INFORMATION-PRESENTING SYSTEM

Abstract of the Disclosure

A special-purpose system for retrieving, presenting and editing written and other data. Microprogrammed microprocessors perform unusual functions of display and retrieval. Data is organized into addressable and rapidly modifiable streams, stored in a scattered manner; these streams are in turn agglomerated into complex files having unique susceptibility to inter-coupling and indirect control. The display microprocessor interprets commands of an idiosyncratic stream-addressing display language. These functions are combined to provide a linked parallel text retrieval and manipulation system.

Background of the Invention

Dynamic computer graphics have shown themselves to be of great use for a variety of purposes. An exemplary system of this type for pictorial drafting was that developed by Ivan Sutherland at M.I.T. Lincoln Laboratory, called SKETCHPAD. Another notable system is the NLS text facility developed by Douglas C. Engelbart at Stanford Research Institute, Menlo Park, California. Such systems allow users to create information (pictures and text in these two examples) and modify it speedily and with ease, through information-handling techniques which render the system sensitive to pointing actions by the user and then carry out the operations designated by the pointing.

Such systems have till now employed computer data structures of conventional types, computer file structures of conventional types, and other methods which render necessary the design of new and idiosyncratic data structures for every new system and new purpose. The data and file structures so far employed have restricted their users in a number of ways. First, they have made large quantities of fast memory imperative in a graphics console. Second, they have rendered necessary the use of a large main computer to drive the console and shepherd its information. Third, they have thrown up obstacles to the free transmission of complex information among graphics and other computer systems, forming a hindrance to the true "electronic publishing" of digital text and pictures.

It is to do away with these encumbrances, and facilitate a variety of other purposes in design, decision-making, scholarship, teaching and other fields, that this invention has been developed. The breadth of the intended contributions of this invention will become manifest from the disclosure.

Summary of the Invention

The development of computer graphics has provided numerous visual facilities for handling data at dynamic display screens. Text, diagrams and pictures are among the types of information now being usefully manipulated by persons using such screens.

However, these developments have been restricted by the specialization of individual systems concerned with specific problems. Stabilized techniques have not previously been evolved. The problem of sending complex information between systems, the control of text editing and anim ation, information retrieval, diagram manipulation and pictorial animation, have not been reduced to uniform and simple techniques which are generally applicable and versatile.

It is accordingly an object of this invention to provide a universal data system for the display, editing, manipulation and transmission of text, pictures, animation records and other forms of information.

It is an object of this invention to provide a means of digital text information retrieval which is quick and direct, not requiring numerous chained lookups or buffering sequences to obtain information quickly. Indeed, it is considered desirable to optimize such a search system in order to obtain quick-seeming lookup from slow memory devices such as cassette tape and DECtape.

It is another object of this system to provide rich facilities for the examination, indexing and search among interconnections of text by inexperienced users.

It is another object of this system to provide a general technique for annotation of text and pictures, and permit scholars, researchers or businessmen to intercouple annotations one to the other ad libitum.

It is an object of this invention to provide a means for the editing of digitally stored text by naive and inexperienced users with little or no preparation, and without the complex command protocols customary in digital text editing systems.

The invention which has been evolved to meet these objectives possesses a number of pleasing and unusual features.

A feature of this invention is a hierarchical storage structure combining great potential size with very efficient lookup and lookahead techniques.

Another feature of this invention is the ability to revise text or other data without summoning unaffected parts to memory or reblocking the data.

Another feature of the invention is the ability to display dynamically the contents of information complexes with incremental roving motions over large seamless data tissues, while meantime retaining the addresses of contents in core memory in stabilized form; and, indeed, these data tissues allowing incremental roving may be multidimensional.

It is a feature of this invention to permit a generalized form of text presentation resembling columnar printing, but to allow the user to call any two texts for this columnar arrangement, displaying, moreover, any linkages which may exist between any two texts so called, and retaining this display of links while moving the texts incrementally on the screen.

It is a feature of this invention that interconnections between files will be correctly retained despite changes in the original versions of the files.

It is a feature of this invention that it permits true electronic publishing of text, animated pictures and other data complexes with no access required to a large computer or even a general-purpose minicomputer.

Brief Description of the Drawing

The specific details of apparatus suitable for practicing the method of this invention, and their mode of functioning, will be described in full, clear, concise, and exact terms in conjunction with the accompanying drawing, wherein:

FIG. 1 shows the Parallel Textface in operation: a human user U employing light pen LP upon text panel T2 on which are visible completed links CL to text panel T1 and broken links BL indicative of linkages to text beyond the ranges of text displayed in panels T1 and T2. Furniture F is also visible on screen 5.

