

視線を利用したウィンドウ操作環境

大野 健彦

NTT コミュニケーション科学基礎研究所

〒 243-0198 神奈川県厚木市森の里若宮 3-1

Tel: (046)240-3662 Fax: (046)240-4707

E-mail: takehiko@rudolph.brl.ntt.co.jp

本研究では、視線を利用してウィンドウを操作する環境 Ewm を提案する。これまでマルチウィンドウシステムにおいてウィンドウの操作にはマウスなどのポインティングデバイスが一般的に利用されてきた。そのため操作のたびにポインティングデバイスを操作する必要があり、また操作をおこなうための特別な知識を必要とした。我々の提案するシステムはユーザの視線を観察することでウィンドウの操作を自動的におこない、またユーザがウィンドウを操作する場合はコマンドを視線によって実行することが可能であるという特徴を持つことで、手を用いることなくわずかな知識で操作可能なウィンドウ操作環境を実現している。

視線インタフェース, マルチウィンドウ, ウィンドウマネージャ, GUI

Eye Mark Window Management Environment

Takehiko Ohno

NTT Communication Science Laboratories

3-1 Morinosato Wakamiya, Atsugi, Kanagawa, 243-0198 JAPAN

Tel: (046)240-3662 Fax: (046)240-4707

E-mail: takehiko@rudolph.brl.ntt.co.jp

In this work, an eye gaze controlled window management environment is proposed. In traditional multi-window systems, control is facilitated by pointing devices like the mouse, and task specific knowledge is necessary. Our window management system recognizes users' eye movement to handle the windows automatically. It is also possible for users to select a command from the control menu by eye mark. Hence, users can control windows without hand movement and with little task knowledge.

Eye gaze interface, multi-window, window manager, GUI

1 はじめに

近年，コンピュータの処理速度が向上すると共に，コンピュータを操作するためのインタフェースの形態は CUI (Character User Interface) から GUI (Graphical User Interface) へ変化してきた．GUI によって実現されたシステムの一つに，複数のウインドウを同時に表示することの可能なマルチウインドウシステムが挙げられる．マルチウインドウシステムによってユーザは複数アプリケーションを複数のウインドウにおいて並列に利用することが可能となった．

マルチウインドウシステムは作業の利便性を大きく向上させたが，一方で操作が複雑になるという問題も発生した．例えば複数のアプリケーションが別のウインドウにおいて動作している場合，利用するアプリケーションを切替えるにはそのアプリケーションが動作しているウインドウをマウス¹によって選択する必要がある．また，多数のウインドウを表示するとウインドウが他のウインドウによって隠れてしまい，必要なウインドウを見ることができない場合がある．この時，ユーザはマウスでウインドウの位置を移動させたり表示順序を変更させる必要がある．表示するディスプレイを大きくすることで問題の一部は解消されるが，ディスプレイの大きさには限界があり，また大きすぎるとかえって使いにくくなるため根本的な解決には至っていない．

マルチウインドウの操作は本来アプリケーションを利用する上では不要な作業であり，初心者がコンピュータを習得する上では障害となり，また熟練者にとっては複雑な作業である．特にキーボードを利用する作業において，ウインドウを操作するために手をマウスへ移動させるのは作業時間の浪費となる．作業効率を高めたいユーザに対してはキーボードショートカットが用意されている場合もあるが，キーボードショートカットの種類を習得する必要があるために，一般的に広く利用されているとは言い難い (例えば Macintosh には殆んど用意されておらず，原則としてウインドウの操作はマウスでおこなう．一方，Windows では多数のキーボードショートカットが用意されており，マウスを利用せずに殆どどのウインドウ操作をおこなうことができる．また，X-Window ではウインドウマネージャによって異なるが，ユーザが自由に定義できる場合が多い) ．

本研究で提案するウインドウ操作環境はマルチウイ

¹ウインドウを操作するポインティングデバイスにはマウスの他にトラックボール，タッチパネルなど多種あるが，本論文では一般的に広く利用されているデバイスとしてマウスと記す．

