Proceedings of the International Conference, "Computational Systems for Health & Sustainability"
17-18, April, 2015 - by R.V.College of Engineering,
Bangalore, Karnataka, PIN-560059, INDIA

Eye Movement-Based Human-Computer Interaction Techniques

SARADA.T
SRI SAI RAM COLLEGE OF ENGINEERING,

MONISHA.S
SRI SAI RAM COLLEGE OF ENGINEERING,

Guided by:
SRINATH D.K,
Asst.Professor,
SRI SAIRAM COLLEGE OF ENGINEERING.

Abstract: User-computer dialogues are typically one-sided, with the bandwidth from computer to user far greater than that from user to computer. The movement of a user's eyes can provide a convenient, natural, and high-bandwidth source of additional user input, to help redress this imbalance. We therefore investigate the introduction of eye movements as a computer input medium. Our emphasis is on the study of interaction techniques that incorporate eye movements into the user-computer dialogue in a convenient and natural way. This chapter describes research at NRL on developing such interaction techniques and the broader issue raised by non-command-based interaction styles. It discusses some of the human factors and technical considerations that arise in trying to use eye movements as an input medium, describes our approach and the first eye movement-based interaction techniques that we have devised and implemented in our laboratory, reports our experiences and observations on them, and considers eye movement-based interaction as an exemplar of a new, more general class of non-command-based user computer interaction.

I. INTRODUCTION

In searching for better interfaces between users and their computers, an additional mode of communication between the two parties would be of great use. The problem of human computer interaction can be viewed as two powerful information processors (human and computer) attempting to communicate with each other via a narrow-bandwidth, highly constrained interface [25]. Faster, more natural, more convenient (and, particularly, more parallel, less sequential) means for users and computers to exchange information are needed to increase the useful bandwidth across that interface. On the user's side, the constraints are in the nature of the communication organs and abilities with which humans are endowed; on the computer side, the only constraint is the range of devices and interaction techniques that we can invent and their performance. Current technology has been stronger in the computer-to-user direction than user-to-computer; hence today's user-computer dialogues are typically one-sided, with the bandwidth from the computer to the user far greater than that from user to computer. We are especially interested in input media that can help redress this imbalance by obtaining data from the user conveniently and rapidly. We therefore investigate the possibility of using the movements of a user's eyes to provide a high-bandwidth source of additional user input. While the technology for measuring a user's visual line of gaze (where he or she is looking in space) and reporting it in real time has been improving, what is needed is appropriate interaction techniques that incorporate eye movements into the user-computer dialogue in a convenient and natural way. An interaction techniques a way of using a physical input device to perform a generic task in a human-computer dialogue [7].Because eye movements are so different from conventional computer inputs, our basic approach to designing interaction techniques has been, wherever possible, to obtain information from the natural movements of the user's eye while viewing the display, rather than requiring the user to make specific trained eye movements to actuate the system. We therefore begin by studying the characteristics of natural eye movements and then attempt to recognize corresponding patterns in the raw data obtainable from the eye tracker, convert them into tokens with higher-level meaning, and then build dialogues based on the known characteristics of eye movements. In addition, eye movement-based interaction techniques provide a useful exemplar of a new, non-command style of interaction. Some of the qualities that distinguish eye movement based interaction from more conventional types of interaction are shared by other newly emerging styles of human-computer interaction that can collectively be characterized as ''non-command-based.'' In a non-command-based dialogue, the user does not issue specific commands; instead, the computer passively observes the user and provides appropriate responses. On-command-based interfaces will also have a significant effect on user interface software because of their emphasis on continuous, parallel input streams and real-time timing constraints, in contrast to conventional single-thread dialogues based on discrete tokens. We describe the simple user interface management system and user interface description language incorporated into our system and the more general requirements of user interface software for highly interactive, non-command styles of interaction.

V. METHODS FOR MEASURING EYE MOVEMENTS

What to Measure

For human-computer dialogues, we wish to measure visual line of gaze, rather than simply the position of the eye in space or the relative motion of the eye within the head. Visual line of gaze is a line radiating forward in space from the eye; the user is looking at something along that line. To illustrate the

difference, suppose an eye-tracking instrument detected a small lateral motion of the pupil. It could mean either that the user's head moved in space (and his or her eye is still looking at nearly the same point) or that the eye rotated with respect to the head (causing a large change in where the eye is looking). We need to measure where the eye is pointing in space; not all eye tracking techniques do this. We do not normally measure how far out along the visual line of gaze the user is focusing (i.e., accommodation), but when viewing a two-dimensional surface like a computer console, it will be easy to deduce. Since both eyes generally point together, it is customary to track only one eye.