FIG. 2 shows detail of light pen control of coupled text in linked panels, already shown in FIG. 1. Text T1 is visibly linked to text T2 by completed links CL and implicitly by broken links BL. Light pen LP being used to impart downward (backward) text motion to "independent" text T1 is employed to adjust position of control pip P downward. Motion of P brings about downward motion of text T1 as seen at arrow A1, and also to complete links CL, as seen at arrow A2, and thus through motion of complete links CL imparts derivative motion of "dependent" text T2 as shown by arrow A3.

FIG. 3 is a schematic view of Basic File BF, composed of streams Data D, Data Control DC and Part Map PM. Additionally, links z1, z2, z3, z2a delineate Parts in D, and links Y1, Y2, Y3, Y4, Y5 from DC to D delineate portions significant to sequential or structured presentation of D.

In FIG. 4, a schematic view of a conventional computer display system, Disk Memory Dk is shown employing Disk Buffer DB as a temporary storage area for input and output to and from said conventional computer display system. Data is moved from Disk Buffer DB to Work Area WK under control of program in Program Area Pr, which then converts it to display list and places said display list in Display Buffer DiB, whence its contents are transmitted to Cathode Ray Tube CRT forming Image IM.

FIG. 5 is a schematic view illustrating the improved system XU herein disclosed, whose data goes to and from Disk Memory Dk under control of Search Microprocessor SEμP with use of its control areas CT and CLIT according to methods described herein. Data arriving from disk is emplaced in a single undifferentiated area of digital core memory DCM, whence images are derived by Display Microprocessor DIμP according to methods described herein, such images IM appearing on screen of Cathode Ray Tube CRT.

This view is given in more detail in FIG. 6. Two mass memories are shown, Century brand "Floppy Disk" FD and Cipher brand cassette drive CST. Search Microprocessor SEμP is shown to contain the elements of Select register SL, Data Buffer DB, Rapid Access Memory RAM, Read-Only Memory ROM1, Crum Table CT, Crum Level Identification Table CLIT, Keyboard Buffer KBB and Light-Pen Buffer LPB. Search Microprocessor SEμP is connected to input and output of data by means of Select Register SL, such input arriving in Data Buffer Register DB prior to passing to Crum Table CT or Digital Core Memory DCM, and such output arriving in Data Buffer Register DB or Digital Core Memory DCM before passing to floppy disk FD or cassette CST. Outputs from Random Access Memory RAM go to Select Register SL, Data Buffer Register DB, Crum Table CT and Crum Level Indicator Table CLIT.

Data Core Memory DCM is a conventional core memory or equivalent addressable read-write storage with normal read-write electronics including Memory Address Register MAR. It is divided for use into this system's singular Wraparound Buffer Beds, or beds for short, holding Text 1 T1, Text Control 1 TC1, Part Map 1 PM1, Textmaster 1 TM1, Counterpart Map from Text 1 to Text 2 CPM 1/2, Text 2 T2, Text Control 2 TC2, Part Map 2 PM2, Textmaster 2 TM2, Counterpart Map from Text 2 to Text 1 CPM 2/1, Left (broken) Links LL, Completed Links LC, Right (broken) Links LR. There are also conventional blocked buffer areas for Screenmaster SM, Furniture F and interpretive DINGO microprogram from Parallel Textface PTF, and Crum/Meander Repacking Buffer CMRB.

Rapid Access Memory RAM also stores divider location tallies for wraparound buffers T1, T2, TC1, TC2, PM1, PM2, TM1, TM2, CPM 1/2, CPM 2/1, TM1, TM2, LL, LC and LR, (all shown by dashed lines in their representations in TCM), such divider location tallies passing to Memory Address Register MAR as well as Crum Table CT.

Display Microprocessor DIμP contains Vector Buffer VB, Count Register CNT, Character Buffer CB, stack STK, and Read-Only Memories ROM2 and ROM3, variously interconnected. Character Buffer CB is shown receiving character codes from wraparound buffer bed for Text 1 T1, with count register CNT being supplied from textmaster 1 wraparound buffer bed TM1, Vector Buffer VB receiving vector line-drawing information from Furniture Buffer F. DINGO microprogram for Parallel Textface is supplied from wraparound buffer bed PTF to Read-Only Memory ROM2, whose subinstructional interpretation sequence is then routed through Read-Only Memory ROM3, with its operands pushed on Stack STK and being popped as required to Vector Buffer VB, Count Register CNT, Character Buffer CB or Read-Only Memory ROM2. Vector Buffer VB is connected to X Output Register OX and Y Output register OY, both connected to Cathode Ray Tube CRT. Read-Only Memory ROM3 is connected to X and Y Output registers OX and OY, to CRT Controller CRTC which is in turn connected to Cathode Ray Tube CRT. Light Pen LP receives light signals from Cathode Ray Tube CRT and is connected to Light Pen Buffer LPB, which is connected to Read-Only Memory ROM1. Keyboard KB is connected to Keyboard Buffer KBB, which is in turn connected to Read-Only Memory ROM1.