ンドウシステムとして広く利用されている X-Window 上で構築されており，ユーザの視線を利用することでウインドウ操作の自動化を実現している．また，ユーザは視線によってウインドウ操作のコマンドを明示的に選択することも可能であり，ポインティングデバイスを利用することなくウインドウを操作することができる．

本環境は以下に述べる目標を持つ．

- 視線を利用した，十分に使いやすいシステムとする．
- ユーザがウインドウ操作を意識することなく作業をおこなうことができる．
- 初心者でも簡単に利用できるように，覚えるべき知識は最小限とする．
- 従来通りマウスによる操作も可能とし，操作性が低下することはない．

これらの目標を達成するために，我々はまず視線を利用したインタフェース (視線インタフェース) の問題点を検討し，その結果に基づいたシステムの構築をおこなった．

本論文では 2 節で視線インタフェースが持つ問題点を述べ，3 節では X-Window におけるウインドウ操作の概要を説明する．4 節では我々の試作システム Ewm の機能およびその構成法を述べ，5 節でシステムの使いやすさに関する議論をおこなう．最後に 6 節でまとめおよび今後の課題について述べる．

2 視線による人の作業支援

ユーザがコンピュータを操作する場面において視線を作業の支援に利用する視線インタフェースは，ユーザの視線を記録するアイカメラ (視線記録装置) の高性能化，低価格化に伴って様々なシステムが提案されてきた．その形態は次の 3 種類に分類される．

もっとも一般的な形態は視線を利用してボタン，アイコン，メニューなどの操作をおこなうシステムである [3, 5, 6, 7] ．ポインティングデバイスを利用してメニューを選択する場合，視線は選択目標となるメニュー上に位置する．そこで視線を直接選択に利用することで，選択時間の高速化，操作方法の簡略化が可能となる．メニューの選択においては，メニュー上に視線が一定時間以上位置した場合を選択と判定していたが [3, 5]，視線選択領域を別個に設け，その領域内に

視線が位置した場合のみ選択がおこなわれるという方法で高速な選択を実現している例もある [6, 7] .

2 番目は他のポインティングデバイスと視線を組み合わせる形態である . 例えばマウスと視線を組み合わせ , マウスポインタが移動した時にユーザの視線を測定し , その位置にマウスポインタを移動させるシステム Magic Point がある [11] . Magic Point は視線を移動位置の最終的な決定には使わないことで , 視線によって誤って選択してしまうという問題を回避している .

3 番目はユーザが視線を利用してコンピュータを操作するのではなく , 作業中の視線をシステムが利用して支援をおこなう形態である . 例えば人が画面を眺めている時の視線を記録して自動的にスクロールをおこなうシステムでは , 文章のブラウジングにおいてキーボードより効率の良い結果を得ている [9] . また , 図形配置時の視線を利用して , 整列に利用する方法も提案されており [8] , ユーザの作業を観察した結果 , その有効性が示唆されている .

これらは視線を利用して人の作業効率を向上させることを目標としているが , 実際には様々な問題が存在するために必ずしもユーザの意図通りに動作せず , 逆に操作効率が低下する場合もある .

2.1 視線インタフェースの問題点

視線インタフェースは視線の持つ特徴に起因する特有の問題を持ち , 使いやすいシステムを実現するためにはこれらを解決していく必要がある .

誤作動について

視線インタフェースは , 他のインタフェースに対してエラーの発生率が高いという問題点がある [7] . エラー発生の原因には , 大きく分けて 2 種類が考えられる . 1 つはアイカメラの測定精度が低いことである . 一般に現在のアイカメラはユーザによって測定精度が異なり , また使用中に精度が低下しやすい . 測定精度が低い場合 , 測定結果とユーザが実際に見てる場所が異なり , 誤ったコマンドを選択したりコマンドを選択しようとしても選択されない場合がある . この問題はアイカメラの性能が向上するにつれて徐々に解決されるであろう .

