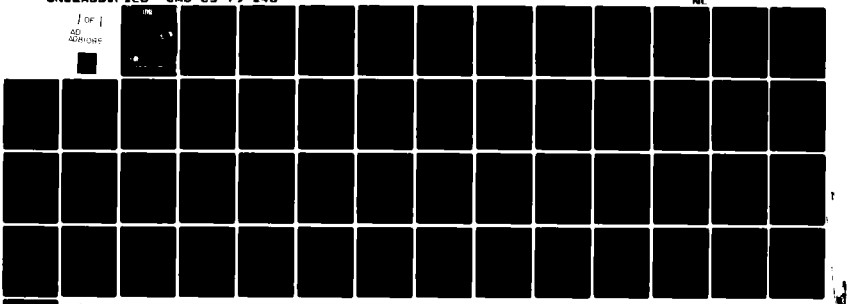
AD-A081 088 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 5/8
THE ZOG APPROACH TO MAN-MACHINE COMMUNICATION.(U)
OCT 79 G ROBERTSON, D MCCRACKEN, A NEWELL N00014-76-C-0874
UNCLASSIFIED CMU-CS-79-148 NL



END
DATE FILMED
3-80
DDC

The ZOG Approach to Man-Machine Communication

G. Robertson, D. McCracken and A. Newell

Department of Computer Science
Carnegie-Mellon University

October 23, 1979

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

(14) CMU-CS-79-148

(9) Interim rept.,

(6) The ZOG Approach to Man-Machine Communication.

(10) G. Robertson, D. McCracken, A. Newell

Department of Computer Science
Carnegie-Mellon University

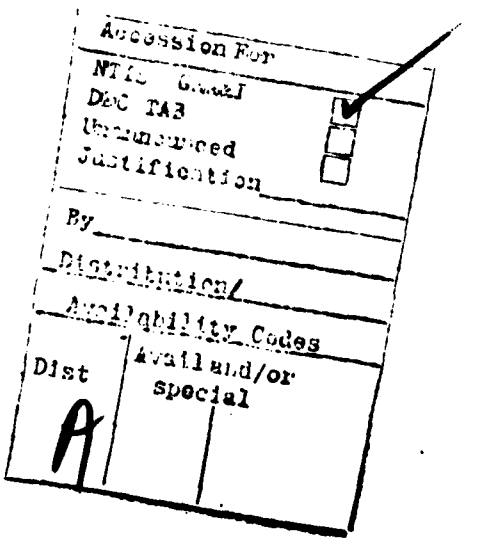
(11) 23 Oct[redacted]79

(12) 55

(15) N00014-76-C-0874
F33615-78-C-1551

This work was supported by the Office of Naval Research under contract N00014-76-0874. It was also partially supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-78-C-1551.

Table of Contents



The ZOG Approach to Man-Machine Communication

G. Robertson, D. McCracken and A. Newell

Carnegie-Mellon University, Pittsburgh, Pa. 15213, U.S.A.

ZOG is a rapid response, large network, menu selection system used for man-machine communication. The philosophy behind this style of communication was first developed by the PROMIS (Problem Oriented Medical Information System) Laboratory of the University of Vermont. ZOG has been used in a number of task domains to help explore the limits and potential benefits of the communication philosophy. This paper discusses the basic ideas in ZOG, describes the architecture of a system implemented to carry out that exploration, and discusses our initial experience.

1. Introduction

We have hardly begun to understand how to communicate effectively with computers. In part, the problem is one of evolution. As the computer continues to evolve into more powerful forms, the resident systems become more extensive and intelligent, requiring and supporting more sophisticated dialogs. Nevertheless, relatively few studies have explored the man-computer interface. There certainly is no well developed theoretical framework for evaluating and improving interfaces. In part, the problem is one of invention. We have been so bound by technology (removing obvious annoyances of existing interfaces and getting the bandwidth up, all within bounds of economy), that we have not addressed substantive issues of man-machine communication.

What would be an ideal communication medium between people and machines? No one knows. We have a tendency to reach in two directions for the answer. One is reaction to present interfaces. Yesterday it was 1200 baud terminals with a modicum of "intelligence" (meaning local computing power). Today it is personal computing with integrated graphics capabilities. The other direction is communication with our fellow man. We wish to speak and

listen, using natural language, and also to paint and sketch. These skills of expression seem ideal because of their long natural development.

If we can draw any lesson from the development of computers it is that we should seize on any notion that seems to expand the frontiers of the possible. Most such efforts will be stillborn, not liberating or not technologically feasible. Even then, if done with some scientific curiosity and attention, they can leave a residue that will help later on. Occasionally, however, such probes can set the stage for new developments.

We report here one such probe, a particular interface for man-computer interaction that has some novel properties. The basic idea is not ours. It was originated by the PROMIS group at the University of Vermont Medical School, who designed and implemented a complete system incorporating the basic scheme, and brought it to practical use. See Schultz and Davis (1979) for a discussion of PROMIS. We call our version of the scheme ZOG (which stands for nothing, but is short, easily pronounced and easily remembered).

The scheme itself is easily stated. Communication is via menu selection, where the menu is displayed on a video terminal with a touch screen. The user touches the option of his choice and is instantly shown a new display, called a frame, with further options relating to the chosen topic. When the user makes a selection, some system action may be performed in addition to displaying a new frame. The network of such frames is large enough so that all communication is by this means; in practice the nets are very large indeed. Thus, communication from man to computer is by discrete selection of semantically meaningful options. Communication from computer to man is by visual display of natural language text in a structured format.

In PROMIS, the interface is embedded within a larger system, called the Problem Oriented Medical Information System, which is the main concern of the PROMIS group. Though they have had a system running for almost seven years, its impact on the development of man-machine interfaces generally has been small. Nevertheless, we know of no of other system with the same essential features. The recent, extremely interesting, development of the British Post Office Viewdata system, described by Ford (1979), does share some characteristics, but is tied to the mass television market and its limitations. We were

sensitized to the PROMIS Lab work from an attempt of our own in 1972 to produce a similar system, described in Newell, Simon, Hayes, and Gregg (1972). Since menu selection is a common technique in man-machine communication (see Robertson (1978), Martin (1973), and Newman and Sproull (1979)), the potential of PROMIS is not easily appreciated. We decided to attempt to extract the scheme from its habitat in the PROMIS application and to study and exploit it as a general communication interface. Our goal is to find out whether this interface does indeed have the potential it appears to have, to demonstrate it, and to study its parameters in order to understand and optimize it.

This paper is a report on our progress toward this objective (see Robertson, Newell and Ramakrishna (1977) for an early description of this effort). The main sections of the paper introduce the basic ideas in ZOG and describe the system architecture. In subsequent sections we cover some application areas we have explored, some initial issues that have emerged, the potential of the ZOG philosophy, its cost, and some thoughts about next steps.

2. Basic Ideas in ZOG

We will first describe the ZOG system and how it operates. The user faces a terminal which is displaying a frame. Figure 1 shows a typical ZOG frame. There is text at the top, a list of options below the text, a column of pads at the right side, an area called a workspace below the options, and a horizontal line of pads at the very bottom. The user, at his discretion, selects one of the options or pads. Suppose he selects option 3. Immediately, the frame on the display is replaced by a new one with all the same parts: text, options, vertical column of pads, horizontal line of pads and workspace. Most of the content will be new except for the horizontal pads, which provide a continuously available set of search and help functions. For example, selecting the "back" pad (b) will cause a redisplay of the frame of Figure 1. Selecting some pads or options will initiate various actions. If the user had selected the PRINT pad, the textual information on this display would have been output on the printer.

That is all there is to ZOG as far as external mechanics are concerned. The user traverses a sequence of frames of his own selection, acquiring the information therein and taking the actions offered to him. It stands, at this level, simply as a menu selection scheme, distinguished

```
General background on ZOG                                    ZOG3

ZOG is part of a research effort to understand communication between humans
and computers.  Various aspects of this research effort are described below.

   1. System specifications of ZOG

   2. What is ZOG used for -- functional characterization

   3. Scientific issues behind ZOG

   4. Who is doing ZOG? Where? When? What sponsors?

   5. Prior research and antecedents

   6. Examples of ZOG projects (real and in progress)

   7. Developing your own ZOG net                            P-PRINT

edit  help  back next mark return  zog  display user  comment  goto find  info
```

Figure 1: Typical ZOG Frame

only by its ability to take actions in addition to presenting knowledge (a property shared by many other menu selection schemes).

A preliminary notion of how ZOG can be used is also needed at this point. Like any general purpose interactive programming language, it can serve in any communication capacity: for example, as a command language, data base retrieval system, CAI system, guidance system, interrogation system, or question-answering system. Also, like any programming language, what it is good for, as opposed to what it can conceivably be used for, is not determined by gross structure, but by more subtle features of operation. However, it is important to understand that ZOG is used not only for initial guidance or with novice users, but also for skilled operation; and not only for exploring knowledge bases, but also for taking action.

In PROMIS, such a system is used as the sole interface in accomplishing the total set of hospital functions on a ward: keeping patient medical records, taking patient histories directly from patients, prescribing drugs and treatments, monitoring patient progress, checking treatments for side reactions, and retrieving medical knowledge. It is used by physicians, patients, nurses, paramedics, and administrative people. It performs a full range of

communication functions with users who range widely in sophistication and in direct skill with the system. The communication interface is only a part of the total system that accomplishes all these functions, but it is a central one.

We can now enumerate the additional basic features of ZOG and fill in the missing design specifications. Some features are more central to the design than others, and the amount of evidence for each feature's role varies. But together they define the ZOG system. We list them in Figure 2 and discuss each of them below.

1. Rapid response. Upon selection the new frame appears instantly.

2. Simple selection. The user's act of making a selection is a simple unitary gesture.

3. Large network. The network of frames is large enough to accommodate all communication and knowledge-exchange with the user.

4. Frame simplicity. The frame display is simple enough to be easily and quickly assimilated.

5. Transparency. The entire system is completely open and understandable to the user.

6. Communication agent. The system is usable with existing programming systems to provide guidance in how to use the programs and how to interpret their results.

7. Active response. The selections can evoke actions that accomplish tasks other than just movement in the network.

8. Subnet facilities. There exists a hierarchical data organization for networks.

9. Modifiability. The user can modify and augment the network.

10. External definition. An external data format exists that completely defines a ZOG frame library.

11. Uniform search. A uniform scheme exists for searching, traversing and orienting in the network.

Figure 2: Basic Features of ZOG

2.1. Rapid Response

Upon selection the new frame appears instantly. "Instantly" is defined with respect to the user: fast enough so the user feels the flow of frames to be limited only by his own volition. If traversing a highly familiar network (as in specifying some operand to be worked on), he can move through the frames almost as a single skilled gesture. If he wishes to take a quick glimpse of a next unknown frame, he feels no hesitation because he need only wait for two responses (one down, one back).

