キーボードによる窓操作機構の作成と評価†

久野 靖†† 角田博保†††

近年，ソフトウェア開発環境を中心にウィンドウシステムの使用が一般化しつつある．これらのシステムの大部分では窓（ウィンドウ）の位置や大きさなどの制御にマウスを使用する設計になっているが，プログラム開発や文書作成などの場合は特に利用者が手をキーボードに置いていることが多く，マウスに手を移す負担は無視できない．この問題を解消する試みとして，筆者らはキーボードのみを使用して窓を操作する機構を X-Window Version 11 上に作成した．また，いくつかの典型的な窓操作をこの機構とマウスを使用する機構の両方で行い，そのタイミングデータを採取して検討した結果，キーボードのみによる窓操作が有効であるとの結果を得た．本論文ではまずウィンドウシステムにおける窓操作機構の位置づけについての検討を述べ，続いてその検討に基づいて筆者らが作成したキーボードのみを使用するウィンドウマネージャ kwm の設計と実現について説明する．続いて kwm の有効性を評価するために kwm をマウスを使用するウィンドウマネージャ uwm と比較した実験について説明を行い，さらに実験結果の検討から上記の結論を導いている．

1. はじめに

近年，ソフトウェア開発環境を中心にウィンドウシステムの使用が一般化しつつある．これらのシステムの大部分では窓（ウィンドウ）の位置や大きさなどの制御にマウスを使用する設計になっているが，プログラム開発や文書作成などの場合は特に利用者が手をキーボードに置いていることが多く，マウスに手を移す負担は無視できない．この問題を解消する試みとして，筆者らはキーボードのみを使用して窓を操作する機構を X-Window Version 11 上に作成した．また，いくつかの典型的な窓操作をこの機構とマウスを使用する機構の両方で行い，そのタイミングデータを採取して検討した結果，キーボードのみによる窓操作が有効であるとの結果を得た．

以下では2章において窓操作機構についてまとめるとともにキーボードによる窓操作の可能性について検討し，3章でキーボードのみで窓を操作するウィンドウマネージャ kwm の設計と実現について述べる．続く4章では kwm とマウスを用いるウィンドウマネージャである uwm を用いて行ったタイミングデータ計測実験とその結果について述べる．最後に5章でまとめを行う．

† Invoking Window Manipulation Functions through Keyboard: Methods and Evaluation by YASUSHI KUNO (Graduate School of Systems Management, The University of Tsukuba) and HIROYASU KAKUDA (Department of Computer Science and Information Mathematics, Faculty of Electro-Communications, The University of Electro-Communications).

†† 筑波大学経営システム科学専攻
††† 電気通信大学電気通信学部情報工学科

2. ウィンドウシステムと窓操作

筆者らは，ウィンドウシステム上で毎回指令等を打ち込むのを，マウスを用いて画面上の情報を指示することである程度代えられるようなツール群[1]を作成し，またそれを使用した作業状況のタイミングデータを調べることによりこれらのツール群を評価する試み[2]を行ってきたが，これらの結果手がキーボードとマウスを往復するオーバヘッドは無視できない大きさであり，利用者の大きな負担になっていることがわかった．

にもかかわらず，現在のところ大多数のウィンドウシステムは窓を操作する(窓の間で入力を切り替える，窓の配置を変更する，等)場合にはマウスを使用する設計になっている．窓は画面上に2次元に配置されるため，それを操作するのにマウスを使用することは確かに自然だし直感的にわかりやすい．しかし，現在のソフトウェア開発環境では利用者はほとんどの場合キーボードに手を置いているので，常に上述のようなペナルティを払うことになり，そうしてまでマウスによる操作を採用するのが有利とは思えない．日常的にウィンドウシステムを使用し，したがってキーボードによるコマンドを覚えることが苦にならない利用者にとっては特にそうであるはずである．以上のような認識から，筆者らはマウスに代わってキーボードにより窓操作を行うことの可能性を検討してみた．ただし，これはマウスによる操作を否定するものではない（実際，筆者らが行っている他の研究[3)~5)]のようにマウスを積極的に使うものでは窓操作もマウスで行うのが自然であろう）．最終的に望ましいのは，どちらでも同程度

に容易に窓操作を可能にし，たまたま手があるほうのデバイスをそのまま利用させることであることを付記しておく．

まず，具体的に窓操作として分類し得る機能としては，次のようなものがあると思われる．

- 窓／アイコンの初期配置
- 窓／アイコンの位置の変更
- 窓の大きさの変更
- 窓の重なりの順の変更
- 窓とアイコンの間の行き来
- 窓への入力の切替え
- よく使う指令の起動

最後のものについてはこれを「窓操作」と呼ぶのは変に感じるかもしれないが，多くのウィンドウシステムの窓操作機構では利用者が窓操作メニューに好みの指令を起動する選択肢を付加することを許しているため含めた．新しい窓の生成，画面の描き直しなどの機能もこれに含めることにする．

次に，これらの機能を起動するにあたって指定しなければならない情報を整理すると次のようになる．

- 対象とする窓（またはアイコン）の指定
- 場所，大きさの指定
- 操作の種類の指定

マウスを使用した窓操作機構では「どの窓か」という情報はその窓の範囲へマウスカーソルを持っていくことで自然に指定できる（ただし窓が別の窓によって完全に隠されている時はまず見えるようにしなければならない，という欠点はある）．次に場所や大きさもマウスによって指定するのは問題ない．操作の種類については