もう 1 種類は視線の持つ役割の多様性である . 視線インタフェースではユーザの視線を入力情報として利用するが , ユーザは画面上の情報を取得するために , 様々な箇所を短時間の間に見る . その結果 , 必ずしも

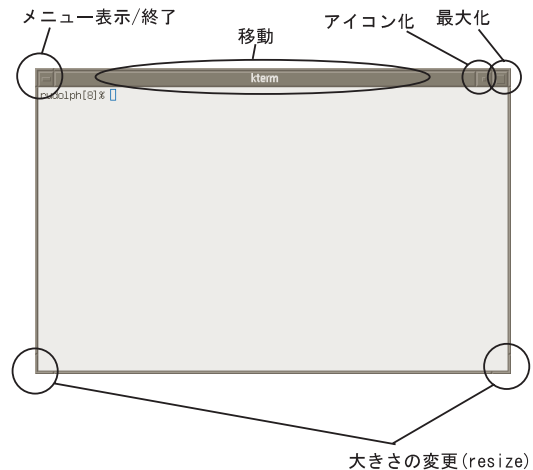


図 1: ウィンドウマネージャの一例 (Fvwm 2.2)

設計者が想定した視線運動とはならない場合がある . 特にユーザがある箇所を見たときに対応するコマンド等が選択される形態のインタフェースでは , ユーザがコマンドを選択したいと思わない場合でも選択してしまう場合がある . この現象は視線を利用したインタフェースの大きな問題となっている [4] .

視線の制約

視線インタフェースによっては , ユーザが通常画面を見ている場合と異なる視線の動かし方を必要とする場合がある . 例えば特定の箇所を見た場合に何らかの変化が発生するシステムでは , ユーザがその動作を望まない場合はその箇所を見てはいけなくなる . このような制約はユーザにとって極めて不便であり , インタフェースを構築する上ではこのような制約が発生しないようにする必要がある .

本研究では視線によるウィンドウ操作環境を実現するにあたり , 上記の問題点を考慮したシステムの検討をおこなった . 5 節でこの点について議論をする .

3 X-Window におけるウィンドウ操作

視線によるウィンドウ操作環境が構築されている X-Window では , ユーザはウィンドウマネージャを利用してウィンドウの操作をおこなう . ウィンドウマネージャはユーザの操作 (実際にはマウスやキーボードの操作によって発生するイベント) を監視しており , マウスポインタの位置に応じたマウスカーソルの変更 , マウスボタンの押下に対応したウィンドウの制御などをおこなっている . ウィンドウの作業をおこなう場合は , 対応するウィンドウにウィンドウを操作するため

表 1: Ewm の持つウィンドウ操作機能

機能	実行方法
ウィンドウフォーカスの変更	視線
ウィンドウを最前部に移動	視線
ウィンドウを後部に移動	コマンド
ウィンドウを移動	コマンド
ウィンドウをアイコン化	コマンド

のイベント (サイズの変更, 表示順序の変更など) を送る [1].

ウィンドウの操作にあたって, ユーザは以下の要素を決定する必要がある.

1. 操作をおこなう対象となるウィンドウ
2. 実行する操作の種類.

これらの決定方法はウィンドウマネージャの種類およびユーザの設定によって大きく異なる. 一般にウィンドウの上部にはタイトルバーが配置され, またウィンドウの周囲にウィンドウ操作の領域が配置されていることが多い. 図 1 にウィンドウマネージャの一例として *Fvwm 2.2* [2] の画面を示す.

まず, 操作をおこなう対象となるウィンドウ (アクティブウィンドウと呼ぶ) は, そのウィンドウのタイトルバーあるいはアプリケーションの作業領域を選択することで決定される. マウスカーソルが位置しているウィンドウが自動的にアクティブウィンドウとなる場合もある.

