bit number

IBM Magnetic Tape (Model 729 M6)

Miscellaneous pulse inputs—9 channels—push buttons or other source

Analog input—Epsco Datrac—nominal 11 bit sample
—27 kilocycle max. rate

2 light pens—work with either scope or both on one

Special memory registers:

Real time clock

4 shaft encoder knobs, 9 bits each

592 toggle switches (16 registers)

37 push buttons—any or all can be pushed at once

Output:

Paper tape punch—300 6 bit lines/sec

2 typewriters--10 characters per second

IBM Magnetic Tape (729 M6)

Miscellaneous pulse/light/relay contacts—9 channels (low rates)

Xerox printer—1300 char. sec

2 display scopes—7 x 7 inch usable area, 1024 x 1024 raster

Large board pen and ink plotter--29" x 29" plotting area
15 in/sec slew speed. Off line paper tape control as well as direct computer control.

Meta Platforms, Inc.
Exhibit 1140
Page 170

4-thing — A four component variable: text, digits, or instance.

Aim — To place the light pen so that light from the picture part aimed at falls on the photocell and so that the center of the light pen field of view is sufficiently close to the picture part.

Atomic — Axiomatic, fundamental, built in. The atomic constraints are listed in Appendix A. The atomic operations are each controlled by a push button listed in Appendix B.

Attacher — For instances, a particular point designated in the master for which in the instance the light pen will have a particular affinity. Also the related point created in the picture containing the instance when the instance was created.

For copying, any drawing part designated in the definition picture. Attachers may be recursively merged with object picture parts when the definition is copied.

Balance — The property of equal weight among constraints obtained by making error in a constraint equal to displacement.

Block — A set of consecutive registers used to represent a picture part. An n-component element.

Chicken — A subordinate ring member, composed of two registers one of which references the block containing the hen for this ring, the other references the next and previous chickens in the ring.

Circle — A circle arc. A full circle is a circle arc 360° or more in length.

Constraint — A specific storage representation of a relationship between variables which limits the freedom of the variables, i.e., reduces the number of degrees of freedom of the system. Also, constraint is sometimes used to mean a type of constraint, as in "there are seventeen atomic constraints."

Constraint satisfaction — The process of moving variables so that all the conditions on them embodied in the constraints are met. It is not always possible.

Meta Platforms, Inc.
Exhibit 1140
Page 171

Copying — Duplication in storage the ring structure of a definition picture. A copy is not to be confused with an instance. Any instance may be changed into a copy by dismembering.

Definition — A master picture. Especially a picture to be used for copying, usually containing a combination of atomic constraints. Also the error computation routine associated with a constraint.

Delete — To erase. Deleted blocks become garbage.

Digits — A set of five decimal digits plus sign, leading zeros suppressed. As a variable digits may be moved, rotated, or made larger on the display. The particular value displayed is that of an associated scalar and may be changed only by moving the scalar.

Dismembering — The process of changing an instance into a copy by creating in the ring structure a duplicate of the internal structure of the instance's master and removing the instance. A dismembered instance becomes a group of lines, etc., which may be individually moved, deleted, etc. Dismembering peels off only one layer of instance at a time.

Dummy variable — A particular two component variable used to locate the arms of a constraint when it is first created. Dummy variables may merge with any other kind of variable leaving any attached constraints applying to that variable. Display for a dummy variable is a ×.

Error — The number computed by the definition subroutine for a constraint. Error is zero if the constraint is satisfied and grows monotonically as the constrained variables are moved.

File — A storage structure. A file may be in either list form or table form. Also a collection of magnetic tape records.

Free — A variable which has so few constraints on it that it may be moved to satisfy all of them. Such a variable will be in the FREEDOMS ring.

Garbage — Free storage inside the range of storage addresses being used to represent the drawing.

Hen — A pair of registers in a block used to indicate the first and last references made to that block by the chickens belonging in the hen's ring. Also called a key.