- マウスボタンと修飾キーの組合せによる
- メニューによる
- 窓の周辺にそれぞれの機能に対応した領域を持つ

という方法を様々に組み合わせて使用している．

一方，キーボードによってこれらの情報を指定する場合には窓の指定，場所や大きさの指定ともにマウスのように「自明な」やり方ができないので工夫が必要である．窓の指定については窓の一覧を表示して選ばせることが考えられるが，毎回それを行うのは煩わしい．しかし頻繁に操作する窓の数はさほど多くないのが普通であるから，最初に選んだ時にその窓に固有の識別子を割り当ててしまえば，次回からはそれを用いてむしろ迅速に窓を指定できる可能性がある．場所や大きさについても，ウィンドウシステムを使用している利用者を観察すると窓をでたらめに配置していることは少なく，窓どうしが収まりよく並ぶようにしている場合が多い．とすれば，そのような位置や大きさがすばやく指定できるようにすれば（マウスで「ぴったり接する」などの細かい位置ぎめをするのは非常に時間を要するので）マウスより速やかに指定できる可能性も小さくない．以上の2点が解消すれば操作の種類の指定はキーボードで速やかに行えるので，全体としてマウスを使用する窓操作機構に優ったものを作ることは十分可能だと判断した．

3. kwm の設計と実現

前章で述べたような考えに基づき，筆者のうち1人が Sun-3/60 M 上の X-Window Version 11[7]（以下X）上にキーボードのみを使用して窓操作を行うウィンドウマネージャ kwm を設計し実現した．本章ではその概要について説明する．

3.1 kwm への入力

前章で述べた情報の入力以前の問題として，まず利用者の入力操作が窓操作のためのものか一般の入力なのかを区別する必要がある．マウスを利用する窓操作機構の場合には窓の周辺や背景部分でのボタン操作を窓操作のための入力として使うという自然なやり方が採れるが，kwm ではそうはいかない．そこで，比較的使用する可能性が小さい「シフトキーとコントロールキーを押しながらさらに英数字キーを押す」という組合せを kwm の指令専用に用いることにした（なお，以下ではシフトキー，コントロールキー，メタキーを押しながら他のキーを押す操作をそれぞれ S-, C-, M-を前置することで表す）．このほかに普段使わないような特別なキーを使う方法も検討したが，ホームポジションに手を置いたまま操作できなくなると意味がないので採用しなかった．キーを3個押すのは最初はやや負担に感じるが，慣れれば問題はなかった．また，kwm 窓（後述）が表示されている状態および窓の移動／大きさ変更のラバーバンドが出ている状態ではすべてのキー入力は kwm への指令となる．これらは一種のモードであるが，視覚的に他の状態とは明確に区別できるので特に混乱することはなかった．

3.2 窓／アイコンの初期配置

uwm, twm, awm などXで使われている多くのウィンドウマネージャでは新しい窓ができる時にその位置（と必要なら大きさ）を利用者がマウスで指定するようになっている．しかし文献 1）での経験によると，

窓はできる時に位置を指定させられるより適当なところにできてから後で動かすほうが使いやすかった．そこで kwm でもこの方式を踏襲し，現在開いている窓の配置を見て一番広く空いている場所に新しい窓を開くようにした．アイコンの位置も文献 1) と同様画面の右下隅から並んで配置されるようになっている．これについては個人の好みもあると思われるが，少なくとも窓が作られる度にそのアイコンを望ましい場所にいちいち移さなければならないのでは大変煩わしい．

3.3 窓の指定

2章で述べたように，窓を操作するためにはその対象となる窓を指定する必要がある．kwm では S-C-W を押すと**図1**にあるように，現在画面上にある窓の名前の一覧を表示する窓（kwm 窓と呼ぶ）が画面左下隅に現れる．一覧の中に白黒反転で表示されている窓（現在窓）が1つあり，その窓が各種の指令の対象となる．指令 p，n により反転位置を上下に動かして現在窓を変更することができる．

当初は表示されている窓の名前のみを見て窓を選択していたが，これでは画面上のどの窓が一覧表上のどれなのかわかりにくいため，反転表示の位置が移動すると同時に窓そのもの（またはアイコン化されていればアイコン）が一瞬白黒反転し，また元に戻るようにした．これによって「操作したい窓が反転するまで n と p を押していく」というやり方で視線を移さずに目的の窓を指定できるようになった．

3.4 窓への操作

現在窓に対する操作のうち最も基本的なのはその窓を選択する指令 s である．この指令は kwm 窓を閉じると同時に指定された窓に入力を切り替える．以後キー入力は（kwm が占有している S-C- なんとか，を除いては）すべてこの窓への入力となる．f/F は現在窓をそれぞれ重なりの一番手前／後ろに移す．i/I は現在窓のアイコン化／アイコンからの戻しを行う．m は現在窓を移動する指令である．移動を開始すると画面に窓と同じ大きさの枠が現れ，b/f/p/n または l/r/u/d の各キーでこの枠をそれぞれ左／右／上／下に移動できる（移動量については後述）．最後に [ret]（リターンキー）を押すと枠の位置に窓が移動する．r は現在窓の大きさを変更する指令であり，この指令が受け付けられると画面上に枠が現れ，その右下隅をm指令の場合と同様のキー操作で上下左右に動かして大きさを指定し（移動量については後述），最後に [ret] を押すと窓が枠の大きさになる．