Electronic Methods

The simplest eye tracking technique is electronic recording, using electrodes placed on the skin around the eye to measure changes in the orientation of the potential difference that exists between the cornea and the retina. However, this method is more useful for measuring relative eye movements (i.e., AC electrode measurements) than absolute position (which requires DC measurements). It can cover a wide range of eye movements, but gives poor accuracy (particularly in absolute position). It is principally useful for diagnosing neurological problems revealed by eye movement patterns. Further details on this and the other eye tracking methods discussed here can be found in [27].

Mechanical Methods

Perhaps the least user-friendly approach uses a non-slipping contact lens ground to fit precisely over the corneal bulge. A slight suction is applied between the lens and the eye to hold it in place. The contact lens then has either a small mechanical lever, magnetic coil, or mirror attached for tracking. This method is extremely accurate, particularly for investigation of tiny eye movements, but practical only for laboratory studies. It is very awkward and uncomfortable, covers only a limited range, and interferes with blinking.

VI. PROBLEMS IN USING EYE MOVEMENTS IN A HUMAN-COMPUTER DIALOGUE

The most naive approach to using eye position as an input might be to use it as a direct substitute for a mouse: changes in the user's line of gaze would directly cause the mouse cursor to move. This turns out to be an unworkable (and annoying) design. There are two culprits for why direct substitution of an eye tracker for a mouse is not possible. The first is the eye itself, the jerky way it moves and the fact that it rarely sits still, even when its owner thinks he or she is looking steadily at a single object; the other is the instability of the available eye tracking hardware. There are significant differences between a manual input source like the mouse and eye position; some are advantages and some, disadvantages; they must all be considered in designing eye movement-based interaction techniques.

VII. EXPERIENCE WITH EYE MOVEMENTS

Configuration

As noted, we use an Applied Science Laboratories eye tracker in our laboratory. The user sits at a conventional (government-issue) desk, with a 16" Sun computer display, mouse, and keyboard, in a standard chair and office. The eye tracker camera/illuminator sits on the desk next to the monitor. Other than the illuminator box with its dim red glow, the overall setting is thus far just like that for an ordinary office computer user. In addition, the room lights are dimmed to keep the user's pupil from becoming too small. The eye tracker transmits the x and y coordinates for the user's visual line of gaze every 1/60 second, on a serial port, to a Sun4/260 computer. The Sun performs all further processing, filtering, fixation recognition, and some additional calibration. Software on the Sun parses the raw eye tracker data stream into tokens that represent events meaningful to the user-computer dialogue. Our user interface management system, closely modeled after that described in [12], multiplexes these tokens with other inputs (such as mouse and keyboard) and processes them to implement the user interfaces under study. The eye tracker is, strictly speaking, non-intrusive and does not touch the user in anyway. Our setting is almost identical to that for a user of a conventional office computer. Never the less, we find it is difficult to ignore the eye tracker. It is noisy; the dimmed room lighting is unusual; the dull red light, while not annoying, is a constant reminder of the equipment; and, most significantly, the action of the servo-controlled mirror, which results in the red light following the slightest motions of user's head gives one the eerie feeling of being watched. One further wrinkle is that the eye tracker is designed for use in experiments, where there is a ''subject'' whose eye is tracked and an ''experimenter'' who monitors and adjusts the equipment. Operation by a single user playing both roles simultaneously is somewhat awkward because, as soon as you look at the eye tracker control panel to make an adjustment, your eye is no longer pointed where it should be for tracking.

IX. USER INTERFACE MANAGEMENT SYSTEM

In order to make the eye tracker data more tractable for use as input to an interactive user interface, we turn the output of the recognition algorithm into a stream of tokens. We report tokens for eye events considered meaningful to the user-computer dialogue, analogous to the way that raw input from a keyboard (shift key went down, letter a key went down, etc.) is turned into meaningful events (one ASCII upper case A was typed). We report tokens for the start, continuation (every 50 ms., in case the dialogue is waiting to respond to a fixation of a certain duration), and end of each detected fixation. Each such token is tagged with the actual fixation duration to date,