How fast "instantly" must be in seconds is not fully known. PROMIS operates with 0.25 seconds response 70% of the time. The ideal speed is not likely to be much slower than this. One version of ZOG operates with 0.10 seconds response in order to permit exploration of this parameter.

2.2. Simple Selection

The user's act of making a selection is a simple unitary gesture. The time it takes the user to make a selection acts in series with the response of the system; it must be equally rapid. Its speed has two aspects: learning what the response should be (since new selections are always occurring), and executing the response. PROMIS uses a touch screen which solves both these problems: the user simply touches the display at the area where the option is stated. ZOG also uses a touch screen, and in addition uses single character selection from a keyboard.

2.3. Large Network

The network of frames (called a ZOGNet) is large enough to accommodate all communication and knowledge exchange with the user. "Large enough" is again defined with respect to the user. It means that at every frame there are options to be taken that deal with whatever information, help, elaboration and explanation is required. The options lead to other frames which also provide whatever is necessary. The user finds himself in a world where all of his

questions and all of the data he requires have already been laid out in advance in the network of frames -- where his needs have been anticipated. (An important exception to this will be taken up in Section 2.6 on ZOG as a communication agent.)

How many frames constitutes "large enough" is not known. The total system of frames is a finite state system. Considering the combinatorial nature of life, the conclusion could be that finite networks are incapable of satisfying this requirement for any interesting task. (Recall Chomsky's (1957) early stated view that no finite state grammar can adequately portray a natural language.) Without doubt this design feature leads to large networks. The PROMIS network appeared to satisfy this requirement with about 30,000 frames; it may eventually grow to 100,000 frames. None of our nets approach this yet. As with the criteria for rapid response, perfection is not necessary; the user need not remain within the frame system 100% of the time. In PROMIS the user types in a response rather than selecting an option (indicating the absence of an appropriate precoded response) less than 1% of the time. Not much is known about the details of this criterion or good measurements of it. What is known is that PROMIS has produced one system that clearly is adequate, in a general enough environment to engender some optimism.

2.4. Frame Simplicity

The frame display is simple enough to be easily and quickly assimilated. The power of the technique comes from the fractionation of the total task into small communication pieces with control by the user of which of these he wishes to acquire. If each frame were, say, like a textbook page, then substantial assimilation would be required, and the user would be thrown back on his own scanning and organizing resources without any help from ZOG. The natural criterion is that the user should never have to acquire knowledge other than what constitutes his final solution or what is necessary to find this solution (by the nature of the task, not of ZOG).

What is "simple enough" is relative to the user. Even less is known about it or how to measure it than about rapid system response or large networks. Perhaps it can be measured structurally by amounts of text and options; perhaps by the average residency time per

frame; perhaps it requires knowing exactly what knowledge was absorbed. PROMIS frame design, which has evolved over several years, tends toward a few sentences of text and no more than half a dozen options.

This design constraint is common to all menu selection systems and does not seem unique to ZOG. It is useful to stress it, however, because it implies even larger networks than would otherwise be the case. Faced with the large network problem, a natural engineering inclination is to permit the knowledge per frame to increase. This simplicity principle inhibits that inclination.

2.5. Transparency

The entire system is completely open and understandable to the user. It should seem totally open to him exactly why the system is doing what it is doing, and what it takes to obtain more information from it or to get it to do something. It should appear completely controllable and non-mysterious. The effect is stated in terms of user's perceptions, since what counts is the way the user reacts to it. The ideal user's model should be that of the "perfect instrument".

The specifications already laid down approach meeting the requirement of transparency: menu selection, rapid response, large network and simple frames. This creates a structure that is simple in concept and completely under the user's control. However, within these constraints it would be easy to realize obscure networks. Thus, the burden of this specification falls on the details of network and frame-content design. ZOG, like any programming system, creates an architecture within which one writes programs. The programs for ZOG are networks of filled-in frames. Thus this requirement can be taken as a requirement on programming style.

2.6. Communication Agent

The system can be used with existing programming systems to provide guidance in how to use the programs and how to interpret their results. This is not a requirement that derives

from PROMIS, but from our earlier attempt to build a similar system. PROMIS is an active system in that the selections can execute programs, but it is a closed system implemented on dedicated hardware. If all programs for ZOG must be coded specially for it, an entire world of applications is excluded. It then becomes a particular form of interactive programming language, though with many special features. But if it can somehow be integrated with any existing programming system, then it becomes a much more flexible tool.

This requirement is realized by making ZOG a communication agent. That is, ZOG sits astride the communication path between the user and an arbitrary program, called the subjob. Thus ZOG can monitor, interpret and modify the input and output streams in both directions. There is a communication language associated with ZOG itself, but it does not operate as a regular programming language. Rather, it translates the user's selections into messages to send to the subjob. It also monitors the subjob's output to the user and can translate these into new messages to the user or into selections, which cause new displays to be shown to the user.

This language will be described in the section on the architecture; it is simple in concept and implementation. The essential point here is that ZOG can operate as a front-end for any program whatsoever. It can provide explanation, instruction and guidance in working with a system. Likewise it can interpret the output for the user and suggest what to do. To do any of these things, of course, requires a ZOG program (i.e., a network) specifically designed to know about the subjob. How good a job it does depends on the size and sophistication of the network.

An alternative way of describing this structure is that ZOG is a command language. It is through ZOG that the user makes contact with all the resources of a computer, executes programs, and does his housekeeping (e.g., file management and message handling). Its usefulness as a command language, of course, arises not from any special logical character, but in its potential for explaining and guiding.

The use of ZOG as a communication agent necessarily carries with it the potential to violate transparency. ZOG clearly need not know all about the programs for which it fronts. In general, it cannot do so (e.g., if permitting access to another programming language such as

ALGOL68 or LISP). Thus, the user will generally see a mixed system in which he will be aware when the subjob is running and he is no longer dealing with ZOG.

The ability to have a system that can interpose between the user and any other job running on the host computer requires certain capabilities in the resident operating system within which the communication agent and the other job run. Such capabilities exist in the operating systems for the computers on which ZOG currently runs, but they are not necessarily available in all operating systems.

2.7. Active Response

The selections in ZOG can execute programs and take actions other than just search and retrieval from the network. Thus ZOG is an active system rather than just a data retrieval system, though the latter aspect is implicit in the large network. This capability could be realized directly in a programming language designed within ZOG; instead it is realized by ZOG being a communication agent, as described above.

2.8. Subnet Facilities

ZOG incorporates a hierarchical data organization for networks, called the subnet, which operates essentially like a subroutine. Subnets have names and can be handled as units. The user can orient himself by subnets, always getting back to the top of a subnet. There can be subnets within subnets recursively.

Given modern practice in data structures, there is nothing striking about this requirement. Nevertheless, it is important for the creation and modification of networks, and for the efficient implementation of large networks. Its importance for the user's operation in the net is less clear and there are no strong grounds for insisting on it (rather than simply having the user see the network as one large integral structure).

2.9. Modifiability

The user can modify and augment the network to suit himself. The goal of transparency by itself probably requires that a user be able to make small changes easily to any existing network, representing his own understanding and preferred way of dealing with the material of the net. There are some closely related additional reasons, such as increasing efficiency. But, as with the communication agent requirement, additional worlds of application become possible if networks are easily constructed and modified. Users can extend explanations, not just to clarify what is there, but in an application centered around the growth of the ZOGNet. Full retrieval systems, involving both augmentation and access, become possible, as opposed simply to searches of fixed data bases.

The requirement for easy modification and augmentation implies that an editor for frames and nets become an integral part of the system. There would be a requirement for an editor in any event, but if it were viewed as just for some class of professional "net builders", its facilities and its integration into ZOG could be seen as much less crucial. In the current scheme, the ZOG editor (called ZED) can be evoked as a selection from any frame.

2.10. External Definition

ZOGNets can be completely defined in an external data format. Because frames are display structures there is a temptation to define them only as internal data structures. In fact this was true of the initial PROMIS system (but not of the current system), and we expect it is true of most menu selection systems. However, considerations of ultimate portability require that some machine independent definition of a ZOGNet exist.

This requirement takes on the status of an imperative as soon as large networks are contemplated. A network of 50,000 frames represents a great body of knowledge (like a 5000 page text book), and becomes extremely valuable. It is easier to get a new ZOG system up and running on a new computer than it is to create 50,000 frames. Under these conditions, portability of the frame library is mandatory.

This requirement interacts with the specifications that ZOG be active and be a

communication agent. If ZOG had associated with it a full programming language, large amounts of the knowledge in a ZOGNet would be encoded in procedures in this language. Because the operating environment of this programming language is the display, it is much less perspicuous than most standard higher level languages. Documentation of these procedures so that the network becomes in fact portable and maintainable is a serious issue (and this is proving so with the current PROMIS system). The ZOG communication language is quite limited and simple (e.g., it has no conditional or loop control), so that essentially all of the knowledge in the net is encoded in the surface structure of frames and their connections. Thus it turns out that a simple representation involving strings of alphanumeric text is quite adequate.

The solution to external definition we have adopted is to embed our simple format within a bibliographic system which is in public use at CMU, called BH (Newcomer, 1976). BH provides the requisite data handling and printing facilities, and is oriented towards relatively large files. It is described in the section on system architecture. Thus ZOG does have a complete external definition simple enough to permit exportation.

2.11. Uniform Search

ZOG has a uniform scheme for searching and orienting in the network. Transparency requires that the user find the system extremely easy to understand and deal with, even when working in networks new to him (which will often occur in acquiring bodies of new knowledge). The operational aspects of how the user finds his way in the net are as important as the simplicity of the frames themselves. The difficulty is that the frame provides only a small and local view on the world so that the user must move around in the net to acquire knowledge, leaving all but the current frame out of sight.

Part of the solution to this is a set of tools for orientation and movement within the network. At any frame, the user may examine the list of frames he has recently visited, the list of frames which point to the current frame, and a list of frames explicitly marked by the user. There is also a facility for searching the network for any specified text string.

Another important part of the solution to uniform search is a set of conventions for how

any net is structured, which the user may rely upon and become familiar with. We have formulated a set of principles and conventions that seem reasonable based on the experience of PROMIS and our own explorations. But there is little data as yet to support our particular set.