FIG. 7 is an excerpted conceptual schematic view of an Enfilade structure and its tree envelope, with arrows designating hierarchy, left to streamhead, right to stream tail, family directions to brother and cousin crums, Level Ø, Level 1, Level 2, Level (n-1) and Level (n), final meanders and the core window.

FIG. 8 is an excerpted conceptual schematic view showing the additive WID structure of an exemplary enfilade with 5 crum levels and an envelope with 3 levels of left-out left brothers LOLB L2, LOLB L4, LOLB L5.

FIG. 9 is a schematic view showing the structure of a conventional chained file CH containing an initial count c and a terminator tm, and pointed to from outside by an outside pointer op1.

FIG. 10 is a schematic view showing an Indexed-Block File IBF indexed by an Index Block IB and pointed to from outside by an outside pointer op2.

FIG. 11 is a schematic view showing a Generalized File with Data stream D1, Data Control stream DC1, Data stream D2, Data Control stream DC2, and Part Map PM.

FIG. 12 is an excerpted conceptual schematic view showing the general form of an Enfilade with hierarchical structure H, crums c, and meanders m comprising stream St.

FIG. 13 is an excerpted conceptual schematic view showing the correspondence of Crum-Level Indicator Table CLIT with Crum Table CT, as instantiated by CLIT pointers cp pointing to crums c.

FIG. 14 is an excerpted conceptual schematic view showing the structure of wraparound buffer or Bed B with Divider or Dam D capable of moving forward FWD or backward BWD, dividing Bed B into first section 1st beginning with first element F and ending with foot FT and second section 2d beginning with Head H and ending with Last element L.

FIG. 15 is a prognosticative excerpted conceptual schematic view showing the possible track of a Bed Pattern BP in a prospective forward babble along stream St, showing relations on stream St of First section 1st, second section 2d, current head CH and current foot CF, and divider D visualized twice in the two manifestations of its current position. NF and NH will be the New Foot and New Head, respectively, if and when present contents of location CF and CH are babbled out in the forward direction.

Description of the Preferred Embodiment

With all these things in mind, and referring to the accompanying illustrations, we will here present an embodiment of a desired general system for text and graphics, able to communicate with others of its kind and perform rich services for even the most naive, incompetent or distracted user.

Conventional computer graphics systems of the line-drawing type, such as IBM 2250, Digital Equipment Corp. 338, etc., generally have a core memory divided roughly into four parts as will be seen in FIG. 4. These four parts serve the following functions: a program area PR, holding a controlling computer program; an I/O buffer area DB, serving as a "way station" for information to be shown, into which the information arrives from the shepherding computer or from mass storage such as disk, tape, etc.; a work area WK, where the information from the I/O buffer area is transformed into suitable information for the display, and a display buffer area DiB, into which the displayable information is placed by the program.

It is from this display buffer area that some displaying mechanism or displaying program routes the information to its intended control of the screen image IM on cathode ray tube CRT. For each screen production this display mechanism or program steps through the "display list" of screen-controlling information. Typically it goes through this list some forty times per second or faster, effecting successive re-illumination of all parts of the screen display. Portions to be animated are modified in the display list on succeeding presentations, causing parts of the drawing or other features to change position, length or angle.

The alternative general arrangement of the system to be described here will be seen in FIG. 5. A microprocessor SEμP assumes the former search, retrieval and editing functions of the computer program, managing the lookup through reserved RAM memories CT and CLIT. Core memory DCM is undivided, having only a General Area which functions indifferently for data arrival, manipulation and display. The display functions of the system are executed by Display Microprocessor DμP. An unusual buffering mechanism permits user "roving" through complex data without the need of its being moved in core memory.

FILE SYSTEM

Because the system is intended to work with files of great size and intricacy, a very general file structure has been developed. Conventionally a file, shown in FIG. 9, is a succession of data blocks having a preset count c or a terminating marker tm. This makes editing by insertion, deletion or rearrangement cumbersome, for the entire file needs to be passed through and copied. A significant alternative, shown in FIG. 10, employs an index section, permitting simpler revision by the rearrangement of this index. However, as this type of file grows larger the rearrangement of the index grows increasingly cumbersome.

