原著論文

Vol.15 No.3, 2013

重なりあったウィンドウ間を移動可能なマウスカーソル操作手法とその評価

山中 祥太 *1　宮下 芳明 *1 *2

Mouse Cursor Operation for Overlapped Windowing and Its Evaluation

Shota Yamanaka*1 and Homei Miyashita*1 *2

Abstract – **When we perform a task that involves operating many windows, we cannot access the objects behind them. As a result, we are frequently forced to switch the foreground window or temporarily forced to move it. Such window manipulations arise because of the difference between the dimensions of the windows and the cursor. In this paper, we propose a technique, in which the cursor is able to go underneath windows using a normal mouse. The results of our user study show that our technique allows users to perform a task faster than they typically operate windows using mouse-only and mouse-keyboard operations.**

Keywords : **Graphical User Interface, Mouse Cursor Operation, Windowing, Pointing Technique**

1. はじめに

マルチウィンドウ環境においては，アプリケーションは個々のウィンドウ内に表示され，操作対象のウィンドウを切り替えることで複数の作業を行う．マルチウィンドウの表示方法には，ウィンドウ同士が重なりあうことを許すオーバーラップウィンドウ方式と，ウィンドウを重ならないように敷き詰めるタイルウィンドウ方式がある．現在はオーバーラップウィンドウ方式が主流であり，Windows OS や Mac OS など多くの GUI 環境で採用されている．

オーバーラップウィンドウ方式において，ユーザは作業時に多くのウィンドウ操作を行っている．移動やサイズの調整はもちろんのこと，ウィンドウをクリックして最前面に出す操作も頻繁に行われる．これらはアプリケーション操作やファイル整理など，本来行いたい操作に付随して必要になってしまうものであり，ときにはそのようなウィンドウの操作量が多くなることがある（具体例は 4.1，4.2，4.3 節で挙げる）．

付随的なウィンドウ操作が必要になる原因は，ウィンドウとマウスカーソルの次元が異なることにあると考えた．図 1(A) は複数のウィンドウが重なりあった状態の 3 次元イメージであり，(B) は普段我々が見ている視点からのイメージである．ウィンドウはこのように奥行きを持って 3 次元に配置されるのに対し，カーソルは (B) 視点から見て上下左右の 2 次元方向にしか移動できないため，奥にあるウィンドウは最前面に出してから操作を行わなければならない．

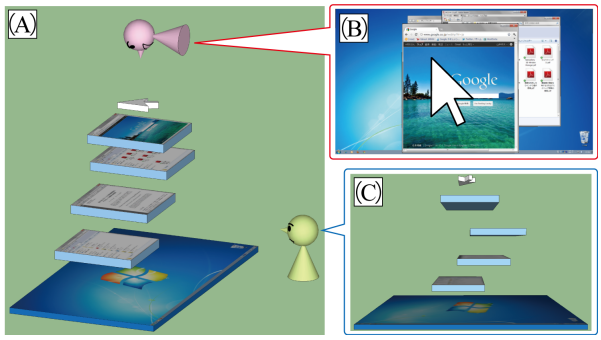


図 1　(A) 重なりあったウィンドウに対する 2 つの視点．(B) 通常我々がウィンドウを見ている視点からのイメージ．(C) 通常の視点から 90 度回転した位置からのイメージ．

Fig. 1　Views of overlapping windows from different perspectives.

最前面のウィンドウを切り替える方法としては，ウィンドウをクリックする，Windows ではタスクバーのアイコンをクリックする，Mac では Exposé（全てのウィンドウを縮小して一欄表示し，目的のウィンドウを選択する機能）を利用するといった方法がある．また，Windows では Alt+Tab や Win+Tab，Mac では Command+F1 といったキーボードショートカットでアプリケーションを切り替えることも可能である．しかし，奥のウィンドウに対して一時的な操作をする場合であっても，一旦奥のウィンドウを最前面に出し，

*1: 明治大学大学院理工学研究科新領域創造専攻ディジタルコンテンツ系
*2: 独立行政法人科学技術振興機構 CREST
*1: Program in Digital Contents Studies, Programs in Frontier Science and Innovation, Graduate School of Science and Technology, Meiji University
*2: JST CREST

さらに元の作業に復帰するために再度ウィンドウを切り替える必要があり，目的の行為に付随したウィンドウ操作量が多くなってしまう．ウィンドウ同士が重ならないようにするためには，位置やサイズの微調整が求められ，さらに各ウィンドウを各作業に適したサイズで使えないという問題も生じる．また，作業で主にマウスを使用する場合には，マウスのみで目的の操作をスムーズに行えることが望ましく，キーボードを用いない操作手法が求められると考える．

本論文では，左右ボタンを同時に押しながらマウスを動かし，カーソルをウィンドウの端から滑りこませることでウィンドウの奥にあるオブジェクト（アイコンやウィンドウ）を操作可能にする手法を提案する．既存研究では，特殊な機器を使用する，ドラッグ中などに操作状況を限定する，あるいは従来の操作を一部妨げてしまう手法で解決が試みられることが多かった．それに対し提案手法は，一般的なマウスのみを使用し，広範な状況下で利用可能で，従来の操作を妨げない．

筆者らはこれまで，提案手法を従来手法のマウス単独操作及びマウス・キーボード併用操作と比較した結果について報告している[1],[2]．本論文では，被験者から得られた主観評価，および実験から数日経過した後に得られた感想を取り上げての考察・議論も行う．

2. 解決しようとする問題と解決方法への要求

2.1 解決しようとする問題

本来行いたい操作をする際に，それに付随したウィンドウ操作が必要な場合がある．その原因の1つには，ウィンドウとカーソルの次元の相違がある．ウィンドウはZオーダ（手前から奥へ並ぶ順序）を持って3次元に配置され，レイヤ構造をなしている．一方，カーソルは2次元平面上でしか移動できないため，ウィンドウに隠れている奥の部分は操作できない．よって，奥のウィンドウをクリックして最前面に出すか，あるいは手前のウィンドウを移動させたり最小化するなどの操作を経てから目的の操作を行わなければならない．

さらに目的の操作を終えてから元の作業に復帰するために，初めに最前面にあったウィンドウをクリックして最前面に出し直したり，移動したウィンドウを当初の位置に戻す操作が必要になる．こういった復元操作まで含めた付随的ウィンドウ操作は，たとえば「隠れた位置にあるフォルダ内のファイルをダブルクリックして開きたい」などという奥のレイヤへの一時的なアクセスに対しても毎回必要になり，操作の手数や所要時間が増大する原因となる．