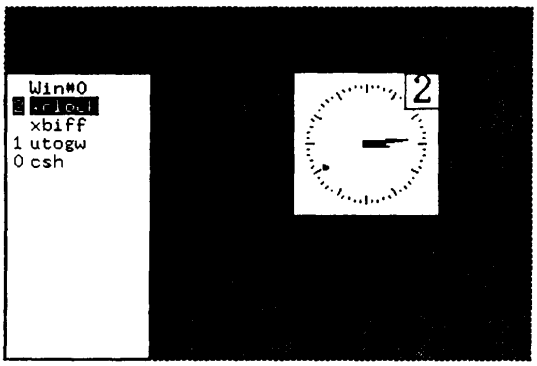

図1 kwm 窓（左）と識別子の表示された窓（右）
Fig. 1 A kwm-window and a window with an ID-label.

3.5 位置／大きさの指定

上記のうち，m と r についてはどこまで動かすかが問題となる．1つのやり方はある決まった量（例えば 16 ドット）ずつ動かすことであるが，それでは長い距離を動かす時非常に煩わしい．ここで利用者の目的という観点から見てみると，窓を移動するのは

（a） ある窓が別の窓を隠して（または別の窓に隠されて）いるので重ならない位置まで移動したい．

（b） ある窓と別の窓を並べて作業したいので重ならない範囲で接近させ（つまり隣接させ）たい．

の2つが主であると考えられる．そこで，例えば窓を右へ移動する場合にはその左端が他の窓の右端に揃う位置，およびその右端が他の窓の左端に揃う位置がそれぞれ(a)，(b)の場合に求める位置であることになるので，それらの位置に「グリッド」があるものと考え，1回 f キーが押されるごとに次のグリッド位置まで飛ぶようにした（残りの3方向も同様）．

大きさを変える場合も同様に考えると，他の窓に接するようになる位置にグリッドがあるとすればよい．ただし大きさを縮める場合には何も「引っかかりがない」場合も多いので，そのような時にはとりあえず大きさが半分ずつになるようにしている．

以上のようにして試してみたところ，かなり快適に窓の位置／大きさが制御できるように思われた．ただし，細かい単位で位置を制御したい場合も時々あるので，そのような場合のためシフトキーを押しながら n，p，f，b 等を押した場合には 16 ドットずつ，コントロールを押しながらの場合には 4 ドットずつ，メタキーを押しながらの場合には 1 ドットずつ移動するようにした．この結果，1ドットの位置を修正しようと

した場合にもマウスより楽に行える．

3.6 ショートカット

ここまでの状態でも kwm は結構実用的に使えるように思われたが，やはり kwm 窓の上でnとpで目的の窓を指定するのはかなり煩わしく感じる．特に窓への入力を切り替えるのは非常に頻繁に使用する機能なので，これが指令1個で行えないのでは不十分であると考えた．

そこで各窓のうち最大10個までに0〜9の id (識別番号) をつけることを許すようにした．id を付加するには，kwm 窓上で目的の窓が反転表示されている状態で 0〜9 のキーを押せばよい．いったん id がつけば，S-C-0〜S-C-9 を押すことでそれぞれの窓に（閉じていれば開き，重なりの後ろにあれば一番前に出た上で）入力が切り替わる．

当初はどの窓にどの id がついているかは kwm 窓上のみに表示していたが，それだけでは不十分で id を間違えることが多かった．そこで図1のように id がついている窓に対してはその右上隅に id を表示し，さらに入力がその窓に切り替わっている時は id が反転表示されるようにした．これによって特に意識しなくてもどの窓がどの id であるかが自然と目に入るため使いやすさが大幅に増した．

以上のほかにも S-C- なんとか，のキーはまだ多数残っていたので，これらを kwm 窓から指定する指令のショートカットとして使用している．具体的には

S-C-F　現在窓を重なりの一番奥に移す．

S-C-I　現在窓をアイコン化する．

S-C-M　現在窓の位置を移動する．

S-C-R　現在窓の大きさを変更する．

を用意した．移動と大きさ変更については指令起動後は m, r 指令と同様に操作する．kwm 窓の指令にあった「一番手前に持ってくる」と「アイコンから戻す」がないのは S-C-id によってアイコンであれば開き，一番前になければ手前に出てくるようにしたためである．

3.7 実現および使用経験

kwm はC言語により実現し，X用ライブラリとしては Xlib を直接使用している．最初の版が動くまでに1人月ほどを要したが，その後に細かい改良を経てここで説明した仕様になっている．現状ではコード行数は1,250行ほどである．

kwm を使用した経験では，当初はどうしても窓を直接指したくなり，無意識のうちに手がマウスに動いていることがあるが，やがて慣れてくるとキーボードのみで快適に操作できるようになり，今度は逆にマウスに手を移すのがうっとうしく思われるようになった．そのようにして十分 kwm に慣れてくると，主観的にははっきり kwm のほうが高速に操作できる感じがする．