実行する操作の種類は, ウィンドウの周囲に配置された操作領域, あるいはウィンドウ操作のコマンドが配置されたメニューを利用して決定する. *Fvwm* ではタイトルバーに複数のボタンが割り当てられ, またウィンドウの四隅に大きさを変更するための領域を配置している. ユーザがマウスカーソルをこれらの領域に移動すると, マウスカーソルの種類が変化して視覚的フィードバックを受け, 操作が可能であることを知ることができる. また, ウィンドウマネージャによっては操作のコマンドメニューを表示して, ユーザは目的のコマンドをメニューから選択できるようになっている場合もある.

4 視線によるウィンドウ操作環境 Ewm

Ewm は X-Window において視線によるウィンドウ操作を実現したプロトタイプシステムである. ユー

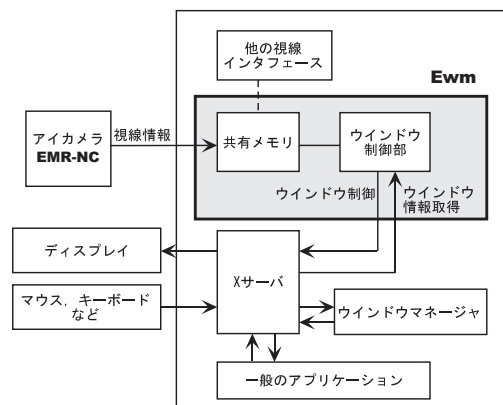


図 2: 視線によるウィンドウ操作環境 Ewm

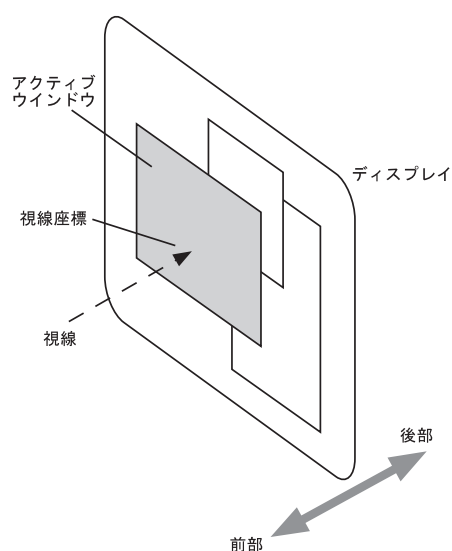


図 3: 視線によるウィンドウの操作

ザが普通に作業をおこなっている時にユーザの視線に応じたウィンドウの制御を自動的におこなう. また, 必要に応じてユーザはメニューの表示および操作コマンドの選択を視線によっておこなうことが可能である.

4.1 Ewm の構成

プログラムは Sun Sparc 10 (SS-10) 上で動作し, C および C++ によって実装されている. その全体構成を図 2 に示す. 視線の測定には, 非接触式アイカメラ EMR-NC [10] を利用している. EMR-NC によって測定された視線情報は共有メモリに保存され, 視線を利用する他のプログラムも, 共有メモリから視線情報を読み出すことで同時に利用することが可能である.

Ewm は共有メモリに保存された視線情報を定期的に取得すると共に, 現在 X-Window において利用されているウィンドウの種類, 表示位置, 大きさおよび表示

表 2: Ewm における操作の決定方法

決定する項目	決定方法
操作をおこなうウィンドウ	視線位置
実行する操作の種類	視線位置
	コマンド

順序を管理する。なお、視線情報の読み出しは 50msec 間隔である (実際には他の処理の関係でもう少し長い間隔になる場合もある。そのタイミングは保証されないが、Ewm を利用する上で特に問題とはならない)。

現在、Ewm は表 1 に挙げた機能を持ち、視線だけで最低限のウィンドウ操作をおこなうことができる。また、Ewm は通常のウィンドウマネージャとは独立に動作しているため、ユーザがウィンドウマネージャを利用しておこなうウィンドウ操作については特に制約は発生しない。