このような付随的ウィンドウ操作を排するために，予めウィンドウ同士が重なりあわないようにサイズや位置を調整しておくことも可能である．しかし，これが可能なのは後に必要となるウィンドウ操作が見越せる場合に限られる．さらに，各ウィンドウのサイズ・位置の設定が互いに影響しあうため，それぞれに適した大きさで使えない問題が生じる．

したがって従来手法では，個々のウィンドウのサイズ・位置を自由に設定できるオーバーラップウィンドウ方式の利点を維持したうえで，手前のウィンドウに覆われた位置にあるオブジェクトを操作するのは困難である．本論文ではこの問題を対象とし，付随的ウィンドウ操作の煩雑さを解消することを目指す．ここから得られる効果には，作業全体にかかる時間の短縮や，本来の操作以外に注意を向けなくてよいことの主観的な快適さが挙げられるが，本論文ではまず局所的に得られるであろう効果としてウィンドウ操作時間の短縮を扱うものとする．

2.2 解決方法への要求

前節で挙げた次元の相違を解消することによる問題解決方策には，カーソルをウィンドウに合わせて3次元に移動させるか，あるいはウィンドウを2次元平面上のみに配置する方法が考えられる．後者はタイルウィンドウ方式と呼ばれ，これを採用するOSもあるが，個々のウィンドウサイズが小さくなることから視認性に問題が生じ，オーバーラップウィンドウ方式に比べると利用されていないのが現状である．

Mac OSのExposéは，全てのウィンドウを一時的に最前面に移動させることで，最前面にあるカーソルにウィンドウ側の次元を合わせる解決方法をとったシステムと見ることができる．しかし，ユーザはウィンドウの配置を空間的・視覚的に記憶する[3]ため，システムが自動的にレイアウトを変更したり前後関係を崩したりすると，ユーザは配置の記憶を頼れずに目的のウィンドウを探索する必要がある．ただし，ウィンドウが移動する際に元の配置からアニメーションさせることで記憶と関連付けやすく工夫されてはいる．ウィンドウの空間的・視覚的記憶をより直接的に利用するためには，ウィンドウの前後関係まで含めた配置を変更しないのが望ましいと考える．

したがって，本論文ではカーソルをウィンドウ側に合わせて3次元方向に移動させる方針をとる．しかし，カーソルを3次元方向に自由に移動可能にするだけの手法は過剰な機能追加であり，操作性を損なうと筆者らは考える．カーソルを奥行き方向へ自由に移動可能にすると，それに伴って適切な3次元入力インタフェース（専用デバイスの利用ないしGUI上）での操作が必要になり，従来の操作に比べ操作時の負担が著しく増大するためである．さらに，3次元に移動可能なカーソルは本来図1(C)のように離散的に配置されたウィンドウに対するアクセスのみが必要なのであっ

て，その中間地点の何もない空間でもカーソルを制御可能にするのは冗長な設計であるといえる．したがって，解決方法に求められる条件は，3 次元方向に離散的に配置されたウィンドウに対してカーソルをアクセス可能にし，ユーザが従来の操作に加えて僅かな負担で操作可能になる手法であると考える．

以上をまとめると，解決方法への要求事項は次のようになる．

- カーソルを奥行き方向へ移動可能にする．ただし，ウィンドウが配置された離散的なレイヤにのみ移動する．
- 一般的なポインティングデバイス（ここではマウスとする）のみで操作できる．
- ウィンドウは奥行き方向も含めて配置を維持する．

3. 提案手法

本論文で提案するのは，一般的な左右ボタンを持ったマウスのみを利用し，目的のウィンドウにカーソルを衝突させるようにしつつ奥へ移動させる手法である．この手法には次の利点がある．

- 手前のウィンドウに覆い隠されたオブジェクトを，カーソルが奥行き方向へ移動することでウィンドウ操作をせずに操作できる．それによって「デスクトップアイコンをダブルクリックしてアプリケーションを起動したい」,「奥のフォルダに入っているファイルを手前に移動したい」といった本来行いたい操作のみを遂行でき，操作の負荷が軽減される．
- 特殊なデバイスを必要とせず，マウスのみで操作可能である．カーソルがウィンドウ間の中間地点に位置しないように制限することで，3 次元入力が可能なインタフェースを使わずに奥行き方向への移動を実現する．ユーザは従来のマウス操作に加えて「マウスの左右ボタンを両方押す」という操作を習得するだけでよい．
- ウィンドウの移動やリサイズ，前後関係の変更が不必要なため，奥のオブジェクトを操作するときにウィンドウレイアウトを維持できる．これにより，一時的にウィンドウを移動させてから奥のオブジェクトを操作し，その後で元の作業に復帰するためにウィンドウを戻すといった二度の手間がかかるのを回避できる．

以下では提案手法の操作について説明する．

3.1 カーソルを奥方向へ移動させる操作

カーソルが奥行き方向へ移動する様子を図 1(C) 視点から見たイメージを図 2 に示す．マウスの左右ボタンを両方とも押している（以下，両押しと呼ぶ）間は，カーソルがウィンドウに衝突するまで奥へ移動し，そのウィンドウの範囲からカーソルが外れればさらに奥へと移動する．このように，両押しを維持したままマウスを動かすことで，段々とカーソルを奥方向へ移動させ，目的のオブジェクトがあるレイヤまでウィンドウ操作なしでカーソルを到達させることができる．

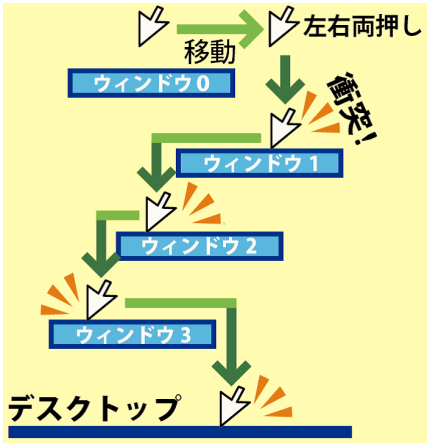


図 2 奥のレイヤへ移動しようとするカーソルがウィンドウに衝突しつつ潜りこんでいく様子．
Fig. 2 A cursor goes down to the lower layer of the window from the edge of the upper layer of the window.