An additional desirable feature does not work sensibly with either file type. This is the provision of means for outside data to couple into the file: to indicate particular items in the file having a certain property, or certain historical origins, etc.; or, most important for features to be explained later, to make possible the creation and presentation of "parallel" text files.

This feature, if allowed for the file types mentioned, prohibits their future change, since an outside pointer in another location op1, op2 indicating an element in these files by numerical position would be disrupted by any changes affecting the numbering.

Similar difficulties hinder the use of such files for holding multiple data types. Either the data types must be arranged in fixed schemes of alternation (as in the conventional storage of mailing lists upon magnetic tape), or a distinct code must be assigned to each type of data and programs must then be used to search and screen these data type codes.

The file structure of this sytem, then (FIG. 11), allows quick retrieval and editing, may grow to great size, has a simplified lookup structure, permits coupling to elements within the file from other files despite positional change of the elements, and permits a wide variety of data structures without interference.

The basic unit of the file is the <u>stream</u>, conceived to be an unbroken countable succession of data elements. However, its storage may be scattered and irregular, allowing the scattered distribution in the storage disk and tapes of various pieces of the stream or <u>meanders</u> ; and so we employ another term, the <u>enfilade</u>, to a stream which has been divided and arranged for storage as shown in FIG. 12.

The structure of the enfilade, believed to be unique to this system, has been optimized for speedy retrieval and revision. The meanders m are pieces of the data stream St. Keeping track of their storage and retrieval are unusual pointers we call <u>crums</u> c (Code, Recursive and Universal, for Meanders).

The structure of the enfilade and crums, and the techniques of its retrieval and editing, will be described by-and-by. The <u>search system</u> is that part of the overall system employed for its retrieval and editing.

(The "stream" terminology, with its image of flowing water, is extended to cover a number of other parts and aspects of the system. Individual pieces of a data stream we call <u>meanders</u>. The area of core memory through which a stream moves we call a <u>bed</u>. The movement of a stream through its bed, a process believed to be singularly employed in this system, we call <u>babbling</u>. A movable dividing line in the bed, moved during the babbling process, we call the <u>dam</u>, or divider or terminator, and the pointer register indicating its location we call the <u>dam pointer</u>, or divider register, etc.)

The meanders of streams retrieved by the search microprocessor semp are brought to core memory dcm and related to the core memory by an unusual method, the buffering structure of the system, which includes beds, babbling and dams. This buffering structure permits complex data to move in and out of the system with little wasted space but also with little data movement around within the core memory.

In addition to the file structure of the system, which is concerned with the ways that the several undivided data streams are linked and manipulated, there is a <u>file-combining structure</u> of the system, concerned with techniques for creating and perpetuating linkages that survive modifications of individual files. All this is to be taught further below.

DISPLAY SYSTEM

The actions on the screen are controlled by the employment in the display microprocessor DμP of an unusual display language which we call DINGO (Display lINGO), whose pictorial operations are executed by the display microprocessor DIμP. DINGO also commands the retrieval acts of Search Microprocessor SEμP. Operations specified by DINGO are combined into "faces," or screen theaters, having particular properties and performing specified services.

We will explicate this overall system part by part, though it will be recognized that the close interdependence of the several subsystems renders exposition difficult.

STREAM-BASED FILE STRUCTURES

In conventional computer systems it is customary to store information in blocks, fixed-length storage units residing in numerically-designated subsections of mass-storage devices; or in what we may call "superblocks," consecutive sequences of such blocks which have been assigned an overall name, which may have the forms seen in FIGS. 9 and 10.

In the present system the unit of file reference is much larger and more fluid: it is the data stream, a sequence of data elements. The stream is of arbitrary length and subject to storage in broken and scrambled form in a variety of mass storage devices and core memory. Parts of it may be revised without calling unaffected parts into core memory or recopying them on mass storage. The enfilade system, to be described later, takes care of this storage and the inventory of the many component pieces or meanders, whilst meantime permitting rapid access to any part by means of an unusual lookup structure, [this to be explained after the basic stream system.]

Any data element is referenced principally by its position in a stream; any data sequence is likewise referenced principally by its starting position in the stream and its length. Finding such data elements by the stream-position designation is likewise a task of the enfilade system.

Given this technique of data reference by stream position, with retrieval carried out by the enfilade system, we define files and interconnections among them. Thus an element of data is referred to by absolute stream address, or "stream address" for short.

The simplest type of file is the one-stream file. This consists of one stream of data of any type. (An exemplary usage of such a file is detailed later, the JOT™ system.) However, the use of one-stream files does not permit the advanced forms of linkage desirable in many applications.