4.2 操作の選択

3 節で述べた通り、ウィンドウを操作するにあたっては、

1. 操作をおこなうウィンドウ
2. 実行する操作の種類

の 2 種類を決定する必要がある。しかしながら、一般的なウィンドウマネージャのように視線でウィンドウ周辺に配置された小さな操作領域を選択することは、極めて困難であり、また画面を見ている時と選択をおこなっているときの視線を区別することもできない。そのため異なる選択方法を利用する必要がある。

Ewm はこれらを以下の方法によって決定する (表 2)。

操作をおこなうウィンドウ

Ewm は他のウィンドウマネージャと同様に、現在選択されているウィンドウ (アクティブウィンドウ) を操作対象とする。アクティブウィンドウはウィンドウマネージャと共有されており、視線がアクティブウィンドウ上にある場合にはそのウィンドウが継続してアクティブウィンドウとなる (図 3)。

一方、視線が他のウィンドウ上にある場合、Ewm はユーザの視線を利用してウィンドウフォーカスを自動的に変更する。視線が存在するウィンドウを常にアクティブウィンドウとする方法も考えられるが、ウイ

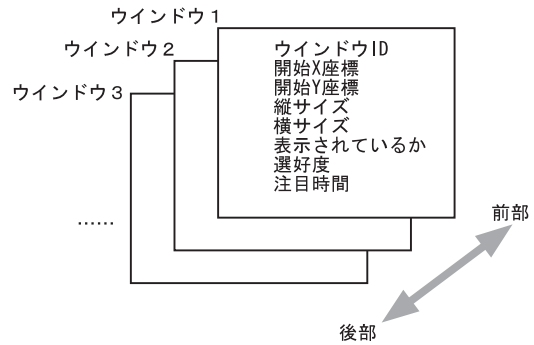


図 5: Ewm が保持するウィンドウ情報。ウィンドウの表示順序も同時に管理されている。

ンドウの境界付近に表示されている文字を見ている時に隣のウィンドウに視線が位置した場合、視線の移動中に、あるウィンドウにたまたま視線が位置した場合など、思わぬ時にウィンドウフォーカスが変更されて誤動作の原因になる。

そこで、Ewm では常にユーザがどのウィンドウを見ているのかを、画面に表示されている各ウィンドウについて 2 種類のパラメータで管理する。

- 選好度。ユーザが特定のウィンドウに注目している度合い。視線が位置するウィンドウは選好度が増加し、そうでないウィンドウは徐々に減少する。また、視線がアクティブウィンドウ上に位置していない場合でも、一定距離内にある場合は距離に応じて選好度が増加する。複数のウィンドウが重なっている場合、選好度が加算されるのは最上部に表示されているウィンドウのみである。
- 注目時間。各ウィンドウにユーザが注目した合計時間。ユーザが良く見るウィンドウは徐々に注目時間が加算される。複数のウィンドウが重なっている場合、注目時間が加算されるのは最上部に表示されているウィンドウのみである。

ウィンドウフォーカスの変更には選好度を利用しており、現在もっとも選好度の高いウィンドウがアクティブウィンドウとなる。従って視線がウィンドウを横切った程度ではウィンドウフォーカスの変更は発生しない。

なお、Ewm は他にもウィンドウの様々なパラメータおよびウィンドウの表示順序を管理しており (図 5)、これらはウィンドウの位置や順序が変化するとリアルタイムで更新される。Ewm はウィンドウの操作にあたってこれらのパラメータを利用している。