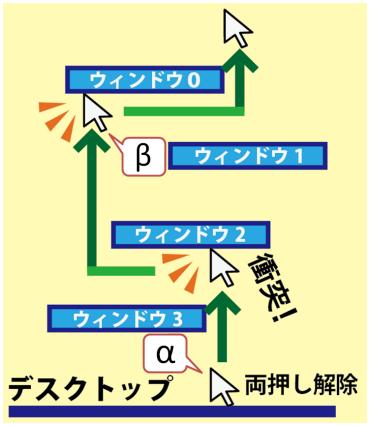


図 3 手前のレイヤへ移動しようとするカーソルがウィンドウに衝突しつつ上昇していく様子．α の位置で両押しを解除するとカーソルはウィンドウ 2 に衝突する．β の位置でマウス操作を行うと，マウスイベントはウィンドウ 3 に送出される．
Fig. 3 A cursor goes up to the upper layer of the window from the edge of the lower layer of the window.

3.2 カーソルを手前方向へ移動させる操作

両押しでないとき，すなわち左右ボタンの少なくとも一方が離されている間は，カーソルはウィンドウに衝突するまで手前に移動し，そのウィンドウの範囲からカーソルが外れればさらに手前へと移動する．たとえば図 3 のように，カーソルをデスクトップに到達

させた位置（α）で両押しを解除すると，カーソルは手前のウィンドウ 2 に衝突し，そのレイヤに留まる．カーソルを動かしてウィンドウ 2 の範囲から外れれば，さらに手前にあるウィンドウ 0 に衝突し，さらに最前面まで戻ってくる．つまり両押しをしていないときには，カーソルは常に手前へ移動しようと振る舞っており，手前にウィンドウがあるか既にカーソルが最前面のときに手前移動が遮られるという設計である．言い換えれば，カーソルを手前へ移動させたくないときには，手前のウィンドウにカーソルを「引っかけておく」ことで，カーソルが潜りこんだ状態を維持することができる．

3.3　ウィンドウの奥に潜りこんだ状態でのマウス操作

提案手法では，マウスイベントはカーソル直下にあるウィンドウに送出される．たとえば図 3（β）の位置でダブルクリックなどを行うと，マウスイベントはウィンドウ 3 に送出される．このようにして，本来はウィンドウに覆われてアクセスできないオブジェクトを操作することができる．手順をまとめると，目的のオブジェクトを操作するためには次のようにすればよい．

1. 両押しを維持してカーソルを奥へ移動させる．
2. カーソルの手前にウィンドウがある状態で両押しを解除し，カーソルを引っかける．
3. 目的のオブジェクトを操作する．
4. 引っかかっているウィンドウからカーソルが外れるように動かし，手前方向に移動させて最前面に戻す．

4.　提案手法の利用場面

筆者らの考えは，従来の方法でのウィンドウ切り替えをせずに提案手法を常に利用すべきだということではない．ある程度の時間他のアプリケーションだけを利用するのであればそのウィンドウを最前面に出した方が良いであろうし，キーボード操作がふさわしい場合にはそうすべきである．ただ，もしそういった操作が煩雑で，提案手法が有効にはたらきそうな場面ではこちらを利用すべきである，という主張である．そのため，提案手法は従来の操作方法と共存でき，場面に応じた柔軟な対応が可能である．現状のオーバーラップウィンドウ方式と親和性が高く，ユーザが慣れていた操作を阻害しないことも特長である．

以下では，提案手法が有効にはたらくと考えられる場面の具体例を 3 つ挙げる．また，従来手法および提案手法での操作例も併せて述べ，提案手法によって得られる利点について説明する．

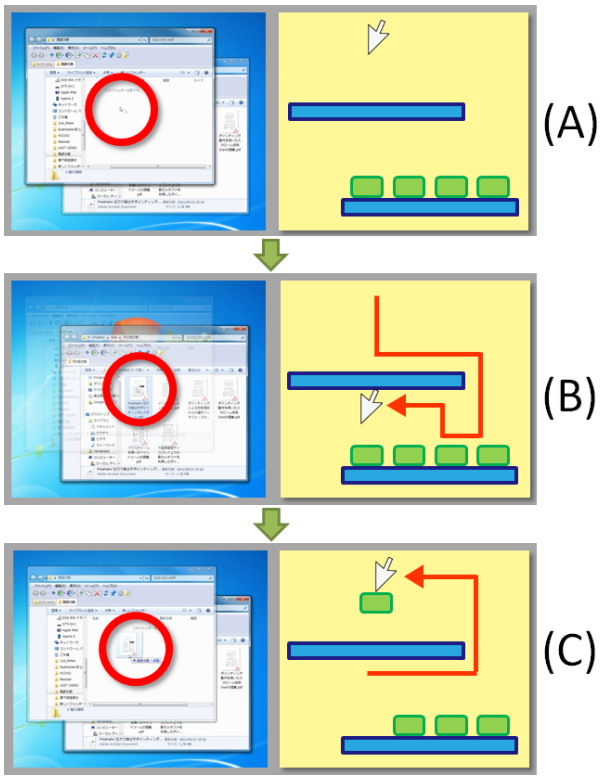


図 4　2 つのフォルダが重なりあって配置されている状態．カーソルの初期位置 (A) からウィンドウの奥にカーソルを潜り込ませ (B)，ファイルアイコンを手前にドラッグアンドドロップする (C)．右の図は左のスクリーンショットを 90 度回転した位置から見た模式図である．スクリーンショット中の赤丸はカーソルの位置，模式図中の緑矩形はアイコンを示す．
Fig. 4　Two folders overlap each other.