4. kwm と uwm の比較実験

3.7 節で述べた主観的結果を客観的に裏付けるには注意深く計画された大人数の実験を行うことが望ましいが，ここではその「めやす」として実施した，筆者2名（以下 YK, HK と記す）による実験について報告する．本実験は kwm とマウスを使用する窓操作機構の両者で同一作業を実行し，そのタイミングデータを採取して比較するものである．被験者 YK, HK はともに日常的にウィンドウシステムを使用しており，kwm が想定する典型的な利用者であるといえる．また特に kwm に習熟した被験者のみを対象とした場合には実験結果の客観性は疑問となるが，今回の場合実験時点において，YK は kwm の開発者であるためその使用にある程度熟練していたのに対し，HK は数日前に初めて kwm を使用したという状況であった．以上のような事情から本実験でも最低限の客観性は保たれていると考える．また上記の事情から YK と HK のデータを比較することにより習熟の効果もある程度推し量ることができるが，それを裏付けるため，HK が十分習熟した時点での追実験も実施している．以下ではまず実験の全体的な設計について説明し，続いて各実験ごとにその概要と結果を述べ検討を行う．

4.1 実験の設計

比較のために使用する作業は全体として一般に行われる窓操作の典型例であることが望ましい．今回の実験では頻繁に行われそうな基本的窓操作として

1. 複数の窓の間で入力を切り替える．
2. 窓をアイコンにしたりアイコンから戻したりする．
3. 窓の位置を移動する．
4. 窓の大きさを変更する．

の4つを考え，それぞれについて典型的と思われる窓の配置と操作手順を用意した．別のやり方として，ウィンドウシステムを使っている利用者の操作系列データを多数集めて，そこからパターンを抽出し実験に使用することも考えたが，十分なデータを収集するのが難しかったため採用しなかった．

実験には Sun-3/50 M (SunOS 3.5) 上の X-Window ver. 11 Release 2 にタイミングデータを記録する改造（文献 2）と同様の方法による）を施したものを使用した．kwm との比較の対象としては，X 上のマウスを用いる窓操作機構として代表的であることから，uwm を選択した．熟練者が作業を行う場合を想定したため，各パターンおよび kwm/uwm の組合せについて一通り練習し，被験者が納得した状態で実験を始めるようにした．各実験は短時間で実行できるので，数回〜十数回を連続して実行し，集計時に各試行を切り分ける方法を採った．

全パターンにおいて，まず最初はどれかの端末窓で作業をしており，窓操作が終わったら再びどれかの端末窓での作業に戻るものとした．したがって個々には明記していない場合でもパターンの最初と最後で指令を 1 個実行している．指令としては打ちやすさから「ls」を選んだ．

4.2 実験結果の見かた

次節以降で各実験について説明する．「状況」はどのような状況でこのようなパターンが現れるかの例を示している．「概要」は窓操作機構の種類にかかわらず，窓がどのように操作されるかを説明している．さらに「検討」で実験の結果について検討している．併せてその実験の窓の配置図と結果のタイミングチャートを示す（図 2〜図 9）．各チャートの横線は時間軸を表し，その左端は最初の ls に続く [ret] を押したタイミングであり，右端は終わりの ls の「l」を押したタイミングである．縦線はその上で起きた事象を示す．1 つの縦線から成るのは打鍵であり，上にそのキーが何であるかを示した．2 つの縦線が横線でつながっているのはマウスボタンの押し／離しを示し，上にそのボタンがどれであるかを示した．実験は数回〜十数回反復して行った（被験者名の下のかっこで囲んだ数字はデータの件数を示す）が，図は元のデータから正常でないと判断したものを除外した後の平均を示す．右側に記した総所要時間も平均値である．各事象に付随する楕円はその直前の事象からの所要時間の標準偏差を示すもので，各事象のばらつき具合いを見るめやすとして記した．また，各操作系列をキーストロークレベル (KL) モデル[8] にあてはめ所要時間を予測した結果もチャートに含めた．KL モデルによる予測の計算方法については別に表 1 にまとめた．

4.3 実 験 A

状況：複数の窓で作業をしていて，次々に別の窓へ移りながら作業を進める．

概要：窓が 3 個並んでいる．まず窓 1 で ls を実行し，続いて窓 2 で ls を実行し，続いて窓 3 で ls

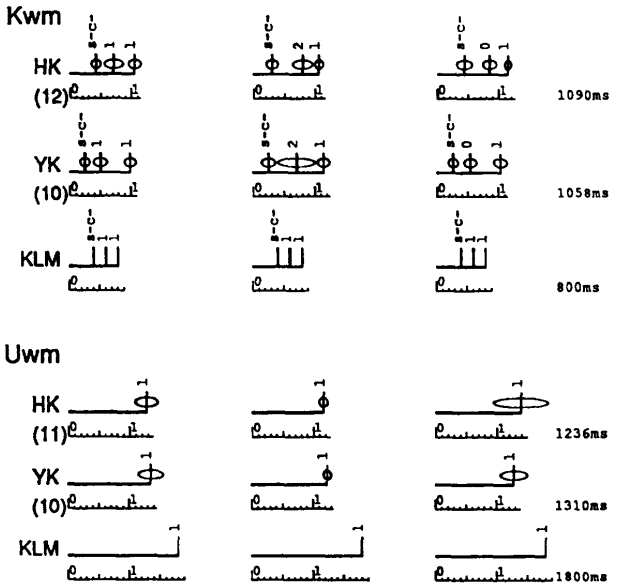

図 3 実験 A のタイミングチャート
Fig. 3 The timing chart of experiment-A.