The uniform search requirement is a constraint on network structure, on the content of frames and their internal arrangement. No features of the architecture proper reflect it. As a programming system, ZOG is defined and usable independent of this requirement, just as a computer is defined without specifying an operating system or an assembler. However, just as with an operating system, ZOG cannot be run without something that provides tools for searching through the net. Thus, we have included this requirement here, even though it is a pure ZOG-software requirement.

The solution adopted in ZOG attempts to satisfy the following principles:

1. No sudden death. No selection taken by a user produces a change that is irreversible. This applies both to movements in the network and to actions taken. Where this is not possible explicit confirmation will be required.

2. Standardized pads. There is a set of selections with standard names that are available in all frames. (This is the horizontal line of pads in Figure 1 consisting of: edit, help, back, next, mark, return, zog, display, user, comment, goto, find, and info.)

3. Anchor points. It is always possible to return to known frames that play the role of anchor points. Anchor points should be dynamically determinable. (This is realized in part by the pads: back, mark, return and zog.)

We have described the basic ideas that we are exploring. Now, let us examine the architecture of the system we constructed to explore these ideas.

3. System Architecture

In order to describe the architecture of the ZOG system, we must distinguish what a user sees from what a frame builder sees. It is important to note that any user may become a frame

builder; in fact, frame modification and augmentation are the primary means of allowing the user to tailor the system to his needs.

As described in the introduction, the user sees a display of a frame which contains some text and a menu of selections. When the user makes a selection, either by touching the screen or by typing the appropriate character, some action may occur and a new frame may be displayed. The user works his way through a network of frames organized into subsystems, called subnets. A subnet's goal is to guide the user to learn something, by reading the text, or to accomplish some action.

The frame builder sees the same basic system for selection processing, but also sees a communications language that allows him to construct frames and actions to accomplish desired tasks in the frames. The communications language that the frame builder uses has four basic functional capabilities: (1) network positioning to control which frames the user sees, (2) communications control to control interactions between the user and the various components of the system, (3) network and frame modification to allow existing frames to be modified and new frames to be added, and (4) hard-copy output of frames in various formats.

Next, we discuss the design of the mechanisms that support the frame builder's view of ZOG. Since the user's view is a subset of the frame builder's view, this covers those support mechanisms also. We will then discuss the current implementations of ZOG.

3.1. Basic System Design

The ZOG system may be viewed as a communications multiplexor. It has a number of logical input devices, each of which may be linked to an arbitrary subset of logical output devices. The basic function of ZOG is to cycle through the set of input devices and route messages to the appropriate output devices. Any one of these input devices may invoke the communications language, through escape characters, to control the position in the frame network, control the communications multiplexor, or manipulate frames.

The logical input and output devices are listed in Figure 3. The input devices are on the left and the output devices are on the right. Since an input can be routed to any set of outputs, no constraints can be placed on the inputs (i.e., although output devices will make

formatting assumptions, ZOG can make no such assumptions). For example, if the input file is routed to selection processing, then the characters being read from the file are treated just as selection characters typed from the keyboard. If the input file is routed to ZOGNet building, then the file being read is assumed to be in the external frame format (discussed in Section 3.1.2). If the input file is routed to the subjob, then it can be anything at all. It should also be noted that output devices are not aware of the source of the character streams with which they deal. We will briefly describe each of these logical devices before describing the rest of the system.

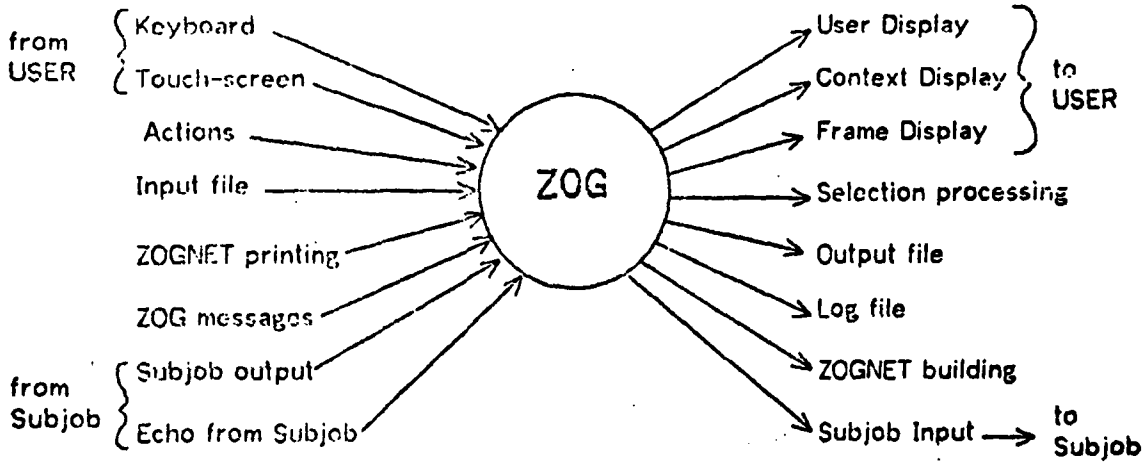


Figure 3: ZOG as a Communication Agent

Input devices generate character streams. The keyboard is read one character at a time. Each character is sent to all output devices linked to the keyboard, unless one of the communications language escape characters is detected. The touch sensitive screen produces a pair of coordinates when touched by a finger. These coordinates are translated into selection characters and then processed as though they had been entered from the keyboard. The communications language escape characters cannot be generated by the touch screen. The action input is processed during selection processing when a selection is made or when a frame is entered. The action is stored as part of the frame, and its format will be discussed

below. The input file (a file as defined and supported by the underlying operating system), once opened, is processed until the end of file is encountered. The subjob is a separate job being controlled by the ZOG job. The subjob may be used to run any arbitrary program, and input from the subjob is terminal output from the subjob's point of view. This job is automatically logged in when the first character is sent to it and logged out when ZOG exits. Subjob echo is produced when characters are sent to the subjob, if the subjob itself is echoing characters. Echo input cannot include communications language commands, even if it contains the appropriate escape characters. ZOGNet printing results from invoking one of the ZOGNet modification commands in the communications language. The printing is in an external frame format discussed below. Finally, ZOG messages result from prompts or errors reported by the ZOG system.

The output devices accept character streams. The selection processing logical output device is what the ZOG user sees most often. It takes a character representing a selection from a menu and does what is appropriate with it (discussed below). Three logical display windows are supported. The frame display is used by selection processing; the user display has no preassigned use (but would normally be used by a subjob); and the context display is a small window used to display the number of marked frames. Each window may include arbitrary areas of the screen. A single output file can be open, and any stream of text characters may be sent to it. The log file is opened if the user requests that a detailed log of his terminal interactions be maintained. It is also one of the general output devices so that the user may send other text to it. Output to the subjob appears as terminal input from the subjob's point of view. Finally, ZOGNet building constructs new frames or modifies existing frames from an external frame format.

When the user first starts interacting with ZOG, there is no file or subjob input, and the keyboard is linked to the selection processor. The initial frame is displayed and the user is expected to make some selection. The initial frame is ZOG1 by default, but the user may declare any frame to be the initial frame.

3.1.1. Frames and Subnets

Let us examine the content and layout of a frame, and the organization of the system into collections of frames, called subnets. From the user's point of view, a frame consists of some text and a menu of selections. The general format he sees is described in Section 2.

The internal description for a frame is shown in Figure 4. Each displayed item has positioning information so that the frame builder has maximum flexibility in defining his frame layout. The frame title is specified with its position (vertical, v, and horizontal, h, position on a 80 character by 24 line display) and the text to be displayed. The frame text is specified in the same way. The frame action is a character stream that will be fed to the communications multiplexor when the frame is entered. The option list contains a list of selections whose description will be given below. The local pad list contains a list of selections in the same form. The global pad frame number is the number of a frame whose options are displayed and accessed as global pads (common to a large part of the network). The accessor list is a list of frame numbers which reference this frame. The frame identification is the text string which appears in the upper right corner when the frame is displayed. It is constructed by concatenating the subnet name and relative frame number within the subnet. The maintenance information includes the subnet number, relative frame number, version number, name of owner, date of creation, name of last modifier, date and time of last modification, and an indication of whether the frame may be modified or viewed by users other than the owner. Finally, a comment may be stored with the frame. This comment is not displayed when the frame is displayed.

The description of a selection, either an option or a pad, is shown in Figure 5. It has position information and the text to display. The displayed text is expected to indicate to the user how to make the selection (e.g., by containing the selector character as the first character of text). The selection may have a next frame, which will be displayed when the selection is made. The next frame is represented as an absolute frame number to allow access to the next frame in a single disk access. The selection may have an action: a character stream that is fed to the communications multiplexor when the selection is made. A

```
-Frame title: v (line number), h (character position), text

-Frame text: v, h, text

-Frame action: text (not displayed)

-Option list: list of selections

-Local pad list: list of selections

-Global pad frame number

-Accessor list: list of frame numbers (not displayed)

-Frame id: text

-Maintenance information (not displayed)

-Comment: text (not displayed)
```

Figure 4: Internal Frame Description

touch rectangle is included for touch screen processing. The rectangle is specified by the coordinates of the upper left corner and the coordinates of the lower right corner. Finally, a selector character is specified. The selection processor uses the selector character (or the touch rectangle, if input is coming from the touch screen) when it tries to evaluate a user selection.

```
-Selection text: v, h, text

-Next frame: absolute frame number (not displayed)

-Selection action: text (not displayed)

-Selector: character (not displayed)

-Touch rectangle: X1, Y1, X2, Y2
```

Figure 5: Internal Selection Description

Actions are represented as simple text streams. A selection may have an action, and a frame may have an action on entry. The text for the action is sent to each logical output device linked to the action input device. Actions may invoke the communications language through appropriate escape characters. Thus, actions may use any ZOG supported facility through the communications language, and may execute arbitrary programs by interacting with the subjob. For example, a set of frames teaching ALGOL could have actions which actually run examples for the user. Because of the open-endedness of actions, they are a powerful means of programming frames.

A set of frames may be organized into a logical network called a subnet. Each subnet has a print name and an internal index (an integer). The number of subnets allowed is an implementation dependent parameter, but it is expected to be large (hundreds). The number of frames allowed in a subnet is also expected to be large, but small subnets are not penalized. Under normal circumstances, a user will enter a subnet through its root node (relative frame one). The root node will usually have a frame action which will set the necessary context for the subnet (e.g., start up an appropriate program on the subjob). Also, the root node frame action will normally mark the frame (described later) so that it can be returned to easily.

3.1.2. External Frame Format

An initial design objective was to maintain an external frame format that satisfied a number of desired properties. The format should be readable, allowing someone without a ZOG system to see what is in the frame library. The format should be easily supported by all ZOG systems (different versions and different machines), so that the frame library would be transportable. Finally, the format should be compatible with some existing facility to make its external maintenance and manipulation easy.