4.1　奥のフォルダから手前へドラッグアンドドロップする場合

例えば図 4 のように 2 つのフォルダを開いた状態で，奥の隠れた位置にあるファイルを手前のフォルダへ移動する操作を考える．通常であれば，手前のウィンドウを移動したり，奥のウィンドウをクリックして最前面に出したりすることで，目的のファイルと移動先のフォルダが同時に見えている状態にしてからドラッグアンドドロップする必要がある．提案手法では，奥へ潜ってファイルをドラッグし，手前に移動してドロップするという一連の動きで操作が完了する．ウィンドウに対する操作をせずに目的の操作ができるため，操作量が減少するだけでなく，意図したウィンドウの配置を崩さないのも利点である．

4.2　奥のウィンドウを操作すると手前のウィンドウが隠れる場合

図 5 のように，ウェブブラウザ（ア）を利用しながら音楽プレイヤ（イ）を起動している状態で，ブラウザ（ア）に隠れた位置にある音楽ファイルを再生する

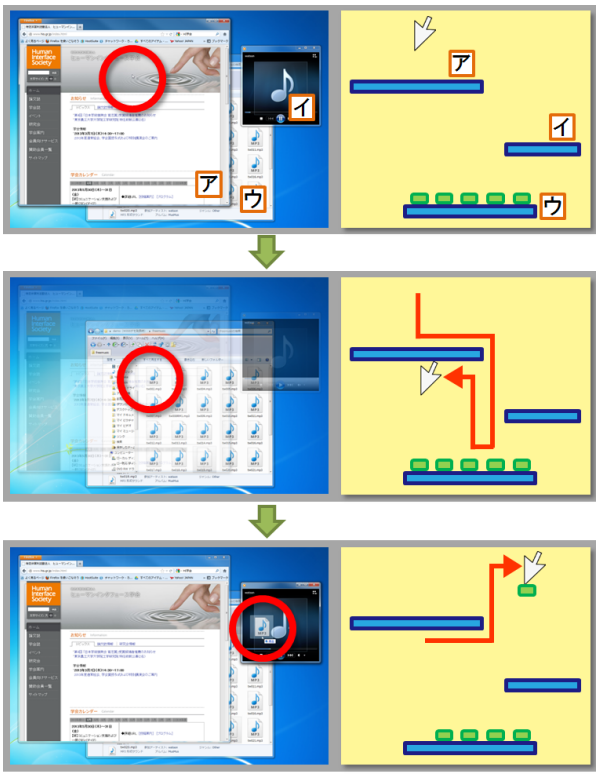


図 5　音楽を再生しながらウェブブラウジングする様子.
Fig. 5　Web browsing while playing music.

操作を考える．通常は，奥のフォルダ（ウ）をクリックして最前面に出し，目的のファイルをプレイヤ（イ）にドラッグアンドドロップする方法が考えられる（プレイヤが関連付けられていれば，ファイルをダブルクリックでも可）．しかし，フォルダ（ウ）をクリックした時点でブラウザ（ア）の一部が隠れてしまうため，操作が完了してからブラウザ（ア）をクリックして再度最前面に出すか，あるいはウィンドウ同士が重ならないように位置を調節する必要があり，多くの操作が求められる．提案手法では，ブラウザ（ア）の奥に潜って音楽ファイルをドラッグアンドドロップするだけでよく，奥のフォルダ（ウ）を一旦最前面に出す操作も，ブラウザ（ア）を再度最前面に戻す操作も必要ない．

4.3　デスクトップのアイコンを操作する場合

図 6 のように，ウィンドウが複数起動しているときにデスクトップのアイコンをダブルクリックしてアプリケーションを起動する場合を考える．通常は，起動中のウィンドウを全て最小化したり移動させたりした後に，デスクトップで目的のアイコンをダブルクリックし，さらに元の作業に復帰するためにウィンドウの状態を戻さなければならない．これに対し提案手法では，デスクトップにカーソルを潜りこませて目的のアイコンをダブルクリックするだけでよい．起動中のウィンドウに対する操作は一切必要ないため，操作量

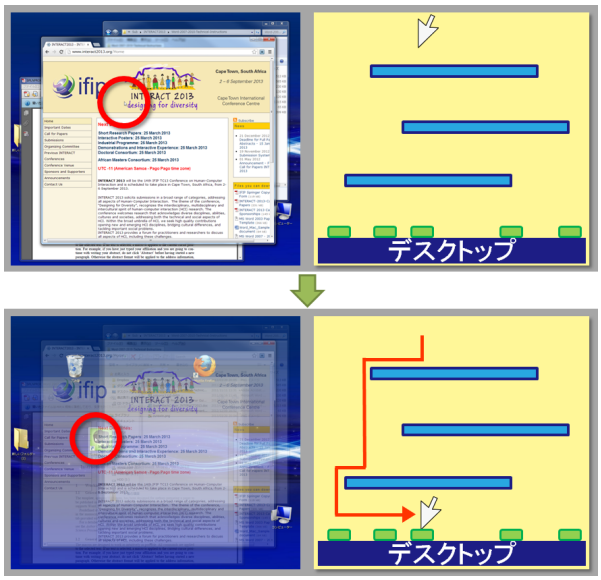


図 6　カーソルが 3 つのウィンドウの奥へ潜りこみ，デスクトップ上のアイコンを操作する様子.
Fig. 6　A cursor goes to the underneath of 3 windows.

が大幅に軽減され，さらに意図したウィンドウの配置を崩さない．

5.　システム

提案手法を実現するためのシステムを，Hot Soup Processor 3.3（以下 HSP）及び Win32API を用いて実装した．本システムは Windows7 上で動作し，表示される全てのウィンドウに対応できる．以下では，カーソルの挙動・表示を制御するモジュールと，ウィンドウを制御するモジュールに分けて実装方法を述べる．

5.1　カーソル制御部

5.1.1　マウスイベントの管理

提案手法ではマウスの左右ボタンを両方とも押下している（以下両押しと呼ぶ）ときにカーソルを奥へ移動させる．この操作を通常のクリック操作などと共存させるためにマウスフックライブラリ (mousehook.dll) を使用する．このライブラリにより，左右のボタンダウン・ボタンアップをフックし，ユーザによるボタン押下のイベントをプログラム側で自在に非送出にする．また，特定の座標にマウスイベントを送出するために，Win32API の mouse_event 関数を使用する．

一方のボタンが押下されてから 100 ミリ秒以内に他方のボタンが押下されれば，左右ボタンが同時に押下されたと見なしてボタンダウンイベントは送出しない．この場合，両押しされているので潜りこむ機能を発動する．一方のボタンが押下されてから 100 ミリ秒経過すれば，そのボタンダウンイベントをそのまま送出する．これにより，ユーザの普段のマウス操作をほとん