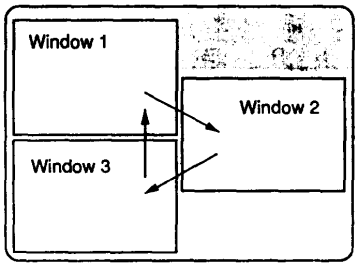

図 2 実験 A：窓間での入力切替え
Fig. 2 Experiment-A: Input switching between windows.

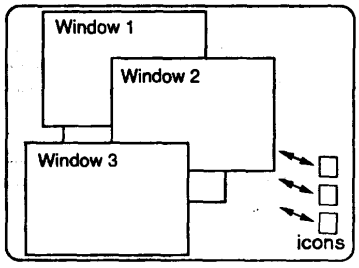

図 4 実験 B：アイコン化と戻し
Fig. 4 Experiment-B: Iconify and deiconify operations.

を実行.

検討：uwm では ls が終了したらマウスに手をやり，カーソルを次の窓に動かし，再びキーボードに手を移して ls と打つので間には計測される事象が何もない．ここでの総所要時間（おおよそ 1,300 ms）は「マウスに行く時間」+「動かす時間」+「マウスから戻る時間」+「l を打鍵する時間」ということになる．KL モデルによる見積りが過大なのは，マウスの移動目標がひどく大きいためきちんとマウスをつかまなくても済む，という特殊事情が反映されないためと思われる．

kwm では各窓にはあらかじめ 0，1，2 を id

表 1 キーストロークレベルモデルによる予測式
Table 1 Calculated operation times in keystroke-level model.

実験	方法	操作列	予測値
実験 A	kwm	$2K$[s-c-]K[0]K[1]	800
	uwm	HP_1HK[1]	1,800
実験 B			
往き (Iconify)	kwm	$2K$[s-c-]$6K$[I2I1I0]K[1]	1,800
	uwm	HK[*mouse*]P_1K[*mouse*]P_1K[*mouse*]P_1HK[1]	3,850
帰り (Deiconify)	kwm	$2K$[s-c-]K[1]K[2]K[3]K[1]	1,200
	uwm	HP_2K[*mouse*]P_3K[*mouse*]P_3K[*mouse*]P_1HK[1]	5,550
実験 C	kwm	$3K$[s-c-1]K[M]$5K$[rrrru]K[*CR*]K[1]	2,200
	uwm	HK[*down*]P_4K[*up*]HK[1]	2,900
実験 D	kwm	$3K$[s-c-M]K[u]K[*CR*]$3K$[s-c-R]$2K$[dd]K[*CR*]K[1]	2,400
	uwm	HK[*down*]P_3K[*up*]K[*down*]P_4K[*up*]HK[1]	4,120

K=200 ms（普通キー），150 ms（マウスキー），H=400 ms（キーボード，マウス間の移動），P_1=800 ms（窓の切替え），$P_2=P(500, 21)=1,260$ ms（遠くのアイコンの指示），$P_3=P(50, 12.5)=1,020$ ms（近くのアイコンの指示），$P_4=P(1,000, 4)=1,600$ ms（窓の移動と位置合せ），$P_5=P(400, 4)=1,460$ ms（近くの窓の移動と位置合せ），$P_6=P(400, 64)=1,060$ ms（窓の拡大）．
なお，$P(D, S)=800+100\log_2(D/S+0.5)$ による．（D は移動距離，S は目標の精度）

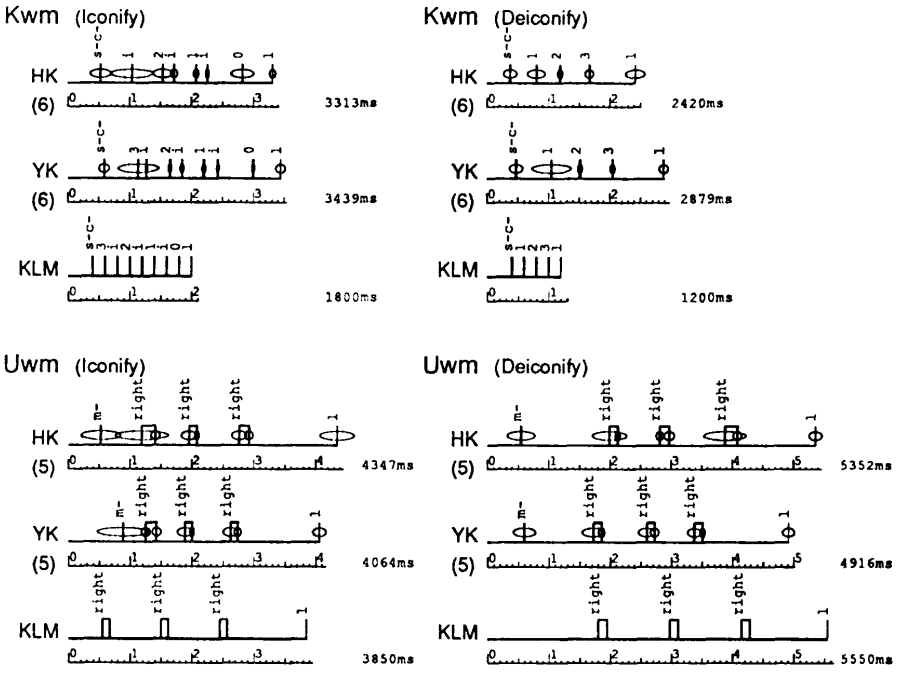

図 5 実験 B のタイミングチャート
Fig. 5 The timing chart of experiment-B.