Figure 6 is a sample of the BH format defining the frame shown in Figure 1. It is moderately readable, is easily supported, and is compatible with the BH bibliography system. Although we have reached our objectives with this external frame format, it does fall short of providing total frame library transportability. Frames are transportable if their actions make

no use of the subjob. However, if an action evokes some program on the subjob, then that program must be transported as well as the frame. Since those programs are completely arbitrary, we have no control over their transportability.

```
+A+ ZOG3 3
+G+ GPads1
+T+ T "General background on ZOG"
+P+ T 1 10
+T+ F "ZOG is part of a research effort to understand communication between humans
and computers.  Various aspects of this research effort are described below."
+P+ F 3 1
+T+ 01 "1. System specifications of ZOG"
+P+ 01 6 3
+F+ 01 ZOG4
+T+ 02 "What is ZOG used for -- functional characterization"
+P+ 02 8 3
+F+ 02 ZOG5
+T+ 03 "3. Scientific issues behind ZOG"
+P+ 03 10 3
+F+ 03 ZOG6
+T+ 04 "4. Who is doing ZOG? Where? When? What sponsors?"
+P+ 04 12 3
+F+ 04 ZOG7
+T+ 05 "5. Prior research and antecedents"
+P+ 05 14 3
+F+ ZOG8
+T+ 06 "6. Examples of ZOG projects (real and in progress)"
+P+ 06 16 3
+F+ 06 ZOG9
+T+ 07 "7. Developing your own ZOG net"
+P+ 07 18 3
+F+ 07 ZOG10
+T+ LP "P-PRINT"
+X+ LP "^EF."
```

Figure 6: Example of BH Format

3.1.3. Selection Processing

Now that we have examined the internal and external structure of frames and subnets, let us take a closer look at the primary mechanism that the user deals with, the selection processor (see Figure 7). Let us start with a displayed frame. The user makes his selection by touching the screen or typing the selector character. In the case of the screen touch, some immediate feedback is given so that the user knows his touch was successful (a cross-hatch is placed on the screen where the touch was detected, and a star is placed over the first character of the selection if the touch fell within the bounds of one of the selection touch rectangles). Then the coordinates are translated into a selector character which is

treated as though it were typed. The selection processor then scans options, local pads, and finally global pads trying to match the selector character. If no match is found, it rings a bell. If a match is found, the selection is evaluated.

1. Get selector character.
2. Find selection. If none, ring bell and back to 1.
3. Interpret selection action, if any.
4. Set next frame, if any.
5. Display frame, if changed.
6. Interpret frame action, if any and if frame changed.
7. Back to 1.

Figure 7: Selection Processor Cycle

To evaluate a selection, the selection processor first checks if the selection has an action. If it does, then the character stream from the action is sent to each logical output device linked to the action input device. The selection processor handles escape characters to the communications language as well. After the action has been interpreted, the selection processor checks if the selection has specified a next frame. If so, it saves the current frame on the frame backup list and sets the specified next frame as current. Finally, the selection processor checks if the current frame is different from the last frame. If it is different, it is displayed and checked for any frame action. If the frame has an entry action, it is interpreted.

3.1.4. Communications Language

The communications language allows the user and the frame builder to maintain control over the interactions between the user and the various parts of the ZOG system. It can be invoked with escape characters from the keyboard, input file, subjob, or actions. It provides four basic facilities: (1) network positioning (escape character ^A (control-A)); (2) communications

control (escape character ^B); (3) frame modification (escape character ^D); and (4) hard-copy output (escape character ^E). This language is very simple: it has no variables, no conditionals, and no repeats. It is invoked by one of the four escape characters mentioned. The character following the escape character is a command. Some commands take operands by examining the character stream following the command.

The network positioning commands (^A) provide support for moving through the network in ways other than simple selection. Whenever a new frame is selected, the previous frame is saved on a backup list. The positioning commands allow the user to back up to the previous frame, mark a frame in the backup list, return to the last marked frame, and clear the backup list. A composite command backs up to the previous frame and takes the next option. An orienting command lets the user examine his backup stack, the list of frames he has marked, and the list of frames that point to the frame he is currently viewing. A find command allows the user to specify a search for an arbitrary string, with control over which parts of the frame to search and which frames to search, by restricting the search to frames created or modified by a particular person, or modified since a particular date and time. There are commands to control several defaults in the system: default initial frame, default global pads frame, default selection character for undefined selections, default time and selection character for display timeout (for controlling how long the user is allowed to view a frame). Finally, there are commands to re-display the current frame, go to an arbitrary frame, and control the statistics gathering package.

The communications control commands (^B) provide support for maintaining and modifying the routing control information for the logical input devices. These commands also support opening and closing the input and output files, manipulating the subjob, exiting from a ZOG system, and manipulating the display in a terminal independent way.

The frame modification commands (^D) provide a basis for an editing system. There are commands for sending individual frames, subnets, or the whole network. These may be used to save frames externally (e.g., if the ZOGNet printing input is routed to the output file) or to send the frames to a more elaborate editor running on the subjob. Frames may be constructed by routing the appropriate input device (usually the input file or subjob) to the

ZOGNet builder output device. There are commands for setting protection of frames (to allow a user to make a frame modifiable only by its creator, or viewable only by its creator). There are commands to delete a frame or a subnet. One command moves a frame from one part of the network to another. One command renames a subnet. Finally, one command enters ZED, the interactive edit mode for a frame.

The hard-copy output commands (^E) provide a means of getting hard-copy of frames in various formats. There are commands for getting a picture form of a frame (hard-copy same as displayed version), an index of frames in a subnet, and an index of all subnets.

3.1.5. Statistics Gathering

Since a principal concern with ZOG is understanding how users interact with such systems, ZOG is instrumented for gathering various kinds of data about its use. There are three kinds of statistics that ZOG deals with: static, dynamic, and log. The static statistics mechanism analyses ZOGNets. Figure 8 shows such an analysis for a typical subnet.

```
Status: 111 frames

368 options, 3.3 per frame
  per frame: 22 16 18 10 12 10 4 4 4 7    >9: 4
121 pads, 1.1 per frame
  per frame: 7 93 6 4 1 0 0 0 0 0    >9: 0

Actions: 0 frame, 13 option, 2 pad
Next frames: 95 option, 116 pad

Characters: 45736 TOTAL, 412.0 PER FRAME
  3814 title, 34.4 per frame
  16699 text, 150.4 per frame
  22916 option, 206.5 per frame, 62.3 per option
  2125 pad, 19.1 per frame, 17.6 per pad
  182 action, 1.6 per frame, 12.1 per action

Global pad frames: GPads1
```

Figure 8: Static Statistics for a Typical Subnet

Dynamic statistics are gathered while a user is interacting with the system. They record and summarize all user actions. Figure 9 shows an example of the dynamic statistics for a typical user session. These statistics are gathered and stored in a central statistics file when

the user exits from ZOG. The user may also request such a summary for his own perusal while in ZOG. There are external statistics packages that take the central statistics file and provide summary and search capabilities for it.

```
ZOG (7.6.2): MF21 using MNG.NET[X321ZG00] on 5/8/79 from 1802 to 1841.
1200 baud. 39 jobs.

84 frames accessed, 1423 secs, 16.9 sec/frame, SD = 20.83
  5 sec intervals: 4 27 26 9 5 5 1 0 3 2 0 0
  1 min intervals: 82 1 1 0 0      > 5 min: 0
14 frames edited, 960 secs, 68.5 sec/frame, SD = 86.54
  1 min intervals: 8 5 0 0 0 1 0 0 0 0      > 10 min: 0
98 frames total, 2383 secs, 24.3 sec/frame

BH files: 0 new frames, 0 replaced frames, 0 errors, 0 frames output
ZED: 1043 text chars, 102 command chars, 1 errors
Total: 1 frames created, 0 frames deleted

Selections: 49 options, 20 local pads, 30 global pads, 2 errors
Actions: 6 frame, 34 selection
Commands: 0 nexts, 15 backs, 2 returns, 3 gotos
          0 finds, 9 others, 1 errors
Chars: 1433 from keyboard, 0 from subjob, 0 to subjob

Subnets accessed: ZOG: 1   Mng: 7   Help: 3   Brostat: 24
Status: 21   Issue: 12   People: 9   Meeting: 1   Admin: 2   Update: 4
```

Figure 9: Dynamic Statistics for a Typical User Session

Log statistics are gathered only if a user specifically asks for them. They are much more detailed than the dynamic statistics shown above. The log file is a history of specific user input (keystrokes) and system response, with a time stamp (in milliseconds) for each action. Such a mechanism is essential for the study of user interaction with a ZOG-like system.

3.1.6. Frame Editing -- ZED

The frame editor, called ZED (for ZOG Editor), has two major parts: (1) a set of frames that allow the user to delete frames, move frames from one subnet to another, and create new subnets; and (2) an edit mode for altering a single frame.

The set of frames simply provides a convenient interface between the user and the communications language frame modification commands. Although the frames are not strictly part of the basic ZOG system, it is useful to think of ZED (frames and edit mode) as a single system.

There are two ways to enter the edit mode. The first is by selecting a selection that has no next frame. The user is allowed to create (define) a frame at that point. If the user decides to create that frame, he is prompted for a frame to copy (with a default for each subnet). The new frame is then displayed, and the edit mode is entered to allow the user to fill in whatever parts of the frame he desires. Once a frame is constructed, the user may select one of that frame's options which may point to a new undefined frame. By systematically stepping through the options and creating new frames at each step, the ZOCNet can be created in a top-down fashion.

The other way of entering the edit mode is through the edit global pad which appears on each frame. The design for edit mode was derived from extending the alter mode of SOS (a line oriented text editor found on many PDP10 systems, developed by Weiher and Savitzky (1970)) from one dimensional line editing to two dimensional frame editing. The frame being edited is always displayed. Edit mode commands are typed (but not echoed), and the results of an editing operation appear as changes to the displayed frame. Thus, visual fidelity is continuously maintained to the frame as modified. A description of these editing tools may be found in McCracken and Robertson (1979).

3.1.7. Conventions

We have adopted a number of conventions that make the system easier to use for both the user and the frame builder. They deal with the frame display organization, selection format, and commonly used pads. The system does not enforce these conventions in any way.

The organization of the displayed frame has only two system-enforced decisions. First, the frame identification is always displayed in the upper right corner. Second, the context display is always just to the left of the frame identification. The context display shows the number of marked frames (if any). The layout of the title, text, options, and pads is completely up to the frame builder.