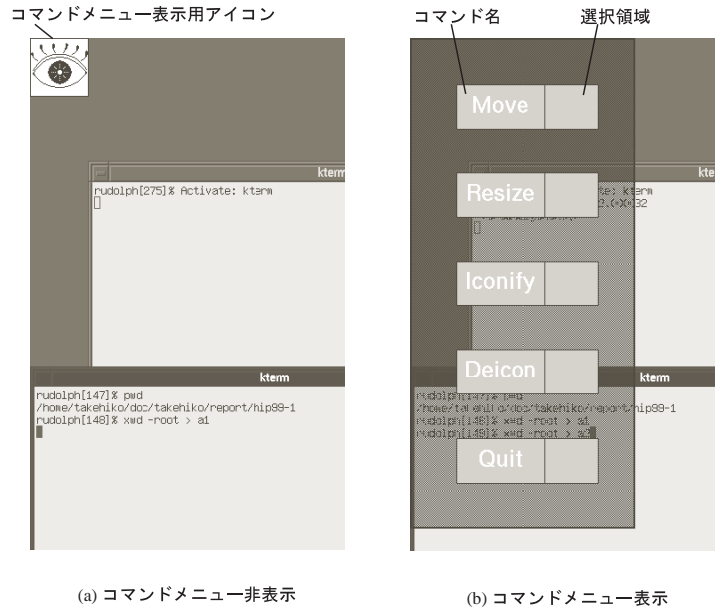


図 4: 視線によるコマンドの選択

実行する操作の種類

実行する操作の種類は、スクリーン上における視線の位置を利用して自動的に決定される場合と、ユーザが視線によって明示的にコマンドを選択する場合がある。

表 1において、ウィンドウフォーカスの変更およびウィンドウを最前部に表示する機能はユーザの視線を利用して自動的におこなわれる。

ウィンドウ表示位置の移動、ウィンドウの最後部への移動などの操作は、ユーザが明示的にコマンドを選択する(図 4)。画面端にコマンドメニュー起動用のアイコンが配置されており、コマンド上に視線がとどまった場合にコマンドメニューが表示される。メニューはコマンド名および選択領域に分離されており、視線がコマンド名に位置した場合は対応する選択領域に選択マークが表示され、視線が選択領域に位置した場合はそのコマンドが実行される。この選択方式はマウスに比べて高速に選択が可能であることが実験で確認されている [7]。この時、操作の対象となるウィンドウは現在のアクティブウィンドウである。また、コマンドの実行を希望しない場合は視線をメニュー外に位置することでメニューを閉じることができる。

4.3 Ewm の各機能

ここでは Ewm の各機能およびその実現方法について述べる。

ウィンドウフォーカスの変更

ユーザはあるウィンドウ上で作業をおこなう時、ウィンドウフォーカスをそのウィンドウに設定してアクティブウィンドウにする必要がある。この時、一般的なウィンドウマネージャではマウスを用いてタイトルバーおよびウィンドウ内部のクリックなどの方法でウィンドウをアクティブにする必要がある。

Ewm ではウィンドウの選好度がもっとも高いウィンドウをアクティブウィンドウとする。あるウィンドウ上で作業を続けているとそのウィンドウの選好度が高まるため、他のウィンドウを見てもフォーカスが移動するまでに時間を要するようになる。一方、すぐに視線を他のウィンドウに動かした場合はアクティブウィンドウの選好度が低いままであるため、ウィンドウフォーカスもすぐに移動する。その結果、次々にウィンドウのフォーカスを移動させることが可能となる。

ウィンドウを最前部に移動

ウィンドウが複数重なりあっている場合、背後にあるウィンドウの情報を利用するにはウィンドウを前面に移動させる必要がある。一般的なウィンドウマネージャでは、タイトルバーなどをクリックすることで、ウィンドウを最前部に移動させることができる。ウィンドウを最前部に移動させた場合は同時にアクティブウィンドウになるウィンドウマネージャが多い。