ど妨げることなく提案手法を実現する．100 ミリ秒経過以降に他方のボタンが押下された場合にも，両押しと見なされる．このとき，先に押下された方のボタンダウンイベントは送出されているので，例えば左ボタンを押下したままファイルをドラッグし，次に右ボタン押下を追加することで，手前のフォルダにあるファイルを奥のフォルダへ移動するといった操作が可能になる．同様に，ファイルを左ボタンでドラッグしただけの状態では手前へ移動するため，奥の隠れた位置にあるファイルを手前へ移動することも可能である．ただし，ここで述べた 100 ミリ秒以内という値は HSP の await 命令で設定したものであり，実際には処理の遅延によってより長い時間同時押しが認められる場合が生じうる．

5.1.2　カーソルを代替するウィンドウの生成と制御方法

提案手法と同等のカーソルの挙動を実現するため，カーソルの画像を表示したウィンドウによって本来のカーソルを代替する方法をとった．まず，OS が本来表示しているカーソルを，透明な画像に差し替えることで非表示にする．それに代わり，カーソルの画像を表示したウィンドウを表示する．このウィンドウはタイトルバーや縁のないものであり（HSP の bgscr 命令で生成），なおかつカーソル以外の部分を Win32API の SetWindowLong 関数と SetLayeredWindowAttributes 関数によって透過設定したものである．このようにして，本来のカーソルと同様の外観を持つウィンドウを生成する（図 7）．以降，このウィンドウをカーソルウィンドウと呼び，本来のカーソルと区別する．カーソルウィンドウを，Win32API の SetWindowPos 命令によって本来のカーソルと常に同じ座標に移動し，さらにマウスイベントを通過する設定（5.2.2 節で後述）を施すことで，ユーザはこれを本来のカーソルと同等に扱うことができる．

カーソルウィンドウは，オーバーラップウィンドウ方式に則って他のウィンドウと同様に Z オーダを持つ．

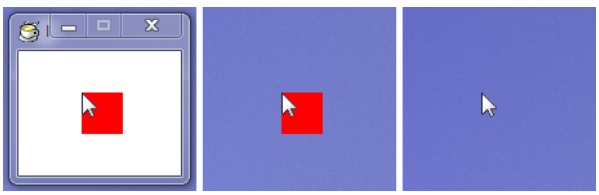


図 7　カーソルウィンドウの生成方法．（左）通常のウィンドウにカーソルの画像を表示した状態．（中）縁なしウィンドウを使用した状態．（右）さらにカーソル周辺の単色部分を透過処理した状態．
Fig. 7　How to create a cursor-window.

この Z オーダは奥のウィンドウほど大きな値を持つ．カーソルウィンドウとその他のウィンドウの Z オーダを SetWindowPos 関数によって任意に設定することで，ウィンドウの奥へ潜りこんだカーソルが表現できる．また，両押し中はカーソルサイズを 15%縮小し，ウィンドウにカーソルを押し付けているような表示にする．さらに，カーソルウィンドウが奥のレイヤに移動したタイミングで，カーソルを一瞬だけさらに縮小し，その際に金属音を鳴らすことで，奥のウィンドウに衝突したことを表現する．手前のレイヤに移動した場合は拡大処理を行い，同様に金属音を鳴らす．

カーソルウィンドウの Z オーダが大きくなるほどカーソルサイズを小さくし，ユーザがカーソルの奥行きを認識しやすくする方法も考えられる．しかし，ウィンドウの数が大きくなるにしたがい，カーソルウィンドウが奥にあるときに小さくなりすぎ，視認性に問題が生じる．逆に，Z オーダが最大のときのカーソルサイズを基準にして手前に来るほど拡大する方法では，カーソルウィンドウが手前にあるときそれに覆い隠される範囲が広くなる問題が生じる．よって，奥行き方向を提示するためのカーソルウィンドウの拡大・縮小処理は行わないこととした．

5.2　ウィンドウ制御部

5.2.1　ウィンドウを最前面化することによるカーソルウィンドウの奥移動表現

Win32API の FindWindow，GetWindowRect，GetWindowLong 関数によって，起動中のウィンドウのハンドル，座標とサイズ，ウィンドウ情報を取得する．これらを Z オーダの昇順に配列へ格納して管理する．両押しすると，カーソルウィンドウ直下のウィンドウよりも Z オーダの小さいウィンドウに半透化処理，マウスイベント通過処理，最前面化処理をする．これらの処理によって，カーソルウィンドウは最前面化されたウィンドウよりも奥のレイヤへ移動したような外観になる．カーソルウィンドウ直下のウィンドウは，配列 0 番のウィンドウから順に座標とサイズを調べ，カーソルの座標がそのウィンドウの範囲内に含まれるかを判定することで特定する．

これはすなわち，デスクトップを含む n 枚のウィンドウがあるとき，カーソル直下のウィンドウが k 番 ($0 < k < n$) であれば，0〜k-1 番のウィンドウに半透化処理，マウスイベント通過処理，最前面化処理が適用される．ただしカーソルウィンドウは除外する．図 8 はこの処理を実際の画面を伴って説明したものである．両押しを維持したままカーソルを移動させ，カーソルウィンドウが k 番ウィンドウの範囲から外れると，k+1 番目から同様にカーソルウィンドウ直下のウィンドウを調べ，最前面化などの処理をすることでカーソ

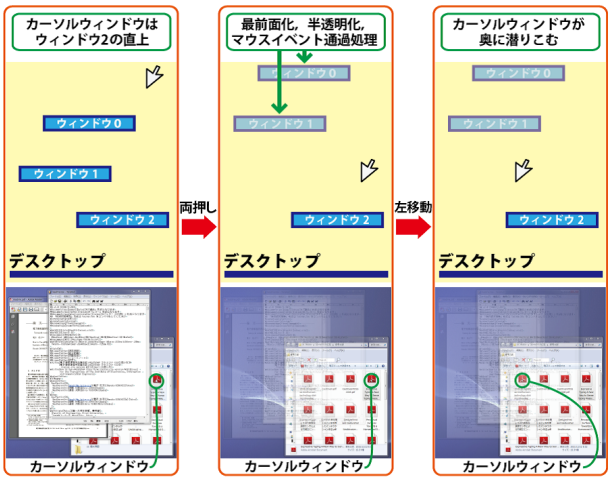


図 8　カーソルウィンドウが奥へ潜りこむ様子．下側のスクリーンショットを 90 度回転した位置から見たものが上側の模式図．(左) カーソルウィンドウはウィンドウ 2 の直上にある．(中) 両押しするとウィンドウ 0 と 1 が最前面化される．(右) 両押しを維持してカーソルウィンドウを左に動かすと，元々隠れていた部分に辿り着く．

Fig. 8　Process of cursor-window going to underneath windows.