Our current display format is shown in Figure 10. It assumes a display with 24 lines of 80 characters. The title is restricted to 56 characters on the first line. A blank line separates the title and text; the text is one to nineteen lines long with 80 characters per line. The

number of options is limited to nine so that digits may be used for selectors. Options share space with the text; that is, the less text, the more options. Each option is preceded by a blank line for better readability and ease of touch selection. The options may be up to 56 characters long. Space on the right is reserved for local pads, which may be 13 characters long. The most common pads (global pads) are on the bottom line. Options are selected by digits, local pads by upper case alphabetics, and global pads by lower case alphabetics. Any selection which is inert (has no action or next frame) is prefixed with a minus sign to indicate that it is not worth selecting. Finally, if a frame needs more than nine options, it is broken into several frames with the ninth option of each pointing to a frame with the continuation of the list.

```
[TITLE.....56.....] [MarkID.11] [FrameID.11]

[TEXT.....]
|      Text-lines + 2*Number-of-options + Workspace-lines = 19      |
[.....]

 1. [OPTION-TITLE.....56.....]    A-[PAD-TITLE.13]

 2. [OPTION-TITLE.....56.....]    B-[PAD-TITLE.13]

 3.-[OPTION-TITLE.....56.....]    C-[PAD-TITLE.13]

 . ..... .. .....

 8. [OPTION-TITLE.....56.....]    H-[PAD-TITLE.13]

 9. [OPTION-TITLE.....56.....]    I-[PAD-TITLE.13]

[USER DISPLAY.....]
edit  help  back next mark return  zog  display user  comment  goto find  info
```

Figure 10: Conventions for Display Layout

Another unenforced set of conventions has to do with the global pads. In Figure 10, a set of global pads is listed on the bottom line. The edit pad enters the edit mode of the frame editor, ZED. The help pad enters a subnet which describes how to use ZOG. The back pad returns to the previous frame. The next pad backs up to the previous frame and automatically selects the next option which has a next frame. The mark pad marks the current frame. The number of marked frames will appear in the context display (if there are any).

The return pad enters a subsystem that allows the user to examine his backup stack, list of marked frames, and list of frames which point to the current frame. From the return subsystem, the user may go directly to any frame on any of those lists. The zog pad goes to the root node of the whole system without disturbing context (i.e., the user may get back). The display pad re-displays the current frame. This is useful if the display has been disturbed by some external cause (e.g., a terminal failure or a message from another user). The user pad displays the user display (a full screen display whose bottom line is normally displayed in line 23 of the frame display). The comment pad provides a means for the user to send a comment to the system designers. The goto pad allows the user to go to any frame in the network. The find pad enters the find subsystem, which allows the user to search the network for an arbitrary string, for frames modified by a particular person, or for frames modified since a particular date and time. The info pad prints the maintenance information for the current frame, including the version number, the name of the creator/owner, date of creation, name of last modifier, and date and time of last modification. Exit from the system is done by ^C, which will log out the subjob if it was logged in, close the user's log file if it was opened, close any opened files, and record statistics about this user session in a central statistics file before exiting the system.

3.2. Current Implementation

There are three current implementations of ZOG; one for the DEC PDP10, one for the DEC VAX-11/780, and one for C.mmp, an experimental multiprocessor computer system. The three systems are basically compatible, although each has some specialized features and limitations not found in the others. In this section, we will briefly characterize the current state of each of these systems.

3.2.1. ZOG on the PDP10

The current implementation of ZOG on the PDP10 is written in L*(1), which is an interactive system-building system described in Newell, McCracken, and Robertson (1977). L* has made implementation and experimentation with ZOG easy and straightforward. The current

implementation follows the design described above except that it does not support rapid response. There are versions for TOPS-10, TOPS-20, and TENEX operating systems. This has been the basic system and has been operational in one form or another for over two years.

The current implementation supports 12 types of display terminals. It also supports a mode which may be used on any other kind of terminal (including a teletype), but with the display format only approximated. Support for other display terminals may be easily added, assuming the terminal in question has cursor addressing, screen clear, and backspace functions, and has at least 24 lines of 80 characters. Response rates depend on the data rate to the terminal and the system load. Most work done on ZOG to date has been with 1200 baud terminals on a moderately loaded PDP10 (KL10 with 40 to 50 users). Response rates under those conditions are typically around 5 seconds per frame, which is barely tolerable for ZOG work. The current implementation stores the frame network in a single disk file, and has a capacity of 225,000 frames (450 frames per subnet, with 500 subnets). We have recently interfaced one led/photocell touch screen (Carroll Manufacturing (1979)) to a standard video terminal with promising results.

3.2.2. ZOG on VAX-11/780

The current implementation of ZOG on VAX is written in L*VAX, a dialect of L*. It has just become operational, and is currently directly compatible with the PDP10 version. VAX ZOG runs under the UNIX operating system.

3.2.3. ZOG on C.mmp

The current implementation of ZOG on C.mmp is written in L*C.(D), a dialect of L*. This version does support rapid response on a touch screen graphics terminal, but does not support ZED (the frame editor) or subjob interactions. The C.mmp version has been primarily used as a vehicle for experimenting with effects of rapid response on user behavior.

C.mmp is a multi-mini-processor that was developed at CMU and is described in Wulf and Bell (1972). It has up to sixteen PDP11's connected through a crosspoint switch to a large shared memory. The operating system for C.mmp is called Hydra and is described in Wulf,

Pierson, Pollack, Levin, Corwin, Jones, and Cohen (1974). C.mmp/Hydra was chosen as the environment for the rapid response version of ZOG because the experimental nature of Hydra made it possible for us to modify the lowest levels of Hydra to help achieve our response goals.

The key to rapid response is the nature of the terminal being used. Conventional terminals are limited to relatively low bandwidths between processor and terminal. To get the kind of response we desired, it was necessary to use terminals that permit transfer rates of 50 kilobaud or higher. Our choice for this terminal was a graphics system developed at CMU, called the GDP (Graphics Display Processor) and described in Rosen (1974). The GDP is a vector drawing graphics system which can change the entire screen in less than 16 milliseconds. To allow full utilization of this speed, the GDP was interfaced directly to one of the PDP11's on C.mmp, with software support for it built into Hydra.

A ZOG terminal is a combination of a C.mmp GDP, a touch screen, and a high speed disk used as a frame cache. The touch screen is a clear plate of glass placed over the face of the GDP. When touched by a finger, it produces a pair of coordinates with a 0.1 inch accuracy (using a surface acoustic wave). The disk being used as a frame cache is an IMS disk (341 frame capacity), with a special feature that allows zero latency access to pages (4,096 16-bit words). The frame network itself is stored on an RP06 disk, with a capacity of 50,000 frames.

Rapid response is achieved by augmenting the interrupt service for the touch screen (part of Hydra, written in BLISS11) to handle most of selection processing. When a touch is made, an immediate response is given by overprinting the first character of the selection. This gives the user immediate feedback to confirm that his touch worked. The interrupt driven selection processor then finds the selection and evaluates it. If a selection action or frame action is encountered during evaluation, then the basic ZOG system is used at non-interrupt level (with actions being interpreted just as in the PDP10 version). However, for a selection with no action, the next frame is found (in primary memory, in the cache, or in secondary memory) and displayed at interrupt level. This design avoids most of the overheads normally imposed by an operating system. The current implementation supports a response rate of 0.1 second for selections without actions (generally about 70% of all selections).

The design and implementation of ZOG, plus the basic philosophy and goals, have now been described. Let us now look at some of our experience to date.

4. Initial Experience with ZOG

ZOG has been operational for about three years in the CMU computer science environment. During most of this time it has undergone development and enhancement, as befits any novel computing system. The local environment and system goals are sufficiently different from the single-task, highly structured medical environment of PROMIS to require such evolution. The initial stages of development were more a hill-climbing effort on obvious deficiencies and opportunities than a place for detailed, controlled psychological study. However, we provide some preliminary assessments and observations from this early phase below.

4.1. Computing Communities

Interactive computing systems require computing communities within which to grow and prosper. Such a view is widely held, for example, in connection with the growth of the software environment in the PDP10-ARPANET community. ZOG appears to be no exception. Until recently, there were two ZOG systems: the C.mmp version with rapid response, touch screen, but a single terminal; the PDP10 version with a slow (1200 baud) response, no touch screen, but available at all hours on all terminals in the environment. The community (PDP10) system became the main pacing system -- where the enhancements occurred and where almost all experience accumulated. That the C.mmp version was a substantially less comfortable software environment no doubt contributed to its infrequent use, but we believe it was availability that forced most use to be on the PDP10 system.

Though slow (see below), the community version attracted people interested in the new applications, permitted exploring applications that involved several users, and permitted the accumulation of experience by casual users.

4.2. Speed of Interaction

The initial view that rapidity of response is an essential feature of ZOG is clearly confirmed by the experience to date. The 5 second per frame response of the PDP10 version produces a qualitatively different interface than does the rapid (0.1 second) response, and intermediate speeds (4800 baud with about 2 second response) are also markedly different. In some applications (see below) the speed of response appears to have been critical in keeping the application from transforming into a routine useful facility.

In general use, the residence time at a frame is of the order of 10 to 20 seconds and appears to be remarkably constant over nets and users. With such times, highly rapid response (0.1 second) appears to offer no advantages over, say, 0.5 second system response. Longer delays may still be significant psychologically, since the user is waiting during the delay, whereas he is not waiting during the residence period. The purpose of the very fast response (0.1 second) is to make ZOG acceptable to expert users when traversing highly familiar paths. In various simple familiar tasks (such as searching a net repeatedly), the user response times on the rapid system never gets much below 1 second. Under these conditions the very fast response (compared to 0.5 second) does not seem critical. However, various system issues, such as the parallax of the touch screen, become limiting factors.

The effect of response speed is seen clearly in another feature of the PDP10 system. ZOG is an active network, intended to work in conjunction with other programs that provide substantial amounts of intelligent support. A key design decision in ZOG is implementing this not via an internal programming system but rather through the subjob communication link. This keeps ZOG simple and widens the range of programs which can be coupled with ZOG. However, the subjob facility in the TOPS10 operating system of the PDP10 is substantially slower than the regular interaction rate (1200 baud). In consequence, few applications have developed which depend on subjob support.

The result is that the PDP10 ZOG system is still located in the wrong place in the space of performance parameters to determine its usefulness and potential. The recent move to a VAX version of ZOG is an attempt to move to a better place in this parameter space.