The typical file of the system for a single type of data, which we shall call the "Basic" File BF, consists of three data streams as shown in FIG.3 : Data D, Data Control DC, and Part Map PM. The function of the Data Control stream is to delineate segments of data as required for specific purposes, as by pointers to individual points Y1, Y2, Y3 and to data sequences y4, y5.

Streams DC and PM are <u>pointer streams</u> rather than data.

(Pointers that point outside a file we call Outpointers. These are not shown here. Typically for simplicity's sake we place all Outpointers in the Interfile to be described later, though exceptions can be made for various purposes.)

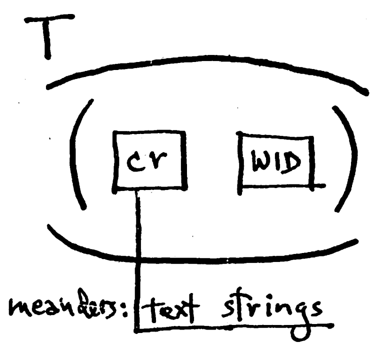
The data may be text or pictures or other things. If text, it is stored in D (then referred to as T) in the American Standard Code for Information Exchange (ASCII) and displayed by a character generator of a standard line-drawing type, like that on the DEC 338 and similar equipment. If pictures, they are stored in D (then referred to as P) as a series of standard binary numerical coordinates, representing positions of points on the screen to be connected by a bright line, and an on/off bit which may be used to suppress the brightening of the beam during its repositioning.

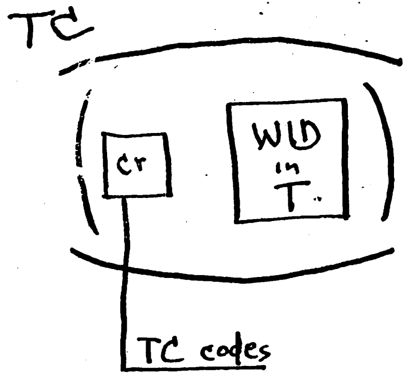
The Data Control Stream DC (or TC or PC for text or pictures) has the same sequence as the Data Stream: that is, the point selection codes in the Data Control Stream appear in the same order in the Data Control Stream as the elements they select in the Data Stream, and the sequence selection codes in the Data Control Stream appear in the same order in the Data Control Stream as the sequences they select in the Data Stream.

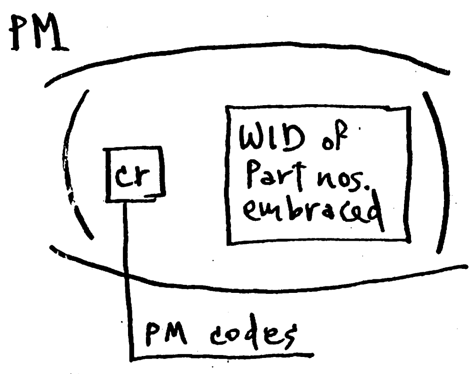
Understood most generally, however, Data Control DC may be thought of as an internal arranger of sections and pieces of file, functioning as a local control of parts which are to be controlled or animated separately in a screen display, or which have been grouped together in storage (in the stream) for some other purpose. It also contains markers inverse to those in the Part Map or external coupling structure PM, so that these markers are encountered during independent processing of the section.
Part Map PM may be thought of as the master lookup for all parts of the file, as well as
an outward-looking coupling structure providing reference to specific elements which is maintained regardless of changes in the file. Data streams D are the actual matter of the file, constrained to sequences fitting the DC sequence. (The linking mechanism between files, Counterpart Map CPM, is an exterior coupling relating to the outward-looking coupling structure, Part Map PM, from outside the file.)

This apparatus, then, furnishes a completely general system for exterior coupling among files having any contents or complexity. DC localizes and permits search for all parts of a file related to a particular purpose. Changes in the file, though may rearrange DC. PM provides exterior "handles" of constant relationship retaining their specificity to the contents of the file by coupling, both to the D streams themselves, and to DC, depending on conditions; usually the latter.
And CPM provides the actual recording of links between two files.

A brief consideration of the Basic File for text will show how text and its collateral information structures are broken into the three streams, composed as follows. A schematic representation of the three streams appears below, where vertical and
5 horizontal parentheses are intended to indicate replication laterally and hierarchically.







Stream 1: text (T).

Characters, spaces, punctuation, no special codes. Note that two spaces always precede a new sentence, even if that new sentence happens to be the beginning of a paragraph. This is to assure compatibility between the Basic File for text and the one-stream JOT file (to be detailed).