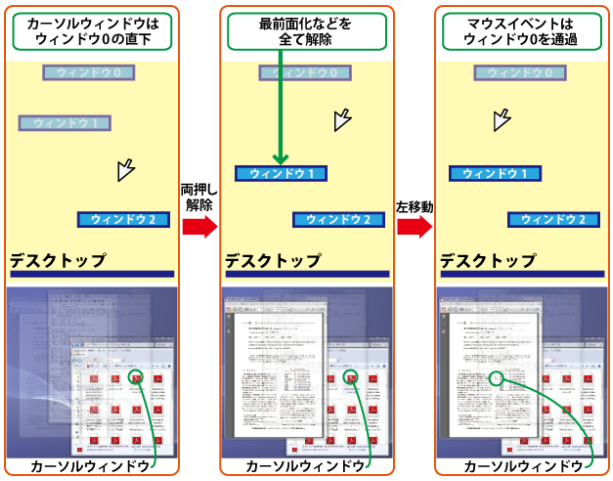


図 9　カーソルウィンドウが手前へ移動する様子．(左) 両押し中．カーソルウィンドウ直上にはウィンドウ 0 がある．(中) 両押しが解除されると，ウィンドウ 1 の最前面化が解除される．(右) カーソルウィンドウを左に動かすと，カーソルウィンドウ直下はウィンドウ 1 になる．ウィンドウ 0 はマウスイベントを透過するので，ウィンドウ 1 を操作できる．

Fig. 9　Process of cursor-window going to forward.

ルウィンドウがさらに奥へ移動したように表示される．こうしてカーソルウィンドウは起動中のウィンドウに衝突しつつ段々と奥のレイヤへ移動していく．

両押し中でないとき，つまり左右ボタンの少なくとも一方が離されているときには，カーソルウィンドウは手前へ移動する．すなわち両押しを解除すると，図 9 のようにカーソルウィンドウの直上にあるウィンドウの 1 つ奥のレイヤにカーソルウィンドウが移動する．カーソルウィンドウ直上のウィンドウは，カーソルウィンドウが k 番ウィンドウ上のレイヤにあるときに，0〜k-1 番目のウィンドウの座標とサイズを降順に調べていけば特定できる．そしてカーソルウィンドウ直上のウィンドウが j 番目 (-1 ≦ j < k であり，j= -1 はカーソルウィンドウ直上にウィンドウがない場合) だったとき，j+1〜k-1 番のウィンドウの最前面化処理などを解除し，SetWindowPos 命令によってカーソルウィンドウよりも Z オーダを大きくすることで，カーソルウィンドウは手前に移動したように表示される．なお，カーソルより手前のウィンドウを完全透過にする方法も考えられるが，平岡の研究 [4] では，ウィンドウをタイトルバーのみの表示にすると見失うと指摘されており，概形を把握可能な半透過にとどめた．

5.2.2　ウィンドウのマウスイベント透過処理による奥への操作

カーソルウィンドウが k 番ウィンドウ上のレイヤにあり，両押しを解除しているとき，マウスイベントはカーソルウィンドウ直下のウィンドウに送出される．これは，カーソルウィンドウよりも手前のウィンドウにはマウスイベント通過処理が加えられており，マウスイベントが 0〜k-1 番ウィンドウを通過するためである（カーソルウィンドウ自身もマウスイベント通過処理されている）．また，本来はマウスイベントを受け取ったウィンドウはアクティブになり最前面に移動するが，0〜k-1 番ウィンドウは最前面化処理されているため，ウィンドウの Z オーダは変更されない．

以上の処理によって，カーソルがウィンドウの奥へ潜りこんだような表現や，ウィンドウの奥での操作を実現している．このようにして，両押し中はカーソルがウィンドウに衝突するまで奥へ移動する/両押しでないときはウィンドウに衝突するまで手前へ移動する，という操作手法が可能になる．

5.3　補助的機能

本研究では元々対象としていなかった状況においても提案手法を利用可能にするため，適用範囲を広げるための追加機能を実装した．従来の提案手法で対応できないのは，ウィンドウが最大化している場合と，同座標かつ同サイズのウィンドウがある場合である．図 10 のように最大化ウィンドウがある場合には，カーソルウィンドウは奥へ潜りこむ余地がなく，提案手法を使用できない．また，図 11 のように同座標に同サイズのウィンドウがある場合にも，カーソルウィンドウはその間に潜りこむことができない．

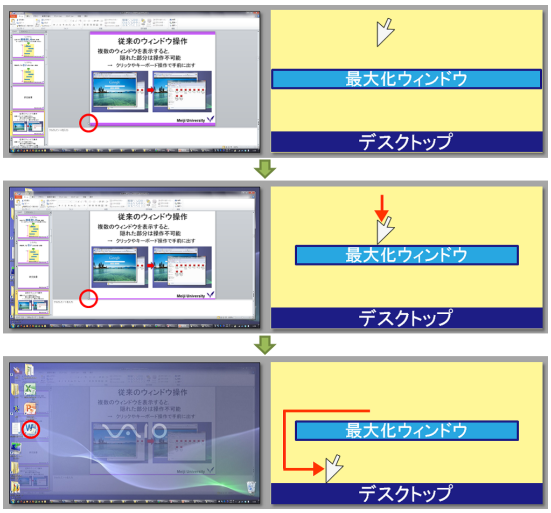


図 10　最大化ウィンドウの縮小機能．両押し中は上下左右がわずかに縮小することでカーソルが奥へ潜りこめるようになる．

Fig. 10　A maximized window(s) becomes slightly smaller when both mouse buttons are pressed.