4.3. Large Nets

We have verified what we knew: that it is extremely difficult to grow large ZOG networks. About 50 nets have been built by the total community and the total number of frames produced is about 10,000. The rate of frame production by hand, using ZED and top-down frame creation, is about 5 frames per hour. Like the typical residence time, this also seems to be remarkably constant over nets and over net builders. (Our data in both cases comes from the automatic gathering of statistics on all ZOG sessions throughout the community.) Typically, manual nets range from 100 to 300 frames; it takes dedication and project organization to get above 500. The really big nets, which account for the bulk of the 10,000, all have some structured or automatic growth features.

Thus, we have not yet produced the really large nets that seem an essential ingredient of ZOG. In part, this stems from the lack of machines aids for growing nets, and in part from the interaction of the current response speed with the computing community, which keeps ZOG as an interesting system for exploration but not effective enough to become self-sustaining in major applications which would drive up the size of the nets.

4.4. Applications

PROMIS offers good evidence that a large network, rapid response menu selection system is effective in the medical domain. However, PROMIS represents a highly engineered, dedicated application. Our purpose is to explore an open range of applications. Immersion of ZOG in a computing community has been our strategy for obtaining diverse applications.

We list here a number of the applications areas that have been explored, either by the ZOG group itself or others in the community. We then discuss several that yield interesting observations.

- Guidance systems: (1) Information about people, projects, systems, education, and other facets of life in the department for incoming students in the Carnegie-Mellon University Computer Science Department. (2) A transparent guide to the use of other programming systems (or even other computer systems if the right kind of local communications network exists). (3) An Academic

Advising System, used by the undergraduate student advisor to help answer students' questions about courses and majors, with information about course offerings and degree programs. (4) A front-end to Pople's (1977) Internist medical diagnosis system.

-Databases: Databases are the most natural application for ZOG-like systems. We have a number of database applications, including the experimental results in Cognitive Psychology, artificial intelligence systems, and a personal relational-database. In addition, we have a major project implementing a library browsing system (described in Fox and Palay (1979)).

-Project Management: Information about project goals, personnel, subtasks, reporting requirements, issues, and status. It is a highly dynamic network shared and updated by all project personnel.

-Issue Analysis: A policy analysis tool for handling large complex issues. The points and counterpoints of an issue are presented in a hierarchical and contrasting format called an issue net. The net is dynamic, and users carry on a debate using this shared medium. Two issue nets are described in Mantei and McCracken (1979).

-Interactive Documentation: Several complex systems have been documented with ZOG frame libraries, including two operating systems. Tools have been developed that aid the frame builder by translating machine-readable documents (in one of three document generation formats) into an initial set of frames. The initial set is inferior to hand-built frames, but provides a good starting point.

-Instruction: A traditional computer-aided instruction system that provides instruction in opening bridge bids. It provides alternate pathways through the instruction network based on student performance, by using a subjob to maintain state information.

The library browsing system, implemented by Fox and Palay (1979) and called BROWSE, is of special interest because of its size and the techniques used for growing it. The BROWSE net is currently 4000 frames and continues to grow. The net is grown by maintaining an external database of library entries (books, journals, articles) and a categorization of them.

An entry system is used by the librarians to augment and modify that external database, and a translation system converts it into a ZOGNet. The user of BROWSE then uses ZOG to browse through the library collection.

The project management ZOGNets are of special interest because of their dynamic nature. Such networks exist for several projects in the department, including the ZOG project itself. The ZOG project management net is about 550 frames, and is jointly maintained by everyone in the project (about 10 people). However, the relatively slow response rate (5 seconds) on the PDP10 has inhibited general use of ZOG management nets in our department. Except for the response problem, management nets have proven to be a valuable aid for modest sized projects.

The Internist front-end is of special interest because of its effective use of the subjob mechanisms. Earlier versions of Internist, described by Pople (1977), required the physician to enter symptoms using exact terminology, which was an undesirable memory load for the physician. The ZOG front-end allows the physician to select symptoms from a hierarchical set of menus. Internist is written in INTERLISP, and runs on a PDP10 under the TOPS-20 operating system. The ZOG front-end interacts with Internist through the ZOG subjob mechanism, and is the best example of the use of ZOG as an active communications agent.

4.5. Major Psychological Issues

In some broad sense, all problems with a (debugged) interface must be psychological. Certain aspects of performance with ZOG seem to raise issues that are strongly psychological and call for intensive psychological investigation.

Users readily get lost in using ZOG. The user does not know where he is, how to get where he wants to go, or what to do; he feels lost and may take excessively long to respond. This happens in all sorts of nets, especially complex nets or nets without regular structure. It does not happen in all, especially if the user does not have expectations about the net. The phenomena of getting lost is not unique to ZOG. There are many informal reports of similar problems in other interactive systems. It is probably the most important phenomenon to understand in psychological terms. Our first substantial psychological investigation on this

problem is being undertaken by Marilyn Mantei.

Users fail to read information on frames. Even though frames have only modest amounts of text and the users are looking for the information that is there in exactly the right form, they often fail to pick it up. Possibly the problem is in maintaining a state of acute attention to frame after frame. This phenomenon does not appear to be unique to ZOG (see Robertson (1977)).

The limited short term memory of the user is everywhere evident in ZOG. This is because the current display surface of 24x80 characters holds so little data that moving to a new frame becomes an act of total replacement. Some part of the phenomena of getting lost surely arises from this feature of ZOG. Like the question of speed, there may be a requirement that displays be large enough to hold not only the current frame, but a fair amount of knowledge accumulated from many frames plus various maps and overviews of where one is.

5. The Potential of ZOG

The most important question is whether ZOG has the potential to be a new communication interface, especially since menu selection techniques are common practice. The answer lies in rapid response into a large network (supported by the other principles), which produces a man-computer interface with qualitatively different properties, best summed up by the slogan of "transparency with speed".

The basic advantage of menu selection is that the user does not have to bring to the situation knowledge of either the functional possibilities for interaction or how to communicate. Either of these can be a strong barrier to communication. Menu selection instructs while operating. It is superior to a hard-copy manual in two ways. First, it eliminates the jump from barely acquired knowledge to how to put it into action. With a manual the user must still decide exactly what to do after reading the textual material; in menu selection he simply does it. Second, it eliminates the search for the relevant information by locating the relevant knowledge at the site of action. Manuals are not the only alternatives (e.g., there are instruction sheets, scripts and on-line help facilities); our purpose is only to make clear the essential dimensions of menu selection.

Menu selection has two disadvantages as normally implemented. First, it is slow. Expressed as channel capacity, in menu selection the user can select about 1 out of 10 alternatives every 5 seconds (i.e., about 1 bit per second). In typing, by contrast, the rate is about 1 word per second for a reasonable typist (i.e., about 10 bits per second). These figures are exceedingly rough; however, they illustrate that one can get messages into a computer about an order of magnitude faster by typing than by menu selection. The second disadvantage is the forced interposition of explanatory text and options. Menu selection becomes especially trying for skilled operators who no longer need the instruction that it offers. A manual can be put away after it is learned. This disadvantage compounds the speed disadvantage, since, even though the text can be ignored, there are usually extra options to wade through. However, the negative reactions probably are generated by more than just slowness, but also by a sense of needless bother and frustration about being forced to operate inefficiently.

These disadvantages do not prevent menu selection from being a useful technique. They do conspire to limit its use to applications where the user is a novice so that he needs explanation and could not communicate rapidly in any event. The relevant domain is broader than might be thought, ranging from rare situations where everyone is a novice, to systems used repetitively by experts who do not wish to acquire technical jargon.

Rapid response and the use of large networks are designed to remove both of these disadvantages. A response time of 0.5 seconds, as opposed to 5 seconds, gives back the missing factor of 10 in speed. Whether communication speed can be then competitive with typing is not determined simply by such raw figures, of course. But now, at least, the issue is a matter of details and not a foregone conclusion. Permitting avoidance of unneeded explanatory steps is basically a question of designing alternative sequences, so there are "short circuits" for experts and "long circuits" for novices.

Grant for the moment the major technical premise, that these two disadvantages can be effectively nullified. Then rapid selection, large network menu selection systems become a distinct type of man-machine interface. They maintain their properties of ease of use and transparency, but also permit both expert and novice operation, and in whatever the user's mixture of experience and familiarity with different aspects of the system. They become a

general purpose interface.

The evidence for the claim of a distinct type of interface comes both from the ZOG experience discussed above and the experience with PROMIS. This claim for ZOG is a minimal one. We will also lay out a much stronger claim, which might be taken as a maximal claim. In doing so, our intent is to describe the potentialities of ZOG. Though convinced of the potential of ZOG and desirous of exploiting it, we are strongly motivated by the desire to understand the interface scientifically, to discover its limits as well as its strengths. The maximal claim is that the ZOG type of interface is a preferred mode of man-machine interaction, even over the use of natural language dialog with the computer in the role of intelligent agent. To understand this claim, we must step back a little.

There are two polar views about how to structure man-machine interaction. One is the computer as tool. Under this view one wishes to make the computer a better tool -- more responsive, easier to wield, more reliable in application, capable of doing a bigger job at a stroke. Control remains with the user. The other view is the computer as intelligent assistant. In this view one wishes to make the computer more intelligent, and communication with it more natural. One does not wield an intelligent assistant, one tells it what one wants. Intelligent agents figure out what is necessary and do it themselves, without bothering you about it. They tell you the results and explain to you what you need to know.

There is a tension between these two views, for in an important sense intelligence is obscure. More precisely, intelligence in oneself is illuminating and transparent, but in others it is obscure and impenetrable. This follows from the tautological principle that to understand another's act of intelligence requires an act of intelligence. And precisely what an intelligent assistant is supposed to provide is freedom (of the user) from the effort of understanding. Put one other way, delegation requires an act of faith.

This trade-off-like opposition between computer as tool and computer as intelligent agent is a fairly deep affair, certainly capable of sustaining more analysis than we can devote to it here. We wish to use it only to make clear a claim about a ZOG-like interface. We make no assertions whether the tension is unresolvable or whether the computer might not permit combining these two views in many as yet unforeseen ways.

ZOG is an evolution in the tool direction. It seeks to produce a transparent device which, in itself, has no intelligence at all, but is immensely responsive to the user. It seeks to do this in the arena where we normally expect to use natural language, namely, dealing with large bodies of knowledge. Indeed ZOG uses natural language for its output. However, its own internal structure, which governs what it says and when it choses to say it, is completely open to examination by the user. In fact, examination takes place as a simple side effect of requesting the knowledge from the system.

The maximal claim then is that this mode of communication will be substantially superior to that of communication with an intelligent agent operating within the usual natural language dialog. The control exercised by the user and the ability to acquire the relevant pieces at a rate matching the users capabilities (no matter how fast) will more than offset the communication capabilities provided by intelligence applied to real time dialog.