として割り当てておき，ls に続き S-C-id で窓を切り替え，再び ls を実行している．所要時間は約 1,100 ms であり，uwm より約 20% 速いことになる．KL モデルの予測値はさらに短く 800 ms となっている．以下の実験すべてについてもいえることだが，kwm に対する見積りは思考時間を含めなかったため全体に実測値よりかなり小さい．この点についてはあとでまとめて述べる．

全体として所要時間の差がここでの実測値である 20% 程度だとしても，窓の選択操作はウィンドウシステムを使用していてもっとも頻繁に行うものなのでその影響は大きいといえる．

4.4 実 験 B

状況：窓がいっぱいできて邪魔なのでアイコンにする．または逆にアイコンにしてあった窓を開いて使う．

概要：「最初の」窓1個の上に別の窓3個が重なって乗っていて，そのうちの1個で作業をしている．3個の窓を閉じて，最初の窓に移る．逆では，最初の窓で作業をしていて，3個の窓を順次開けて最後の窓で作業をする．この実験では「往き」と「帰り」を分けて集計した．

検討：uwm では M-Right に「アイコン化／戻す」の機能を割り当てておき，3個の窓を順次アイコンにしてから最初の窓にポインタを移す．逆では次々に3個のアイコンを開く．kwm では最初の窓には0，残りには1～3の id が割り当ててある．往きでは各窓を S-C-id で選んで S-C-I でアイコンにし，最後に窓0を選ぶ．逆では各窓を S-C-id で選ぶだけで開く．

往きについてみると，まず uwm では所要時間は4秒強であるが，マウスとの行き来および3回の指示動作（P）のため本質的にこの程度の時間がかかるものと考えられる．KL モデルによる予測が実際のタイミングとよく似ていることもこれを裏付けている．一方，kwm では S-C- は1回押したらそのまま押さえていればよいから合計9打鍵ですべての動作が済む．全体として2割くらい kwm が速い．帰りについては kwm では6打鍵で済むのに対して，uwm では指示動作の数が多い上，対象が小さく所要時間が余計にかかる指示動作を含むため，両者の差はさらに大きい．

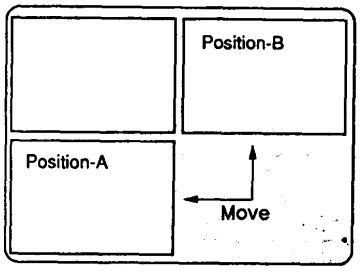

図 6 実験C：窓の移動
Fig. 6 Experiment-C: Moving a window.

4.5 実 験 C

状況：1つの窓に，別の窓を揃えて配置したくなって動かす．

概要：1つの窓の下隣に別の窓があり，その窓を最初の窓の右隣に移動する．

検討：uwm では M-Center ボタンに「窓移動」が割り当ててあり，これを使って動かしたい窓をつかんで移動し，よいと思った位置で離している．kwm では動かしたい窓を S-C-id で選択し，S-C-M で移動モードに入り，f，p 等のキーで移動し，望む位置に来たら [ret] を打っている．

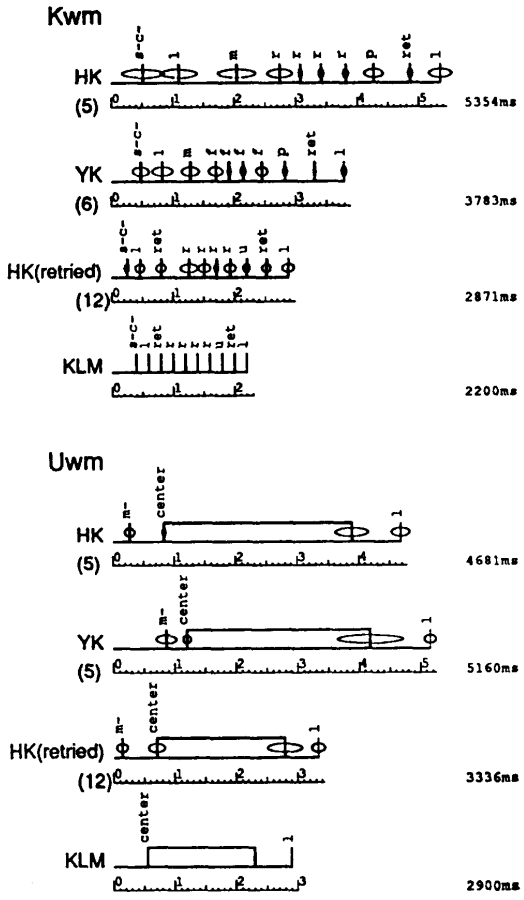

図 7 実験Cのタイミングチャート
Fig. 7 The timing chart of experiment-C.