so an interaction technique that expects a fixation of a particular length will not be skewed by delays in processing by the UIMS (user interface management system) or by the delay inherent in the fixation recognition algorithm. Between fixations, we periodically report a non-fixation token indicating where the eye is, although our current interaction techniques ignore this token in preference to the fixation tokens, which are more filtered. A token is also reported whenever the eye tracker fails to determine eye position for200 ms. and again when it resumes tracking. In addition, tokens are generated whenever a new fixation enters or exits a monitored region, just as is done for the mouse. Note that jitter during a single fixation will never cause such an enter or exit token, though, since the nominal position of a fixation is determined at the start of a fixation and never changes during the fixation. These tokens, having been processed by the algorithms described above, are suitable for use in a user-computer dialogue in the same way as tokens generated by mouse or keyboard events. We then multiplex the eye tokens into the same stream with those generated by the mouse and keyboard and present the overall token stream as input to our user interface management system [12]. The desired user interface is specified to the UIMS as a collection of relatively simple individual dialogues, represented by separate interaction objects, which comprise the user interface description language (UIDL). They are connected by an executive that activates and suspends them with retained state, like coroutines. A typical object might be a screen button, scroll bar, text field, or eye-selectable graphic object. Since, at the level of individual objects, each such object conducts only a single-thread dialogue, with all inputs serialized and with a remembered state whenever the individual dialogue is interrupted by that of another interaction object, the operation of each interaction object is conveniently specified as a simple single-thread state transition diagram that accepts the tokens as input. Each object can accept any combination of eye, mouse, and keyboard tokens, as specified in its own syntax diagram, and provides a standard method that the executive can call to offer it an input token and traverse its diagram. Each interaction object is also capable of redrawing itself upon command (it contains the needed state information or else contains calls to the access functions that will obtain such from its domain object). An interaction object can have different screen extents for purposes of re-drawing, accepting mouse tokens, and accepting eye tokens. A standard executive is then defined for the outer dialogue loop. It operates by collecting all of the state diagrams of the interaction objects and executing them as a collection of coroutines, assigning input tokens to them and arbitrating among them as they proceed. Whenever the currently-active dialogue receives a token it cannot accept, the executive causes it to relinquish control by co routine call to whatever dialogue can, given its current state, accept it. If none can, executive discards the token and proceeds. For example, each ship (see Figure 6) is a separate interaction object (but all are of the same class, **Ship**). An additional lower-level interaction object (**Gazer**) is provided to perform the translation of fixations into gazes, as described above. That is, every interaction object such as **Ship** also has a **Gazer** interaction object associated with it. The **Gazer** accepts fixations on (or near, according to the criteria described above, and by means of a class variable shared by all the active **Gazer**s) its parent object and then combines such consecutive fixations into a single gaze token, which it sends to its parent object (the **Ship**). Figure 7shows the syntax diagram for **Gazer**; it accepts the tokens generated by the fixation recognition algorithm (**EYEFIXSTART**, **EYEFIXCONT**, and **EYEFIXEND**), tests whether they lie within its extent or else meet the criteria for off-target fixations described above (implemented in the call to Is Mine), accumulates them into gazes, and sends gaze tokens (**EYEGAZESTART**, **EYEGAZECONT**, and **EYEGAZEEND**) directly to its parent object. The **Ship** interaction object syntax then need only accept and respond to the gaze tokens sent by its **Gazer**. Figure 7 also shows the portion of the **Ship** interaction object syntax diagram concerned with selecting a ship by looking at it for a given dwell time (for clarity the syntax for dragging and other operations described below is not shown in the figure; also not shown are the tokens that the selected ship sends to the other ships to deselect the previously-selected ship, if any). When a user operation upon a ship causes a semantic-level consequence (e.g., moving a ship changes the track data), the **Ship** interaction object calls its parent, an application domain object, to do the work. Although the syntax may seem complicated as described here, it is well matched to the natural saccades and fixations of the eye.

X. INTERACTION TECHNIQUES

Interaction techniques provide a useful focus for this type of research because they are specific, yet not bound to a single application. An interaction technique represents an abstraction of some common class of interactive task, for example, choosing one of several objects shown on a display screen. Research in this area studies the primitive elements of human computer dialogues, which apply across a wide variety of individual applications. The goal is to add new, high-bandwidth methods to the available store of input/output devices, interaction techniques, and generic dialogue components. Mockups of such techniques are then studied by measuring their properties, and attempts are made to determine their composition rules. This section describes the first few eye movement-based interaction techniques that we have implemented and our initial observations from using them.