This claim does not assert the unimportance of intelligence. A good network results only from intelligent analysis of the topic -- it is frozen intelligence. The claim refers to the process of communication, to the efficiency of knowledge acquisition or complex process control by a human user. Nor does this claim make an assertion about the scope of its application. Dynamic situations surely exist in which time to construct a network must be thrown in the balance against time to communicate by some more direct way from the naturally occurring situation.

What grounds are there for thinking the interface might contain the seeds of such an eventuality, rather than simply the minimal claim of being another (different and useful) interface? The unique property of ZOG-like interfaces is the rate of change of visual textual (and pictorial) information under user control. We can think of no other situation where this particular high degree of informational selectivity is evident. We can expect it to yield some quite new communicative phenomena. Other dynamically controlled visual situations (as in Sutherland's (1963) Sketchpad) operate at a lower interactive rate informationally (and are unique in other ways). The computer, as its speed and memory increases, opens a continual sequence of genuinely new interactive experiences. The rapid response large network interface is simply one of these.

6. Cost

We have discussed the functional potential of ZOG. It is also necessary to discuss the costs. ZOG (and PROMIS) are expensive systems in two ways: the cost of the hardware, and the cost of preparing the networks. The appropriate form to discuss costs is in terms of system demands for processing and memory, and manpower demands for networks. Reduction of these to dollars confounds the issue with particular technology and minor design decisions.

The technical demand of rapid response ranges from 50 to 300 kilobaud peak transmission, depending on implementation techniques. The unpredictability implied by the large network essentially means this rate must be available from large storage devices, though of course it can be shared for all terminals. Such a data rate is by no means out of the question, but it is like current disk-core transmission rates and is very high for terminals. The technical demand for the large networks is about 25 megabytes of storage (700 characters per frame times 35,000 frames). Again, this is not out of the question, but is substantial. However it is mostly shared among all terminals. Touch screen capability is itself not common on terminals and is still moderately expensive. But here demand could no doubt bring the cost under control. In summary, compared to currently popular interfaces the ZOG interface looks unattractive; only substantial conviction of its useful properties and/or a strong change in technological costs would lead to its exploitation.

The cost of developing the large network may be by far the largest barrier to the adoption of this philosophy. Under current art each frame must be hand crafted by a professional skilled in the knowledge embedded in the frames. The extra skills to be a good frame and net designer are not yet understood. PROMIS estimates a total of 100 man-years invested in producing its library of 35,000 frames. As mentioned earlier, our experience with ZOG indicates a production rate of about five frames per hour. This tremendous cost factor implies that work done on machine aided (or semi-automatic) frame generation will become critical to the adoption of this philosophy.

We are exploring three possible ways of reducing the cost of frame development with machine aided frame generation. The first, mentioned in connection with the interactive

documentation applications, is to convert existing machine-readable documents into an unpolished initial set of frames. The second is the use an externally maintained database with tools for automatic conversion to ZOG frames. This approach is being used by the library browsing application (see Fox and Palay (1979)). The third approach is to use collections of frames, called schemas, with parameterized display elements. The investigation of the schema approach is being done by Kamesh Ramakrishna. All three approachs show promise, although none appears to be universally useful.

7. Next Steps

From the general description we have given, it should be clear that there remain a number of unanswered questions about the parameters of ZOG-like interfaces. The main lines of further study and development are as follows.

The ultimate appeal of ZOG-like schemes is in the quality of interaction they offer, so that establishing the performance aspects and developing an appreciation for them will be the priority issues. Our recent implementation of ZOG on the VAX is a move in this direction. The VAX version appears to have significantly lower overheads on subjob interactions and faster response rates, which should encourage exploration of action oriented applications.

Development of techniques for constructing large networks is essential. As mentioned in the previous section, we are exploring three different approachs to machine aided frame generation. In addition, we are studying alternative mechanisms for global editing of frame libraries, which is further described in McCracken and Robertson (1979).

Studying how users interact with ZOG is essential, both in the classic human factors investigations of parameters (such as response time and display layout) and in more cognitive issues (such as search strategies, learning time and style of using ZOGNets). Our approach, originally described in Newell (1977) (see also Card, Moran and Newell (1980)), is to construct information processing models of user behavior. Some progress is already underway on this (see Mantei (1979) and her work on disorientation). We are particularly interested in developing new methodologies for using user study experiments to guide the development of systems (work being carried out with Kamila Robertson).

It is clear to us that ZOG and PROMIS represent a novel and potentially important kind of man-machine interface. Our experience with various applications supports that position. The areas of effort described above represent the current focus in our study of man-machine communications.

8. Acknowledgements

This work was supported by the Office of Naval Research under contract N00014-76-0874. It was also partially supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-78-C-1551.

This paper represents work done by a group of researchers including the authors and Kamesh Ramakrishna, Marilyn Mantei, Mark Fox, Andrew Palay, Kamila Robertson, Craig Somerville, Andrew Reiner, Rick Cattell, Robert Akscyn, Jeff Bonar, Lynne Reder, Lee Gregg, Joyce Hannah, Casey Quayle, and Harry Pople.

9. References

Card, S., Moran, T. and Newell, A. (1980). Computer Text Editing: An Information-processing Analysis of a Routine Cognitive Skill. Cognitive Psychology, Vol. 12, No. 1, 1980.

Carroll Manufacturing (1979). Touch Technology.

Chomsky, N. (1957). Syntactic Structures. The Hague: Mouton.

Ford, M. (1979). PRESTEL - The British Post Office Viewdata Service. Proceedings of the 1979 International Conference on Communications. IEEE, June 1979.

Fox, M. and Palay, A. (1979). The BROWSE System: An Introduction. Proceedings of the Annual Conference of the American Society of Information Science. Minneapolis. October 1979.

Mantei, M. (1979). Modeling User Behavior in Computer-Presented Learning Tasks. Presented at the American Education Research Association Annual Meeting. San Francisco. April 1979.

Mantei, M. and McCracken, D. (1979). Issue Analysis with ZOG, a Highly Interactive Man-Machine Interface. Proceedings of the First International Symposium on Policy Analysis and Information Science. Duke University. June 1979.

Martin, J. (1973). Design of Man-Computer Dialogues. Prentice-Hall.

McCracken, D. and Robertson, G. (1979). Editing Tools for ZOG, a Highly Interactive Man-machine Interface. Proceedings of the 1979 International Conference on Communications. IEEE, June 1979.

Newcomer, J. (1976). BH - A General Information Organization Program. Carnegie-Mellon University Technical Report, May 1976.

Newell, A. (1977). Notes for a Model of Human Performance in ZOG. Carnegie-Mellon University Technical Report, August 1977.

Newell, A., McCracken, D., and Robertson, G. (1977). L*: An Interactive, Symbolic Implementation System. Carnegie-Mellon University Technical Report, October 1977.

Newell, A., H. Simon, R. Hayes, and L. Gregg (1972). Report on a Workshop in New Techniques in Cognitive Research. Carnegie-Mellon University Technical Report, June 1972.

Newman, W. and Sproull, R. (1979). Principles of Interactive Computer Graphics. McGraw Hill.

Pople, H. (1977). The Formation of Composit Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. Proceedings of the Fifth IJCAI. Cambridge, Mass., August 1977.

Rosen, B. (1974). Graphics Display Processor Programmer Guide. Carnegie-Mellon University Technical Report, January 1974.

Robertson, C.K. (1977). The Division of Information Processing Labor between User and Support System--Exploration of a Concept. Ph.D. thesis, University of Pittsburgh, 1977.

Robertson, G. (1978). Some Design Considerations for the ZOG Man-Computer Interface. Proceedings of the Third NATO Advanced Study Institute on Information Science. Chania, Crete, Greece, August 1978.

Robertson, G., Newell, A. and Ramakrishna, K. (1977). ZOG: A Man-Machine Communication Philosophy. Carnegie-Mellon University Technical Report, August 1977.

Schultz, J. and Davis, L. (1979). The Technology of PROMIS. Proceedings of the IEE.

September 1979.

Sutherland, I. (1963). SKETCHPAD: A man-machine graphical communication system. Proc. AFIPS 1963 SJCC, Spartan.

Weiher, W., and S. Savitzky (1970). Son of Stopgap (SOS). Stanford Artificial Intelligence Laboratory Operating Note.

Wulf, W., and C. Bell (1972). C.mmp -- A Multi-Mini-Processor. Proc. AFIPS 1972 FJCC, Vol. 41, AFIPS Press, Montvale, N.J., pp. 765-777.

Wulf, W., C. Pierson, F. Pollack, R. Levin, W. Corwin, A. Jones, and E. Cohen (1974). Hydra: The Kernel of a Multiprocessor Operating System. CACM, Vol. 17, no. 6, June 1974, pp. 337-345.

Appendix I: ZOG Communications Language

Under normal circumstances, ZOG routes character streams from a number of sources of input to a number of logical output devices. Most of these input sources may invoke the communications language with one of six escape characters: ^A (control-A), ^B, ^C, ^D, ^E, or ^L. Two of these escape characters are direct commands: ^C forces an exit from the system, and ^L forces a clear of the user display. The remaining escape characters take the following character as a command, which may interpret characters following it as parameters. These escape characters provide support for the four basic facilities that ZOG provides: ZOGNET positioning, communications control, ZOGNET modification, and hard-copy output. After the command has been executed, the character stream continues to flow as determined by the new state. The following is a complete summary of features of the communications language (with the appropriate escape character shown before each command).

I.1. ZOGNET Positioning Commands

^A^A - Send ^A as though no escape occurred.

^AD - Display current frame. This command is useful if the display area becomes cluttered and the user wishes to redisplay the frame.

^AU - Show full user display. Normally, only the last line of the user display is shown (in line 23 of the frame display). This command allows the user to view the entire user display.

^AM - Mark current frame for later return.

^AR - Return to the last marked frame.

^A^ - Return anywhere. This command enters a subsystem that allows examination of the backup stack, the mark stack, and the list of frames pointing to the current frame. The user may go to any item on any of these lists.

^AB - Back up one frame.

^AN - Next option. This command goes to the previous frame and selects the option following the one last selected. It is useful while exploring an unfamiliar network.

^AG<frameid>; - Go to frame.

^AF - Find string. This command enters a subsystem that allows the user to specify and control a search of all or part of the ZOGNet. The search specification includes a string, the name of the last person who modified the frame, the date of last modification, whether to search text only or the whole frame, and what subnets to search. A list of frames matching the search specification is constructed, and the user is allowed to examine that list and go to any frame on it.

^AC - Clear backup list. Each time the user makes a selection, the frame he was at is saved in a backup list. This list is used by Backup, Mark, Next, and Return to preserve state. The

Clear command empties this list, with the side effect of removing any "marks".