Stream 2: text control (TC).

Includes text codes (paragraphs, italics, indents, page numbers, illustration numbers, captions, headers, footings, footnotes, jumps). Part numbers by ascending text position (separate beginning, end). Partitions dividing off separated parts, e.g. footnotes.

Stream 3: Part Map (PM).

Numbered points and strings within T, arranged by ascending part number independent of their positions in T.

Let us follow this brief description of the Basic file with a fuller and more exact description of its structure as generalized for all purposes, which we shall call the "Generalized File" GF, as illustrated in FIG. 11.

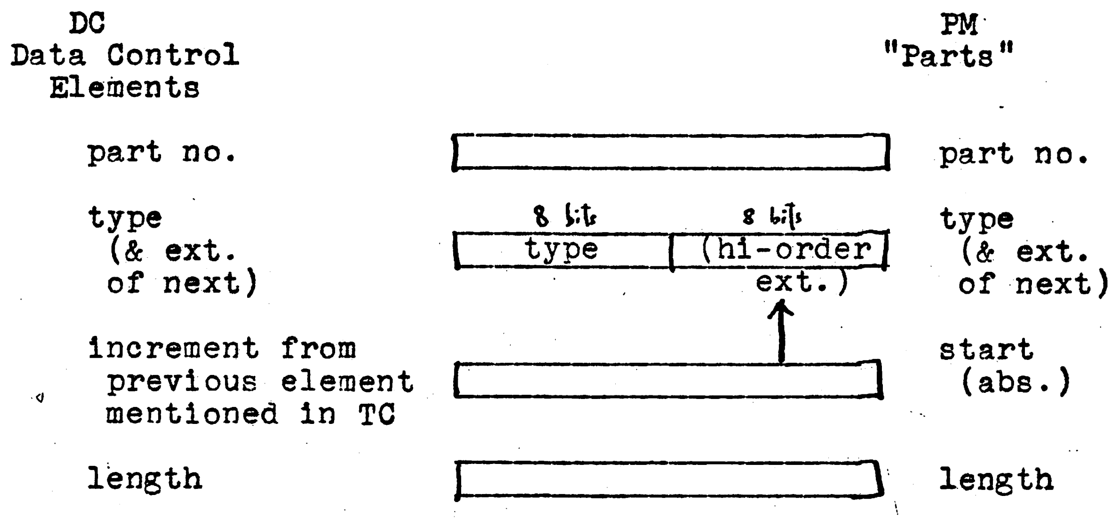
The Generalized file consists of any number of data streams as required for specific purposes. The division into these data streams may be by type or by independence. In division by type-- for instance, having stream A be text and stream B be coordinate-sequences for drawing pictures a line at a time-- the streams are divided according to type of data. In division by independence, seq uences of data of similar type may be divided into separate streams when independent access to their contents will be needed. For instance, a body of text may be broken into two independent streams if a section of one such stream is intended to remain fixedly or move slowly on a disp lay screen and the contents of the other stream are intended to move rapidly on the screen or be subject to frequent replacement.

In this Generalized File, each Data Stream has its own Data Control Stream, delimiting points and segments within it. The Part Map, however, designates points and segments, in any of the Data and Data Control Streams, and a specific Part may be defined as a plurality of points and segments in a plurality of streams.

These structures are basically the same for all data types, but the length of the elements will vary. For instance, text is stored in the Data Stream as 8-bit bytes, counts being in these units, while pictures are usually stored in 10-bit coordinate pairs filled out with various logic and filler bits to 16-bit data pairs; counts are then actually by data pairs totalling 32 bits each, though for consistency all streams are addressed by 8-bit bytes.

The following are the codes shared by TC and PM in the Basic text file. Each code consists of four 16-bit words. The slight differences between the two stem from TC having two codes for the end of a stream.

SHARED CODE FOR TC & PM (4 WORDS)



The type codes are:

type, 8 bits
(the leftmost of these eight bits is used only by TC stream, which has a separate code for the beginning and end of a designated text string. Leftmost bit =1 indicates that the code points to the right end of such a string. "Length" field nevertheless is the same, appearing identically in left and right string-end codes and giving the length of the string.)

type no. (octal;
right string-ends
begin with "2"
instead of zero)