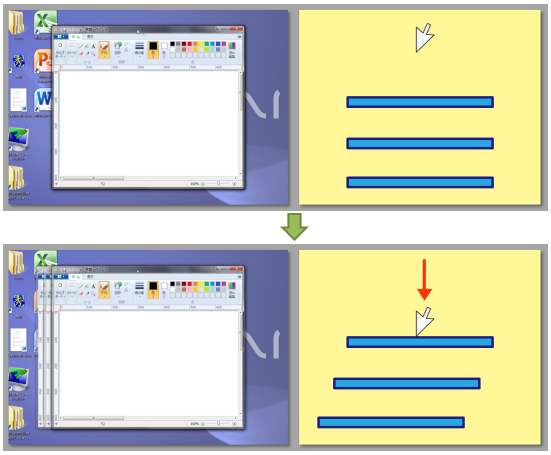


図 11　同座標・同サイズのウィンドウがある状態．両押し中は奥のウィンドウがずれることでカーソルが奥へ潜りこめるようになる．

Fig. 11　The windows hidden by a front window (same size and same position) move to produce a gap when both buttons are pressed so that the cursor moves to their layers.

これらの状況にも対応可能にするため，両押し中は最大化ウィンドウの上下左右を 50px ずつ縮小する機能を実装した．これによって奥へ潜りこむための隙間ができる（図 10）．また，同座標・同サイズのウィンドウに対しては，両押しした時点で次のように処理する．まず 0 番ウィンドウのサイズと位置を決定する（つまり移動しない）．次に k 番ウィンドウ $(1 \leqq k < n)$ の座標とサイズを j 番ウィンドウ $(0 \leqq j < k)$ と比較し，もし同座標・同サイズであれば k 番ウィンドウを 50px 移動する．移動方向は左・上・右・下の順に探索し，ウィンドウが画面外へ出ない方向へ移動する．言い換えれば，全てのウィンドウを手前から順に調べ，もしそれよりも手前のウィンドウで同座標・同サイズのものがあれば僅かに移動させる，という処理である．このようにして Z オーダの昇順にウィンドウの座標を決定していくことで，全てのウィンドウがその 1 つ前後のウィンドウと確実にずれ，カーソルウィンドウを潜りこませることが可能になる（図 11）．

6.　実験

提案手法の有効性を確かめるために，タスクの実行に費やす時間を計測し，従来手法と比較する実験を行った．実験で比較したのは，提案手法での操作，マウス単独操作，マウス・キーボード併用操作である．マウス単独操作は，ウィンドウのドラッグやクリック，スナップやタスクバー操作など，Windows7 においてマウスのみで可能な操作である．マウス・キーボード併用操作は，マウス操作に加えてキーボードショートカット（Alt+Tab による最前面のウィンドウ切り替えや，Win+D によるデスクトップ表示など）を使用した操作である．

6.1　実験手順

異なるフォルダ間でのファイル移動を行うタスクを被験者に与え，操作を完了するまでの時間を計測した．被験者は情報系の大学学部および大学院に所属し，PC 操作に習熟した学生 10 名である．操作手法の順序は被験者毎に変更し，それぞれで 5 回ずつタスクを行った．5 回の平均操作時間を各被験者の記録とする．実験ではあらゆるマウス操作を認め，キーボードでも禁止した操作はない．ただし，マウス・キーボード併用操作では，マウス単独操作と実験内容を区別するため，必ず 1 回以上のキーボードショートカットを使用するものとした．被験者には可能な限り速くタスクを行うように伝えた．

実験開始時のフォルダの配置を図 12 に，移動前後のファイル配置のイメージを図 13 に示す．この配置は，手前のウィンドウによって奥の操作対象オブジェクトが覆い隠されており，従来手法では操作量が多くなる状況に相当する．また，実験開始時にカーソルは最前面のウィンドウの中央に置かれる．なお，ファイルを移動する順序は被験者の任意であり，実験終了時のウィンドウの配置は問わず，ウィンドウを閉じる操作も認めた．

各操作手法での実験を行う前に，実験と同じファイル・ウィンドウの配置で 15 分間を上限に練習を行わせた．これは，実験を開始してからのファイルを移動する順序やウィンドウを移動する位置などの試行錯誤を

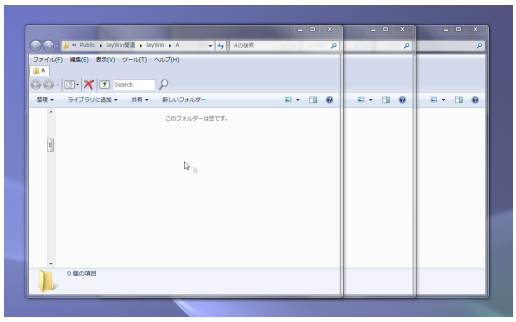


図 12　実験開始時のウィンドウの配置．最前面のフォルダ A にはファイルがない．奥のフォルダ B，C とデスクトップには，手前のフォルダに隠れた位置にファイルが置かれている．
Fig. 12　Initial window layout.

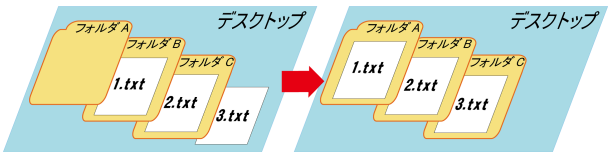


図 13　移動前（左）と移動後（右）のファイルの配置．
Fig. 13　File locations before the task and after the task.

防ぎ，操作をスムーズに行った場合の時間を比較するためである．さらに，実験時に誤った操作をした場合にも，同様の理由でタスクを再試行した．なお，提案手法による実験では，カーソルウィンドウより手前のウィンドウの透明度を，筆者らが実際にタスクを行って操作しやすいと感じた 23%に固定した（実際にはユーザによって調整可能である）．

使用した PC は SONY 社製 VPCF11AGJ(2.53GHz, 4GB, ノート型)，キーボードはノート PC 標準，マウスは BUFFALO 社製 BSMOU05M(1000dpi)，ディスプレイはノート PC 標準 (1920 × 1080px, 16.4inch)，OS は Windows7 Pro である．Windows7 のエアロ及びスナップ機能は有効，タスクバーのアイコンは小で結合なし，デスクトップ上及びフォルダ内のアイコンサイズは大，OS におけるカーソル速度の設定値は最大（C-D 比 2000dots/inch）であり，コントロールパネルの「ポインターの精度を高める [1]」のチェックを ON にしている．被験者には椅子の高さ調節，ディスプレイの角度調節，ノート PC の位置調整を認め，実験時に快適な操作ができるよう配慮した．また，マウス・キーボード併用操作の練習時にはショートカットコマンド一欄を提示した．

1: http://msdn.microsoft.com/ja-jp/library/windows/hardware/gg463319.aspx（2012 年 11 月 18 日閲覧）

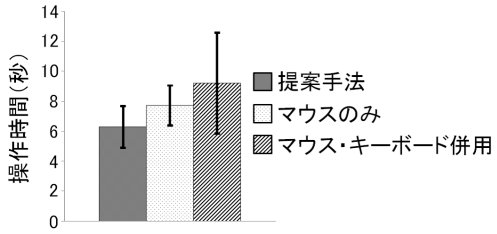


図 14　平均操作時間．
Fig. 14　Average times taken to complete the task by the subjects.