^AP1 - Pop backup stack once.

^AP<frameid>; - Unwind backup stack to first occurrence of frame.

^A1<frameid>; - Set initial frame (used on entry to the system).

^A*<frameid>; - Set default global pad frame. If no global pad frame is referenced in a frame, the default is used.

^AHH - Hold statistics and log.

^AHR - Resume statistics and log. The hold and resume commands are used if detailed user data is being recorded and some interruption forces the user to leave the terminal for awhile.

^A. - Reset statistics counters.

^A? - Print dynamic statistics.

^AS<char> - Set selection character for undefined selections. Normally, the system simply rings a bell if the user types an undefined selection. With this command, any undefined selections will be translated into the specified character.

^AT<time>; - Set maximum display time in 100 millisecond units.

^AX<char> - Set display timeout selection character. The timeout time and timeout selection character commands enable an experimenter to control how long a user sees a display and what happens when the user does not respond within a given time period.

I.2. Communications Control Commands

^B^B - Send ^B as though no escape occurred.

^BXE or ^BXS - Set output level, E for expert, S for standard.

^B; - Comment to next carriage-return.

^BP<input> - Push input route. The current list of outputs to which the specified input are routed is saved. <input> is a single character: F for file, K for keyboard, T for touch-screen, A for action, Z for ZOGNET printing, M for ZOG messages, J for subjob, and E for echo from subjob.

^BU<input> - Pop input route. Restores the list of outputs for the specified input, assuming that a ^BP<input> was previously done for that input. Otherwise, it does nothing.

^BR<input><output1>...<outputn>; - Route input to outputs. This command establishes a list of outputs for a specified input. <outputm> is a single character: S for selection processing, Z for ZOGNET building, J for subjob, F for file, L for log file, U for user display, D for frame display, and C for context display (for number of "marked" frames).

^B? - Print routing information. This command will print a map of where each input is currently routed. It also prints information about opened files and the state of the subjob.

^BI<filenam.ext>; - Open input file. Once opened, an input file will be read to its end, with each character being sent to each output devices specified in the file input route. ^BI; will cause a prompt to be printed and the file name to be obtained from the keyboard.

^BO<filenam.ext>; - Open output file. Any output directed to the output file before it is opened will be ignored. ^BO; will cause a prompt to be printed and the file name to be obtained from the keyboard. If the file already exists, the user will be given a chance to abort the open.

^BC - Close output file.

^BL<filenam.ext>; - Open log file. In addition to opening the log file, this command starts the logging operation. Every keystroke and every frame display is logged with a time stamp.

^BM - Place subjob in monitor mode. This is equivalent to sending a series of ^C's to the subjob.

^BE - Exit from ZOG. If a subjob has been logged in, this will logout the subjob. If an output file has been opened, this will close it. ^C is trapped by ZOG and will cause a ^BE unless the keyboard is routed to the subjob, in which case it will be passed on to the subjob.

^BDh,v; - Position display cursor. This command takes a vertical line number, v, from 1 to 25, and a horizontal character position, h, from 1 to 80, and moves the cursor to that location. When ZOG is started, it asks the user what kind of terminal he is using. This allows actions and subjobs to be terminal independent.

^BT<char> - Set terminal type.

^B. - Clear display.

^BN - No clear on next frame display. This command is used to overlay frame displays to get special effects.

^BS<selection-char><char> - Print <char> immediately before selection specified by <selection-char>.

^BK<input><char> - Wait for character <char> from input source <input>. This command will suspend input from the current source until the condition is satisfied.

^BW<state> - Wait for state, where <state> is F for end of input file, J for subjob in monitor mode, and K for carriage-return from keyboard. This command will suspend input from the current source until the condition is satisfied.

1.3. ZOGNET Modification Commands

^D^D - Send ^D as though no escape occurred.

^DF<frameid>; - Send frame in BH form. This command sends the external form of the specified frame to outputs linked to the ZOGNET printing input.

^DC<frameid>; - Send frame comment in BH form.

^DS<subnetid>; - Send subnet. This command sends the BH form of all frames in the specified

subnet.

^DW - Send whole ZOGNET. This command sends all frames in BH form.

^D* - Send all frames modified during the current user session.

^DN<subnetid>; - Send next free frameid for subnet.

^D?<frameid>; - Send maintenance information for frame. The maintenance information includes the creation date, name of creator, date and time of last modification, and name of last modifier.

^D. - Send current frame identification.

^DU - Send current user identification.

^DV - Send current ZOG version number.

^DT - Send current log time (in milliseconds).

^DY<frameid>; - Display frame. This command displays the specified frame in the same way the selection processor displays frames.

^DO<frameid>; - Overlay display of frame.

^DD<frameid>; - Delete frame.

^DM<frameid>,<subnetid>; - Move frame to another subnet.

^DR - Rename a subnet. This command prompts for subnet to rename and new name to give it.

^DI<frameid>; - Initialize frame. This command will prompt for the name of a frame to copy, then enter ZED.

^DE<frameid>; - Edit frame (enter ZED).

^D - Resume edit. This command reenters ZED editing the frame it was last editing. ZED has several ways of suspending state and giving the user freedom to move about the ZOGNet, and this command restores that state.

^D- - Set indicator marks. This command scans the entire net and checks all selections to determine if they are inert or not. Inert selections are given a minus sign after the option number to indicate to the user that they go nowhere.

^DP<frameid>; - Set protection for frame. This command prompts for the type of protection required. Two types are currently supported: (1) only owner can write frame, and (2) only owner can view frame.

^DA - Set default protection for all new frames.

I.4. Hard Copy Output Commands

^E^E - Send ^E as though no escape occurred.

^EF<frameid>; - Send picture form of frame. Picture form is a hard copy image of what the user sees on his video display.

^ES<subnetid>; - Send picture form of all frames in subnet.

^E* - Send picture form of all frames modified during this user session.

^EI<subnetid>; - Send index for subnet. This command will list all frames that are defined in the specified subnet.

^EX - Send index of all subnets. This commands will list the names of all subnets.

^EA<subnetid>; - Send static analysis of subnet.

^E? - Send static analysis of all subnets.

Appendix II: ZED - The Frame Editor

The frame editor, called ZED, was based on an extension of the one dimensional SOS alter mode (Weiher and Savitzky, 1970) to a two dimensional frame alter mode. The frame is always displayed while it is edited. The user types single character commands (listed below), which are not echoed. The effect of the commands is to alter the frame as displayed. All alterations are made to a copy of the frame, so that the user may abort the edit with no changes made.

When alter mode is entered, a check is made to ensure that the frame is already defined. If it is not, alter mode aborts with an error message. If it is, a copy is made of it, and the copy is displayed. The cursor is set to the first character of the text, which becomes the current item being edited.

Most of the following commands may be preceded by a number (represented by italicized n), which indicates a repetition of the operation. In general, upper case commands are for manipulating items (title, text, options, local pads, global pads), and lower case commands are for manipulating text (within the current item).

General commands:

h or H or ? - Print alter mode help.
q or Q - Quit alter mode; no change to frame.
e or E - Exit alter mode with altered frame.

Item commands:

n<space> - Skip n characters in current item.
n<del> - Back up n characters in current item.
l - Back to start of current item.
*n*s<char> - Search for n'th occurrence of <char> in current item.
i<char>$ - Insert characters in current item to altmode.
*n*d - Delete n characters in current item.
^U - Restore current item.
*n*k<char> - Kill (delete) to n'th occurrence of <char> in current item.
*n*m<char><chars>$ - Munch (kill and insert).
t - Transpose next two characters in item.
*n*r<chars>$ - Replace n characters (same as ndi<chars>$).
*n*c<n-chars> - Change next n characters in item.
*n*v - Invert case of next n characters in item.
p - Alter position of current item.
n - Alter next frame of current item.
a - Alter action of current item.
b - Alter box (touch rectangle) for current item.
x<chars>$ - Extend item (insert at end).
z - Alter selection character of item.

Frame commands:

n<line-feed> - Skip n frame items.
n<altmode> - Backup up n frame items.
L - Back up to title (first frame item).
S<char> - Search for selection with <char> as selector.
I - Insert item; prompts for arguments.
D - Delete item.
_ - Replace whole frame; prompts for frame to copy.
@ - Exit to net with edit resume enabled.
A - Alter frame action.
B - Display boxes (touch rectangles).
C - Alter frame comment.
F - Format option positions.
G - Replace global frame pad.
J - Justify text.
M - Add mark command to frame action.
R - Replace item.
X - Extend options.

Appendix III: External Frame Format - BH Files

The external frame format was chosen to provide a means of transporting and exporting ZOG frames and nets to other ZOG systems. It was chosen to be compatible with a bibliography maintenance program, called BH (Newcomer, 1976), to allow use of the sorting features of BH if desired. A BH file contains a series of entries with each entry containing a number of elements. The first element must be a +A+ element. The other elements are all labelled with +<char>+. The following BH-compatible format is used for ZOG frames.

+A+ frameid v - Entry header. Define frame with version number v. If v is less than or equal to an existing version number, then bypass this old version. If v is zero, then augment the existing frame rather than replacing it.

+G+ frameid - Global pad frame.

+C+ "comment" - Comment. If a frame is being defined, the comment is included as part of the frame, although it is not normally displayed. Otherwise, the comment is ignored.

+F+ <desc> frameid - Next frame. This specifies the next frame for either an option or a pad. <desc> is described below.

+P+ <desc> v h - Position. This specifies the cursor starting position (vertical and horizontal) for an option or pad.

+T+ <desc> "text" - Text. This specifies the text which is displayed as part of an option or pad. It should include the selector character and some indication of whether making the selection would result in any action or next frame (i.e., if not, the selector character should be preceded by a "-").

+X+ <desc> "action" - Action. This specifies the character string which will be used as action input when the option or pad is selected.

+F+, +P+, +T+, and +X+ are used to characterize a selection (option or pad). Any of them may be omitted. <desc> specifies which option or pad is being characterized as follows: T is for frame title and if valid only for +P+ and +T+; F is for frame text for +P+ or +T+, frame action for +X+, and illegal for anything else; O<char> is for an option with a specified selection character; and L<char> is for a local pad.

Text appearing between double quotes may be multiple lines long. Several characters must be quoted within text strings (plus, ampersand, double-quote, and control-Q). The quote character is control-Q.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-CS-79-148	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE ZOG APPROACH TO MAN-MACHINE COMMUNICATION		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) G. Robertson, D. McCracken and A. Newell		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0874
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, Pa. 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA. 22217		12. REPORT DATE October 1979 13. NUMBER OF PAGES 54
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)