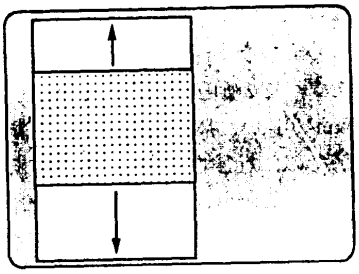

図 8 実験 D：窓の大きさ変更
Fig. 8 Experiment-D: Re-sizing a window.

uwm での所要時間のうち，指示動作の時間は窓を目指す場所にどのくらい正確に置くかにも依存している．今回の実験では 5mm 程度の精度で置くものとして実験を進めたが，実際に作業をする場合には（利用者の性格にもよるが）もっと正確に置くのが普通だと思われる．kwm では移動すると「ぶつかって止まる」のでそのような問題はない．

この実験では被験者 YK では kwm が速く，HK では uwm が速いという結果が出た．しかし，HK のデータを見ると全体的にキー操作が遅いことから，これはまだ kwm の指令を十分に覚えていないためと思われた．そこで最初の実験から 4 か月後に HK のみ再度実験を行い，その結果も併せて示した．これを見ると kwm での時間は大きく減って予測値に近づき，上述の予想を裏付けた（uwm の時間も減少したのは指示動作の精度が変わってしまったためと思われる）．したがって本実験においても kwm のほうが uwm よりも高速であるという結果が得られたことになる．

4.6 実 験 D

状況：できるだけ上下に大きな端末窓が欲しくなり，窓の大きさを広げる．

概要：画面のまん中に標準の大きさの窓があり，これをまず画面の一番上に移動し，続いてその窓の下端が画面一杯になるように大きさを変更する．

検討：kwm では S-C-M で移動モードになり 1 回上に移動すると画面の上端なのでそこで [ret] し，次に S-C-R で大きさ変更モードになり 2 回下へ伸ばして画面一杯になったところで [ret] を押す．uwm ではまず M-Center で窓をつかんで上へ移動し，次にそのまま同じ場所を M-Left でつかみ，下へひろげる．

この場合でも uwm は指示動作が 2 回あるのでそれだけで 2 秒程度かかる．なお，2 回なのは移動したあと下へのばす時はボタンを押し直すだけでポインタの移動は不要なためである．この実験も，HK と YK で結果が異なったため再実験を行ったが，やはり再実験では両者とも kwm が速いという結論を得た．また uwm のほうも速くなっているが，これは最初の実験では「そのまま同じ場所をつかむ」のに慣れていないため余計な時間がかかっていたことによる．

4.7 その他の実験および議論

ここに記したほかにもいくつか実験を行ったが，複数の基本操作が複合したものであることと，紙面の制約から概要のみを記す（詳しくは文献 6）参照のこと）．

新しい窓を作る操作について行った実験では uwm

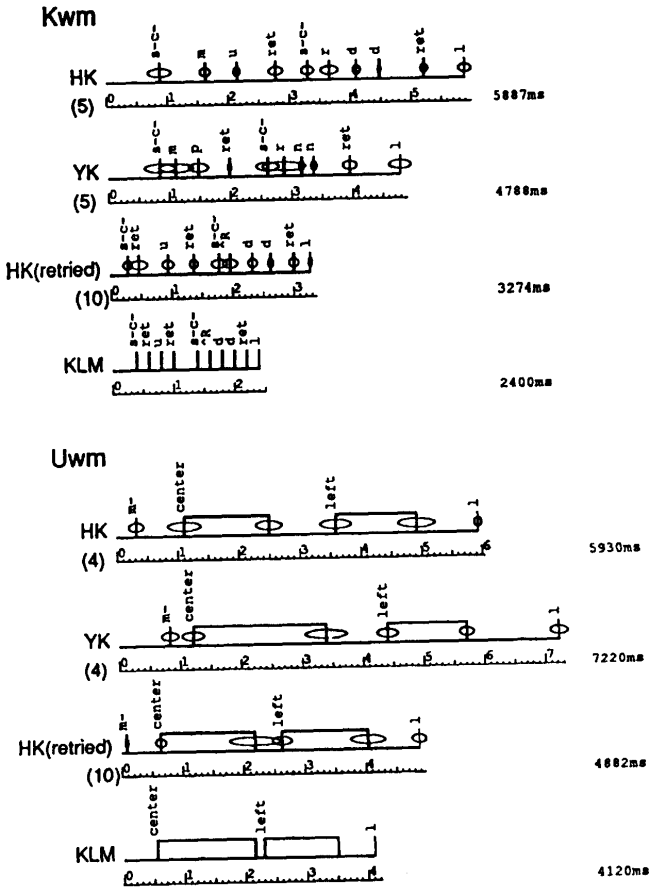

図 9 実験 D のタイミングチャート
Fig. 9 The timing chart of experiment-D.