Moving an Object

Another important interaction technique, particularly for direct manipulation systems, is moving an object on the display. We experimented with two methods. Our initial notion was that, in a direct manipulation system, a mouse is typically used for two distinct operations–selecting an object to be manipulated and performing the manipulation. The two functions could be separated and each assigned to an appropriate input device. In particular, the selection could be performed by eye position, and the hand input device devoted exclusively to the manipulations. We therefore implemented a technique whereby the eye selects an object (ship) to be manipulated (moved on the map, in this case) and then the mouse is used to move it. The eye selection is made precisely as in the previously-described interaction techniques. Then, the user grabs the mouse, presses a button, drags the mouse in the direction the object is to be moved, and releases the button. There is no visible mouse cursor in this scheme, and the mouse is used as a relative position device–it starts moving from wherever the eye-selected ship was. Our second approach used the eye to select and drag the ship, and a pushbutton to pick it up and put it down. The user selects a ship, and then presses a button; while the button is depressed, the ship drags along with the user's eye. (Since the processing described previously is performed on the eye movements, the ship actually jumps to each fixation after about 100 ms. and then remains steadily there–despite actual eye jitter–until the next fixation.) When the button is released, the ship is left in its new position. Our initial guess was that the second method would be too unpredictable; eye movements would be fine for selecting an object, but picking it up and having it jump around on the screen in response to eye movements would be annoying–a mouse would give more concrete control. Once again, our initial guess was not borne out. While the eye-to-select/mouse-todrag method worked well, the user was quickly spoiled by the eye-only method. Once you begin to expect the system to know where you are looking, the mouse-to-drag operation seems awkward and slow. After looking at the desired ship and pressing the ''pick up'' button, the natural thing to do is to look at where you are planning to move the ship. At this point, you feel, ''I'm looking right at the destination I want, why do I now have to go get the mouse to drag the ship over here?'' With eye movements processed to suppress jitter and respond only to recognized fixations, the motion of the dragging ship is reasonably smooth and predictable and yet appears subjectively instantaneous. It works best when the destination of the move is a recognizable feature on the screen (another ship, or a harbor on a map); when the destination is an arbitrary blank spot, it is more difficult to make your eye look at it, as the eye is always drawn to features.

Eye-controlled Scrolling Text

A window of text is shown, but not all of the material to be displayed can fit. As shown at the bottom left of Figure 8, a row of arrows appears below the last line of the text and above the first line, indicating that there is additional material not shown. If the user looks at the arrows, the text itself starts to scroll. Note, though, that it never scrolls when the user is actually reading the text (rather than looking at the arrows). The assumption is that, as soon as the text starts scrolling, the user's eye will be drawn to the moving display and away from the arrows, which will stop the scrolling. The user can thus read down to end of the window, then, after he or she finishes reading the last line, look slightly below it, at the arrows, in order to retrieve the next part of the text. The arrows are visible above and/or below text display only when there is additional scrollable material in that direction.

Listener Window

In a window system, the user must designate the active or ''listener'' window, that is, the one that receives keyboard inputs. Current systems use an explicit mouse command to designate the active window; in some, the command is simply pointing, in others, it is pointing and clicking. Instead, we use eye position–the listener window is simply the one the user is looking at. A delay is built into the system, so that user can look briefly at other windows without changing the listener window designation. Fine cursor motions within a window are still handled with the mouse, which gives an appropriate partition of tasks between eye tracker and mouse, analogous to that between speech and mouse used by Schmandt [22]. A possible extension to this approach is for each window to remember the location of the mouse cursor within it when the user last left that window. When the window is reactivated (by looking at it), the mouse cursor is restored to that remembered position. This method works well in practice because the resolution required for selecting a window on a typical display is fairly coarse. It is convenient and fast, but the difficulty of using the eye tracker in a natural setting and posture for an extended time has made it difficult to frame a good comparison to a conventional window manager in everyday use.

XI. TOWARDS AND BEYOND NON-COMMAND INTERFACES

Finally, we return to the broader question of the nature of future human-computer interaction styles. Eye movement-based interaction provides an example of several of the characteristics–as well as the problems–of what seems to be an emerging new user-computer interaction style. The new style, which combines the non command attribute with other somewhat correlated characteristics, is seen most dramatically in virtual reality interfaces, but its characteristics are common to a more general class of rich user-computer environments,

such as new types of games, musical accompaniment systems, interactive entertainment media, as well as eye movement-based interfaces. They all share a higher degree of interactivity than previous interfaces–continuous input/output exchanges occurring in parallel, rather than one single thread dialogue. Most also go a step further away from the traditional dialogue toward a more subtle, implicit interchange based on passive monitoring of the user's actions rather than explicit commands. The concepts behind these emerging new interface styles can be better understood by decomposing them into two main attributes, suggested in Figure 11. The non command-based quality, which was described earlier, is just one of the two.