Meta Platforms, Inc.
Exhibit 1140
Page 172

Instance — A fixed geometry subpicture represented very compactly in storage by reference to a master and indication by four numbers of the size, rotation, and location of the subpicture. Internal structure of an instance is visible and may contain other instances, but since it is identical in appearance to the master it cannot be changed without changing the master. Except for size, rotation, and location, all instances of one master look the same.

Key — See hen.

Line — A line segment. No representation for an infinite length line exists in Sketchpad.

Line segment — A topological thing connecting two points. Contains no numerical information. Sometimes called a line.

List — A particular form of storage structure in which each element stores not only the information pertinent to it but also the address of the next element. Not to be confused with a table.

Location — A position in the coordinate system represented by a pair of coordinates. Not to be confused with a point which has a location. Also the address of a particular piece of information in storage.

Master — A picture which is used to define the visible internal structure of an instance.

Merging — Combination of two storage blocks to identify two picture parts, which must be of like type, permanently. The result of a merger of variables takes on the value of the historically older variable. In the ring structure, merging makes one block out of two, reducing the other to garbage. In certain cases merging is recursive.

Moving — Changing the numerical information stored in a variable. Moving a point stores a new coordinate location over the previous one. Moving an instance, text, or digits includes size change and rotation. Moving a scalar implies changing its value but does not change the position of its display. Moving is also the state a thing is in when it is attached to the light pen; it may be stationary on the display. Moving is not to be confused with relocating.

Meta Platforms, Inc.
Exhibit 1140
Page 173

N-component element

A particular form of storage in which various properties of each object represented are stored in consecutive registers. Also the block of registers representing an object.

Numbers

See scalars and digits. Number often refers to digits and scalars collectively. Also the binary numbers stored for a variable.

Object picture

A particular picture currently being worked on. Especially a complicated picture of particular interest to a user as opposed to a definition or master picture which is to be used as a portion of the object picture.

Older

The older of two blocks is the one with the lowest numbered address, illustrated higher on the page. Since new blocks are taken from the free space in addresses higher numbered than the drawing storage, an older block was usually created sooner.

Picture

A storage device to collect together related drawing parts. A "sheet of paper". Also the lines, points, instances, and constraints, etc., that are drawn in the picture, collectively. Pictures are numbered so that any one may be called to appear on the display. Within the limits of storage, as many pictures as desired may be set up and used.

Point

A specific representation in the ring structure used as an end point for a line segment. Not to be confused with location or spot. Also as a verb, to aim at something with the light pen.

Pointer

A storage register which contains the location of another storage register rather than numerical data. Such a register is said to point to the register whose address it contains.

Pseudo pen location

A location near the axis of the light pen which is used as the "point of the pencil". The pseudo pen location lies exactly on an existing point or line or circle or at the intersection of lines if the pen is aimed at them.

Relocating

Changing the address at which a particular block is stored in memory. Not to be confused with moving.

Result

The single thing which remains after two things have been merged.

Ring

A set of pointers which closes on itself. In Sketchpad all rings point both forward and back. A ring is composed of one hen and many chickens.

Meta Platforms, Inc.
Exhibit 1140
Page 174

Ring structure — The type of storage structure used to represent the drawing's topology. See ring.

Satisfy — See constraint satisfaction.

Scalar — A one component vector whose value can be displayed by a set of digits. For display of the scalar itself a # is used.

Spot — One of the bright dots on the display. Not to be confused with point or location.

Table — A form of storage structure in which successive pieces of information are stored in successive registers in memory. Tables are the "conventional" form of storage. See also list and ring structure.

Termination — The process of taking things out of the moving state. Termination is usually done by giving a flick of the light pen. Pressing "stop" also terminates. Upon termination, merging may take place.

Texts — Lines of textual material typed in and appearing in a standard type style on the picture. Text is treated as a four component variable.

Tie — An attacher.

Value — The particular information stored in the numerical portion of a variable. E.g., the location of a point. Especially the value of a scalar as opposed to the location of the set of digits displaying this value.