Ewm では、現在アクティブウィンドウでないウイ

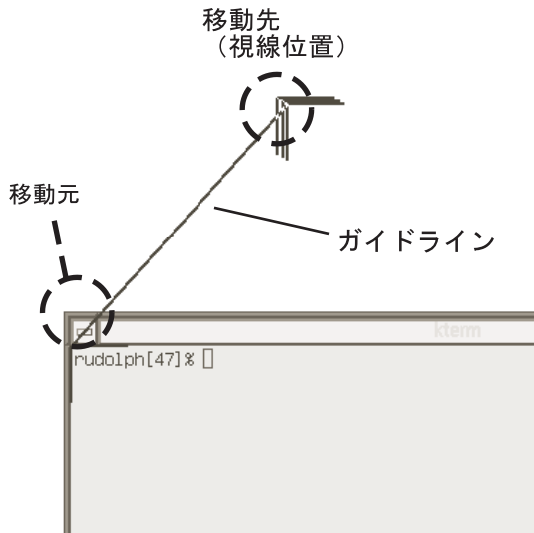


図 6: 視線によるウィンドウの移動

ンドウに対してウィンドウフォーカスが変更された場合に、同時にそのウィンドウを最前部で表示する。その結果、現在見ているウィンドウは常に他のウィンドウによって隠されることがない。

ウィンドウを後部に移動

ウィンドウが複数重なりあっている場合、前部に表示しているウィンドウを背後に移動させて、後部のウィンドウが見える必要がある。一般的なウィンドウマネージャでは、タイトルバーなどにはこの機能は特に割り当てられておらず、後部に移動するためのコマンドを選択して実行するが多い。

Ewm では、コマンドメニューからコマンドを選択することで実行される。対象となるウィンドウは現在のアクティブメニューである。コマンドを実行すると現在のアクティブウィンドウは最後部へ移動するとともに、アクティブウィンドウではなくなる。そのかわりに現在の視線座標にある最前部のウィンドウがアクティブウィンドウとなる。

ウィンドウの移動

ウィンドウの位置が現在の場所では好ましくない場合、ウィンドウの移動が必要となる。通常のウィンドウマネージャでは、タイトルバーあるいはウィンドウ外枠をマウスでドラッグすることでウィンドウの移動をおこなう。

Ewm ではコマンドメニューからコマンドを選択することでおこなわれる。この時、対象となるウィンドウはアクティブウィンドウである。ウィンドウの移動

においては、移動先も視線によって指定する必要があるが、Ewm ではユーザの視線および瞬目を利用して、ユーザが移動コマンドを選択するとアクティブウィンドウの左上端から現在の視線座標に対してガイドラインが表示される(図 6)。そこでユーザはウィンドウの移動先を確認してから瞬目をするすることで、移動先を決定する。Ewm は瞬目を検出して(実際には瞬目に伴う視線検出エラーを利用している)、その直前の視線座標にウィンドウを移動する。

5 議論

2節で述べた通り、視線を利用したインタフェースはユーザの意図通りに動作せず、作業効率がかえって低下する可能性がある。そこで、本節では視線を利用したウィンドウマネージャを実現するにあたって検討した点について述べる。

5.1 コマンドの利用

Ewm ではウィンドウの操作において、視線を利用した自動操作とユーザによるコマンド操作の両方を取り入れているが、どの機能をどちらで実現するかはトレードオフの関係にある。自動操作はユーザが何もしなくて良いので便利であるが、判別し失敗するとまったく思った通りに動作しなくなる場合もある。一方、コマンドの利用は確実であるが操作に時間を要し、またユーザはコマンドの意味と機能について学習する必要がある。

そこで Ewm では、通常のウィンドウマネージャとの併用を前提として、ウィンドウフォーカスの切り替えおよび表示順序の変更のみを視線による自動操作とした。これらはウィンドウを操作する上で頻繁に利用され、また自動判別に失敗した場合の副作用も深刻ではない。また、ウィンドウの操作には通常のウィンドウマネージャを利用することも可能であり、失敗してはいけない操作に関しては従来通りマウスを利用することも可能である。

5.2 視線による目的位置の指定

Ewm ではウィンドウの移動において視線で目的位置を定める。視線は元々細かい座標の指定には適さないが [7]、ウィンドウの移動では場所を細かく指定するよりは高速に移動させる方が重要と考えて視線による目的位置の指定を取り入れている。