でポップアップメニューにより操作を起動するよりもキーボードから指令を打ち込んで操作を起動するほうが速く，また窓が生成される位置をマウスで指定するよりも，kwm により適切と思われる場所に自動的に配置し，その後で利用者が好ましいと思う場所に移動するほうが操作は複雑であるにもかかわらずやや速いという結果を得た．もし自動的に配置した場所がはじめから好ましい場所なら完全に kwm が有利となる．

ある窓を別の窓の横に接して配置した後に画面からはみだしている分だけ窓の幅を狭める操作を行った実験では，uwm のようにマウスを使う場合には操作したい窓や大きさの基準となる窓が隠されたり画面からはみ出して見えない状態で操作することができないため余計な操作を必要とし，その結果 kwm に大きく劣るという結果が出た．加えて「位置を揃える」操作を精度よく行おうとするとさらに操作時間が増大する．

これらの実験全体を通じて，uwm における操作時間は KL モデルによる予測値と比較的よく一致し，被験者が合理的に操作を行っていることを示唆している．したがって，操作時間がこれ以上大幅に短くなる見込みはあまりないといってよいであろう．一方，kwm における操作時間は KL モデルによる予測値よりかなり大きい．これは予測において思考時間を全くないものとして扱ったためであり，実際にチャートを見ると操作の切れ目に当たるところ（つまり「考えそうなところ」）で余計に時間がかかっている例が多くみられる．しかし，kwm による操作に熟練してくると思考時間が次第に少なくなるものと考えられ，被験者 HK による再実験では大幅に KL モデルの予測値に近づいていることもそれを裏付けている．

5. まとめ

先にも述べたように，窓は2次元の画面上に配置されているのでその操作にマウスを使用するのは直感的にわかりやすい．一方，kwm のようにキーボードで窓を操作するのはまず操作のメカニズムを習得しなければならず，当初の心理的負担が大きい．実験結果でもこのことは，最初の実験における被験者 HK のデータで打鍵が平均して遅くなっているという点に現れている．しかし，操作に習熟してくると打鍵はマウス操作に比べてはるかに高速であるため，全体の操作時間としてほとんどの場合でマウスによる操作に優るようになる．実際，再実験における被験者 HK のデータは KL モデルによる予測時間にかなり近づき，習熟が進んだことにより時間が短縮されたことを端的に表している．

一方，マウスを用いた操作のハンデは当初の予測ではキーボードとマウスの往復に時間がかかる点にあると考えられたが，実験 B，D の結果では KL モデルでのマウスとキーボードの往復時間 800 ms を差し引いたとしてもまだ kwm のほうが速いという結果を得ている．これはマウスによる位置指定操作は，必要とされる精度にもよるが予想外に時間を要するため，結果として一見まわりくどく思えてもキーボードによる操作が優る場合が少なからずあることを示していると思われる．また，マウスによる操作系列は単純でありこれ以上あまり改良の余地がないのに対し，キーボードによる方法では kwm のやり方は一例にすぎず，よりよい方法を考える余地が残されている．

これらを総合すると，窓操作に限らず「画面上の対象物をキーボードによって操作する」というこれまであまり省みられなかった方式は，一般に行われているマウスによる操作と比較して，予想されたより広い領域で優っている可能性があるといえる．

参考文献

1) 久野 靖，角田博保：流れて行かない Unix 環境，情報処理学会論文誌，Vol. 29, No. 9, pp. 854-861 (1988).
2) 角田博保，久野 靖：流れて行かない Unix 環境の評価，第 29 回プログラミングシンポジウム報告集，pp. 95-104 (1988).
3) 佐藤直樹，久野 靖，鈴木友峰，中村秀男，二瓶勝敏，明石 修，関 啓一：CLU マシンのユーザーインターフェース，第 29 回プログラミングシンポジウム報告集，pp. 13-22 (1988).
4) 久野 靖：ビットマップとマウスの使い方，JUS 第 11 回 Unix シンポジウム発表論文集，pp. 15-18 (1988).
5) 久野 靖：新しいプログラマ・インタフェースの利用，情報処理，Vol. 30, No. 4, pp. 396-405 (1989).
6) 久野 靖，角田博保：窓はねずみ無しでも操れるか？，第 30 回プログラミングシンポジウム報告集，pp. 93-104 (1989).
7) Scheifler, R. W., Gettys, J. and Newman, R.: *X Window Systems*, 701 pp., Digital Press, Bedford, Massachusetts (1988).
8) Card, S. K., Moran, T. P. and Newell, A.: The Keystroke-Level Model for User Performance Time with Interactive Systems, *Comm. ACM*, Vol. 23, No. 7, pp. 396-410 (1980).

(平成元年9月27日受付)
(平成2年2月13日採録)

久野　靖（正会員）

1979年東京工業大学情報科学科卒業．1984年同大学院博士後期課程単位取得退学．同年東京工業大学理学部情報科学科助手．1989年筑波大学経営システム科学専攻講師．理学博士．プログラミング環境，並列プログラミング，オブジェクト指向言語，分散OSなどに興味をもつ．日本ソフトウェア科学会，ACM 各会員．

角田　博保（正会員）

昭和25年生．同49年東京工業大学理学部情報科学科卒業．同51年同大学院博士課程修了．同57年同大学院博士課程修了．同年電気通信大学計算機科学科助手．理学博士．文字列処理，プログラミング方法論，ヒューマンインタフェース等に興味を持つ．ACM，日本ソフトウェア科学会各会員．