XII. CONCLUSIONS

Following Brooks' taxonomy [3], we present ''observations,'' rather than more formal ''findings'' of our experiences with eye movement-based interaction:

• An eye tracker as an input device is far from ''perfect,'' in the sense that a mouse or keyboard is, and that is caused both by the limitations of current equipment and, more importantly, by the nature of human eye movements. Obtainable accuracy is more similar to a traditional touch screen than a mouse, and the range can barely cover a single CRT display. The equipment, while non-intrusive and non contacting, is difficult to ignore. Nevertheless, it is perhaps amazing that eye movement-based interaction can be done at all; when the system is working well, it can give the powerful impression of responding to its user's intentions rather than his or her explicit inputs.

• To achieve this, our overall approach in designing interaction techniques is, wherever possible, to obtain information from a user's natural eye movements while viewing the screen rather than requiring the user to make specific eye movements to actuate the system. For example, we tried and rejected long gazes because they are not natural eye movements, preferring to use gazes only as long as natural fixations. We also found it important to search for and recognize fixations in the raw eye tracker data stream and construct our dialogue around these higher-level events.

• Our research concentrated on interaction techniques that can naturally and conveniently incorporate eye movements into a user-computer dialogue, rather than on the underlying eye tracker technology itself. In our initial interaction techniques, we observed the value of short dwell time eye-only object selection for cases where a wrong pick immediately followed by a correct pick is acceptable. For moving an object we found filtered eye movements surprisingly effective, even though a mouse initially seemed more appropriate for this task. For menu commands, we found the eye alone appropriate for popping up a menu or tentatively choosing an item, but executing an item requires a button for confirmation rather than a long dwell time. The next step in this research is to perform more controlled observations on the new techniques. Our first experiment will compare object selection by dwell time with conventional selection by mouse pick. The extraneous details of the ship display are removed for this purpose, and a simple abstract display of circular targets is used, as shown in Figure 10. In the experiment, one of the targets will be designated, and the subject's task is to find it and select it, either by eye with dwell time or mouse. Response time for the two methods will be compared. (Initial pilot runs of this procedure suggest a 30 per cent decrease in time for the eye over the mouse, although the eye trials show more variability.)

Finally, we can view eye movement-based interaction as an instance of an emerging new style of user-computer interaction. This style exhibits changes from explicit to implicit commands and from turn-taking, single-stream dialogues to simultaneous, parallel interactions.

Such interfaces will levy new requirements on user interface software and on the languages used to describe user-computer dialogues to handle and describe complex and substantial input/output processing, simultaneous parallel inputs, continuous inputs and outputs, imprecise inputs, and real-time constraints.

XIV. REFERENCES

[1]. R.A. Bolt, ''Gaze-Orchestrated Dynamic Windows, 'Computer Graphics **15**(3) pp.109-119 (August 1981).

[2]. R.A. Bolt, ''Eyes at the Interface,'' Proc. ACM Human Factors in Computer Systems Conference pp. 360-362 (1982).

[3]. F.P. Brooks, ''Grasping Reality Through Illusion– Interactive Graphics Serving Science, ''Proc. ACM CHI'88 Human Factors in Computing Systems Conference pp. 1-11, Addison-Wesley/ACM Press (1988).

[4]. S.K. Card, W.K. English, and B.J. Burr, ''Evaluation of Mouse, Rate-controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT,'' Ergonomics **21**(8) pp.601-613 (1978).

[5]. H.D. Crane and C.M. Steele, ''Generation-V Dual-Purkinje-image Eye tracker,'' Applied Optics **24**(4) pp. 527-537 (1985).

[6]. S.K. Feiner and C.M. Beshers, ''Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds,'' Proc. ACM UIST'90 Symposium on User Interface Software and Technology pp. 76-83, Addison-Wesley/ACM Press, Snowbird, Utah (1990).

[7]. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, Computer Graphics: Principles and Practice, Addison-Wesley, Reading, Mass. (1990).

[8]. F.A. Glenn and others, ''Eye-voice-controlled Interface,'' Proc. 30th Annual Meeting of the Human Factors Society pp. 322-326, Santa Monica, Calif. (1986).

[9]. M. Green and R.J.K. Jacob, ''Software Architectures and Metaphors for Non-WIMP User Interfaces,'' Computer Graphics **25**(3) pp. 229-235 (July 1991).

[10]. R.N. Haber and M. Hershenson, the Psychology of Visual Perception, Holt, Rinehart and Winston, New York (1973).