目的位置をユーザにフィードバックさせるために、図 6でも示した通り、Ewm ではウィンドウ端から視

線座標へガイドラインを表示しており，ユーザはウィンドウの移動先をおおまかに知ることが可能である．視線によってウィンドウ全体を直接移動させるインタフェースも検討したが，試作の結果，極めて使いにくいことがわかったので却下した．ユーザはウィンドウ全体を一度で見ることはできずウィンドウ全体を見るための視線移動が発生するために，結果的に目標の場所へウィンドウを移動させることが難しくなるのである．この結果は，視線による物体の移動をおこなう場合は移動先を一点で指定できるデザインにする必要があることを示唆している．

5.3 アクティブウィンドウの選択方法

Ewm ではユーザがアクティブウィンドウを選択するのではなく，視線情報を利用して Emr が自動的に決定している．そのため，ユーザが想定していないウィンドウの境界付近に表示された文字を見ている場合，ばしば視線はウィンドウ外に位置するため，他のウィンドウがアクティブになりやすい．Ewm では視線を利用してウィンドウの選好度を逐次計算することでこの問題を回避している．なお，選好度の算出方法に関しては現在のところ試行錯誤で決定している．

6 まとめと今後の課題

本論文ではユーザが視線によってウィンドウを操作することのできるシステム Ewm を提案した．Ewm はアクティブウィンドウを自動的に決定することでフォーカスウィンドウおよびウィンドウ表示順序の変更を自動的におこなう．また，ユーザはメニュー中のコマンドを視線で選択することによって，ポインティングデバイスを用いることなくウィンドウ操作が可能である．その結果，通常のアプリケーションの利用を中断することなくウィンドウを操作することが可能となった．

現在は実際に著者がプログラム作成時などに利用して機能の拡張および問題点の抽出をおこなっている．今後はベンチマークとなる作業を定め，複数被験者による使いやすさの評価を定量的におこなう予定である．また，選好度の算出方法についても改良していく予定である．また，Ewm とウィンドウマネージャは独立に動作しているが，将来的には両者を統合することで，より使い心地の良い操作環境を実現することを検討している．

参考文献

- [1] Adrian Nye: Xlib プログラミング・マニュアル, ソフトバンク, 1993.
- [2] *Official FVWM Homepage*, <http://fvwm.math.uh.edu/>
- [3] Hansen J.P. et al.: *Eye-Gaze Control of Multimedia Systems*, Symbiosis of Human and Artifact, Elsevier Science, Vol.20A, pp.37-42 (1995).
- [4] Robert J. K. Jacob: *What You Look At Is What You Get: Eye Movement-Based Interaction Techniques*, Proceedings of CHI'90, ACM Press, pp.11-18 (1990).
- [5] Jacob R.J.K et.al.: *Interaction Styles and Input/Output Devices*, Behaviour and Information Technology, Vol.12, pp.69-79 (1993).
- [6] 大野健彦: 視線による高速な目標選択方法, 信学技報 HIP98-33, pp.9-16 (1998).
- [7] 大野健彦: 視線を用いた高速なメニュー選択作業, 情報処理学会論文誌, Vol.40, No.2, pp.602-612 (1999).
- [8] 高木啓伸: セレクションタスクにおける視線, インタラクティブシステムとソフトウェア IV, 田中二郎編, 近代科学社, pp.131-140 (1996).
- [9] 大和 正武 他: 視線によるテキストウィンドウの自動スクロール, 情報処理学会論文誌, Vol.40, No.2, pp.603-622 (1999).
- [10] 吉川 厚, 大野 健彦: 視線を読む - ユーザにやさしい視線測定環境 -, NTT R & D, Vol.48, No.4, pp.399-408 (1999).
- [11] Shumin Zhai, Carlos Morimoto and Steven Ihde: *Manual and Gaze Input Cascaded (MAGIC) Pointing*, Proceedings of CHI'99, ACM Press, pp.246-253 (1999).