000	Not used
001	"Part" delimiter (not used by PM)
002	Title
003	Author
004	Abstract or footnote
005	Footing

006	Paragraph
007	Squared paragraph
010	Indent from here, one "tab stop"
011	Page
012	Italic field
013	Boldface field
014-17	Reserved for other typographical fields
020	Footnote hole (Note: a footnote is stored as a "hole" in the text and its own associated text. Such text is stored last in the Basic Textfile.)
021	Footnote matter or text.
022	String association (associates note with string in text rather than hole or point, like marginal note rather than footnote)
023	Optional jump elsewhere in text.
031	Illustration point
032	Caption point
033	Caption matter or text
034	Other text partition
041	Quotation
042	Origin of quotation
043	Annotated point
044	Annotated string
045	Annotation matter
051	Picture
052	Difference picture (adding to picture provides expansion or shrinkage)
053	Screen furniture element (Note: inclusion of screen furniture within stream file system places them within overall stream lookup structure, even though core-resident)

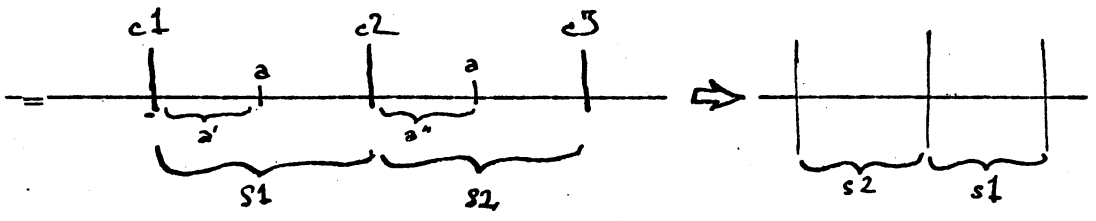
054	Associated subroutine for lightpen query of screen furniture element <u>n</u>
055	Coupling Nodes of Picture (Note: these are the points of the picture to which reference coupling may be made by other pictures and animations.)
056	Animation sequence (a series of positions)
06X	Clock elements linked to by animation (pictures, timers, pen-pointings. Type of animation specified by last 3 bits: independent animation, clocked animation with linear timing elements, clocked animation with nonlinear timing elements)

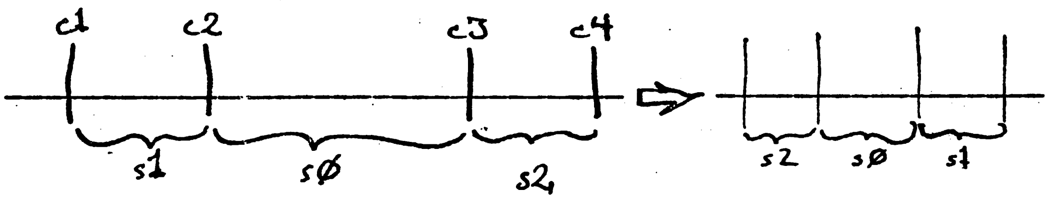
An extension of this method, desirable for large files, would be to assign ordinal codes within and among each type of item within the TC/PM numbering system. In the current method, to find footnote 43 it is obviously necessary to scan TC to the 43d footnote code; ordinal numbering would not require scanning from the beginning.

A somewhat better method for large files under which is desired random access to numbered elements of a certain type, creates additional lookup streams devoted exclusively to elements of the desired type or types; in which case ordinal numbering is optional but not strictly necessary.

Editing operations upon the data streams place certain requirements upon the pointer streams. During edit operations upon the streams (as managed by the Enfilade System), the stream addresses of specific data elements will change. For instance, if _m_ data elements previous to element _a_ are inserted,

the new stream address is the old one decreased by the number m; similarly, if _n_ data elements are inserted previous to _a_, the new address of _a_ is specified by the old address increased by _n_. If the information is rearranged with a three-cut rearrangement, as shown below, or a four-cut rearrangement,





the lengths of the segments created by the cuts are as follows:

3-cut rearrangement	Length of segment s1 = c2-c1
	Length of segment s2 = c3-c2
4-cut rearrangement	Length of segment s1 = c2-c1
	Length of segment sØ = c3-c2
	Length of segment s2 = c4-c3

Thus the new addresses in the streams are given by the following formulas (we denote the addresses of cuts as the addresses of element positions which they follow):

	Old address of <u>a</u>	New address
3-cut rearrangement	In s1	a+(c3-c2)
	In s2	a-(c2-c1)
3-cut or 4-cut rearrangement	Ahead of c1	unaffected
	After c4	unaffected
4-cut rearrangement	In s1	a+c4-c2
	In sØ	a-(c2-c1)+(c4-c3)
	In s2	a-(c3-c1)

Therefore editorial changes in the data streams require the corresponding modification of the pointer streams. These are done one way for the Data Control Streams and another way for the Part Map. (However, the same formulas are used in both cases for address recomputation.)