Variable — A picture part which contains numerical information. Scalars, points, instances, texts, digits and dummy variables are the only variables at present. Also used to denote a type of variable.

Meta Platforms, Inc.
Exhibit 1140
Page 175

BIBLIOGRAPHY

1. Clark, W. A., Frankovich, J. M., Peterson, H. P., Forgie, J. W. Best, R. L., Olsen, K. H., "The Lincoln TX-2 Computer," Technical Report 6M-4968, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Mass., April 1, 1957, Proceedings of the Western Joint Computer Conference, Los Angeles, California, February, 1957.

2. Coons, S. A., Notes on Graphical Input Methods, Memorandum 8436-M-17, Dynamic Analysis and Control Laboratory, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Mass., May 4, 1960.

3. Johnston, L. E., A Graphical Input Device and Shape Description Interpretation Routines, Memorandum to Prof. Mann, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Mass., May 4, 1960.

4. Lickleder, J. C. R., "Man-Computer Symbiosis," I.R.E. Trans. on Human Factors in Electronics, vol. HFE, pp. 4-10, March 1960.

5. Lickleder, J. C. R., and Clark, W., "On-Line Man-Computer Communication," Proceedings of the Spring Joint Computer Conference, San Francisco, California, May 1-3, 1962, vol. 21, pp. 113-128.

6. Loomis, H. H. Jr., Graphical Manipulation Techniques Using the Lincoln TX-2 Computer, Group Report 51G-0017, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Mass., November 10, 1960.

7. Moore, E. F., "On the Shortest Path Through a Maze," Proceedings of the International Symposium on the Theory of Switching, Harvard University, Harvard Annals, vol. 3, pp. 285-292, 1959.

8. Roberts, L. G., Machine Perception of Three Dimensional Solids, Ph.D. Thesis, Massachusetts Institute of Technology, Electrical Engineering Department, Cambridge, Mass., February, 1963.

9. Southwell, R. V., Relaxation Methods in Engineering Science, Oxford University Press, 1940.

10. Vanderburgh, A. Jr., TX-2 Users Handbook, Lincoln Manual No. 45, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Mass. July, 1961.

11. Walsh, J. F., and Smith A. F., "Computer Utilization," Interim Engineering Report 6873-IR-10 and 11, Electronic Systems Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., pp. 57-70, November 30, 1959.

12. Handbook for Variplotter Models 205S and 205T, PACE, Electronic Associates Incorporated. Long Branch, New Jersey, June 15, 1959.

Meta Platforms, Inc.
Exhibit 1140
Page 176

Biographical Note

Ivan Edward Sutherland was born on May 16, 1938 in Hastings, Nebraska. After an early childhood near Chicago, he moved to Scarsdale, New York where he graduated from Scarsdale High School. Mr. Sutherland was a George Westinghouse Scholar during his four years at Carnegie Institute of Technology, Pittsburgh, Pennsylvania where he received the Bachelor of Science degree in Electrical Engineering in June 1959. While at Carnegie he twice won the American Institute of Electrical Engineers Student Prize Paper Contest for District 2 (1958 and 1959). As a graduate student he held a National Science Foundation Fellowship for three years (1959 to 1962). He received the Master of Science degree in Electrical Engineering from California Institute of Technology, Pasadena, California in June 1960. From September 1960 to December 1962, Mr. Sutherland was associated with the Research Laboratory of Electronics at Massachusetts Institute of Technology first as a full-time doctoral student and then as a research assistant during the fall semester of 1962. During the summers of 1960, 1961 and 1962 he was a Staff Member of the MIT Lincoln Laboratory.

Mr. Sutherland is a coauthor of "An Electro-Mechanical Model of Simple Animals," (Computers and Automation, February 1958) and is the author of "Stability in Steering Control," (Electrical Engineering, April 1960). He is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, and Pi Mu Epsilon. Mr. Sutherland belongs to the Institute of Electrical and Electronics Engineers and the American Society of Mechanical Engineers.

Meta Platforms, Inc.
Exhibit 1140
Page 177