Data Control Streams are modified correspondingly to the editorial change in the Data Stream. When material is added to the Data Stream, the system checks to see what new Data Control Elements are required, and creates them if necessary; it then increments that incremental address following the insertion-point a (and therefore affected by the change) by the length of the inserted elements.

For a deletion in the Data Stream the system determines whether any elements designated in the Data Control Stream have been deleted, and deletes or modifies the corresponding Data Control Elements, decrementing the affected incremental all address following that of the deletion by the length of the deletion.

For a rearrangement, the system determines the number of cuts determining the rearrangements and modifies the affected addresses of all codes corresponding to points in the Data Stream between the first and final cuts by the formulas in the last table.

In addition, however, the Data Control Stream itself is subject to rearrangement, to preserve the correspondence of the elements of the Data Stream and the Data Control Stream. (This rearrangement is not done if only one segment of the Data Stream contains correspondences to Data Control Elements, or if none do.) Affected incremental element counts are also modified in the Data Control stream according to the appropriate formulas.

Because the parts of the Data Control Stream are in corresponding order to those of the Data Stream, the addresses in the Data Control Stream need only be changed between the first and last cut of a rearrangement.

Editing operations on the Part Map are also undertaken when Data Streams are edited. These conform to the same formulas for address change. However, the entire Part Map must be scanned, since the order of its parts does not correspond to the addresses they point to; each affected address is replaced by its new altered address during the scan, according to the formulas given.

The intended use of the system concerns not just individual files but couplings between them, so that one file may access and employ the contents of other files freely. In addition, it is desired that any file may be sensible to be changed, yet still remain true to its previous couplings.

This we accomplish by means of an intermediate file, or Interfile for short. Each pair of files having couplings between them normally require an Interfile. The Interfile is typically composed of two streams: the Counterpart Map 1/2 and the Counterpart Map 2/1.

These Counterpart Maps (CPMs) are data streams specifying couplings and correspondences between Parts in two files, by Part no. Between two Basic Files 1 and 2, Counterpart Map 1/2 lists its correspondences in the order of the Parts of File 1. Counterpart Map 2/1 lists its correspondences in the order of the Parts of File 2.

Thus Counterpart Map 1/2 says what parts of File 1 are coupled to parts of File 2. Counterpart Map 2/1 says what parts of File 2 are coupled to parts of File 1. (The assumption here is of directed or asymmetrical links; but if the links are symmetrical the two Counterpart Maps have identical content in different sequences.)

The CPM code is eight 16-bit words, viz.:

Counterpart number (2 words; ascending in order of historical creation)

Type code (same as TC/PM type code)

Part No. in File 1 (2 words)

Part No. in File 2 (2 words)

Unused.

In some circumstances it may be desired to keep an historical record of changes in and between files. This is done with the addition of historical streams in the File and Interfile. These historical streams then permit backtracking to any previous state of the file.

These streams we call Historical Push, which holds the record of each change (made consecutively and of identical length), and Historical Matter (which holds the actual contents of each insertion and deletion and is of arbitrary length). In order for historical search operations to go incrementally either backward in time (deconstruction) or forward in time (reconstruction), the length, location and content of both deletions and additions must be supplied.

HISTORICAL PUSH CODES, General File

Insert, Delete	Historical Op. No. (2 words)
	Abs. time (1½ words)
	Stream no. within file (halfword)
	Length (2 words)
	Location (2 words)
	Start, End in Stream (4 words)
	Start, End in Historical Matter (4 words)
Rearrange	Historical Op. No. (2 words)
	Abs. time (1½ words)
	Stream no. (halfword)
	Char. after cut 1: old, new (4 words)
	Char. after cut 2: old, new
	Char. after cut 3: old, new
	Char. after cut 4: old, new (zero if 3-cu)

Historical Matter Streams may contain material inserted or deleted in any of the Data Streams, regardless of type.

It will be noted that a single complex editing operation may affect a number of streams (as when a picture with text annotations is moved). Such an operation then has several historical entries bearing the same operation number.

The optional historical code for the Interfile is similar, but concerns only changes occurring between files. It contains the following fields:

```
Interfile Op No.

File 1 Op No. (each interfile operation likewise
                participating in, and corresponding
                to,
                operations within the linked files)
File 2 Op No.

Abs. time

Copy                 \
                      } Matter
"Transport"          |
   (move between     |
   files)            /

Make    \
Destroy  }  Link
Move    /

Location(s)
```

In order to provide relative synchronization for historical operations between files, each file-change operation is assigned an Interfile Operation Number and stored with the number of the most recent operations occurring in File 1 and File 2.

There is also an Interfile Historical Matter stream.