表 1　アンケート項目と回答の平均値．
Table 1　Questionnaire items and average scores of the answers.

質問項目	回答
左右ボタン両押しによって潜る，という操作のしやすさ	3.4
非両押し時は手前に移動する操作のしやすさ	3.5
潜った先での (ウィンドウの裏側での) マウスカーソル操作のしやすさ	3.7
目標のウィンドウへの辿り着きやすさ（意図したレイヤでカーソルを止められるか）	4.0
音によるレイヤ移動の分かりやすさ	4.4
マウスカーソルの拡大・縮小によるレイヤ移動の分かりやすさ	2.8
潜って操作するという機能の便利さをどのくらい感じたか	4.4
提案手法での，操作したいファイルへの辿り着きやすさ	4.3
マウス単独操作の場合の，操作したいファイルへの辿り着きやすさ	2.4
マウス・キーボード併用操作の場合の，操作したいファイルへの辿り着きやすさ	2.9

実験後に 5 段階評価のアンケート（5 が最高値）を実施し，本システムのユーザビリティについて尋ねた．

6.2　結果

図 14 に操作時間の平均値を示す．10 名のうち 1 名はマウス・キーボード併用操作が最短であったが，他 9 名は提案手法が最短であった．平均操作時間を 1%水準の被験者毎対応あり分散分析（反復測定）した結果，3 種の操作手法間に有意差が見られた（$F_{2,18} = 11.922$）．さらに，操作手法をペア毎に多重比較したところ，提案操作とマウス単独操作間で $p < .001$，提案手法とマウス・キーボード併用操作間で $p < .01$ の有意差が見られた．

アンケートの質問項目と回答結果を表 1 に示す．この他に，タスクを最もこなしやすかった操作を選ぶ項目では，提案手法が 8 名，マウス単独及びマウス・キーボード併用操作が各 1 名であった．さらに，「今後，提案システムを使いたいと思うか」という質問に対して被験者全員が「そう思う」と回答したことから，提案手法が好意的に受け止められていると考えている．タスクの再試行平均回数は，提案手法で 3.5 回，マウス単独で 2.6 回，マウス・キーボード併用で 3.9 回であった．ただし，被験者の誤操作かシステム側の問題かを区別できない場合も全てカウントしている（前者は両押しミスやドロップ先のフォルダ間違い，後者は処理遅延による両押し認識ミスがある）．

6.3　考察

6.3.1　操作時間の考察

操作時間について，3 手法のうち提案手法での操作時間が最も短いことから，提案手法は多くの被験者にとって 15 分以内に習得でき，かつ実験環境では従来

手法での操作よりも作業を効率化可能であるといえる．しかし，被験者のうち 1 名はマウス・キーボード併用操作が最速であり，アンケートでも併用操作が最も操作しやすかったと回答している．その被験者は普段からキーボードをメインに PC 操作をしており，ショートカットコマンドの熟達者であるといえる．一方で，それ以外の 9 名も普段から PC 操作は行なっており，キーボードショートカットを問題なく使えることを確認している．したがって，キーボード操作に特に秀でているユーザにとっては，提案手法は従来手法を超えるほど操作効率を向上させるものではないといえる．逆に，それ以外のユーザにとっては，普段からキーボードショートカットを使いこなしているとしても，提案手法の方が速く作業を行える場面があるといえる．キーボードショートカットに慣れていない一般的なユーザでは，マウス・キーボード併用での操作時間はさらに増長する可能性もあり，提案手法の貢献度はより高くなる可能性もあると考える．

6.3.2　アンケート結果の考察

表 1 によると，両押しによって潜る，両押しでないときには手前に移動するという操作はそれぞれ 5 段階中 3.4，3.5 と，高いとはいえない評価結果だった．両押しは普段は行わない操作であり，また試行時にミスをしてしまうこともあって，操作しやすいとは感じられなかったのだと考えられる．自由記述によるアンケートでは，初めての操作方法であり慣れなかったとする意見と，問題なく操作できるようになったとする意見に分かれ，両押しによって潜りこむ機能を発動することが最適とはいえない結果となった．問題なく操作できたと答えた被験者は，「右ボタンを押しながらホイールを前後回転させる操作を（ウェブブラウザの拡張機能で）使っているが，それと似ていて案外早く慣れた」という感想を口頭で述べた．このように普段から 2 本の指を同時に使うマウス操作をしていることが習熟度にも影響すると見られる．逆に慣れなかったと回答した被験者の中には，「普段はやらない操作で，実験が終わるまでずっと違和感があった」とする者もいた．10 名中 9 名の被験者が従来手法よりも早くタスクを終えることができたものの，提案手法によって操作時間が短縮されたという結果と，操作方法の良し悪しとは別個の問題であり，より適切な操作方法を採用すべきであると考える．

ウィンドウの裏側でのカーソル操作のしやすさは平均 3.7 であり，これも高いとはいえない結果になった．実験では不透明度を 23%としたが，3 枚のウィンドウの奥にカーソルが潜りこんだときにはカーソルの視認性が落ちてしまう（図 15）．ウィンドウの枚数が多くなるにつれてこの影響は大きくなり，操作が困難になってしまう問題がある．不透明度をユーザがカスタマイズすることである程度は解消できるものの，枚数が極端に多くなった場合には対処できないのが現状である．

図 15　実験環境で，カーソルがデスクトップに潜りこんでファイルをポインティングしている様子．
Fig. 15　A cursor behind the three windows.

目標のウィンドウへの辿り着きやすさについては平均 4.0 と低くはないものの，これは今回の実験では問題なく全てのレイヤに移動できるウィンドウの配置であったことが理由であると考えられる．実際の作業時にはより複雑なウィンドウの配置も存在するが，そのような環境での調査も今後は実施したい．

提案手法のうち，特に好意的に受け入れられたのは音によるレイヤ移動の分かりやすさであった．カーソルが奥行き方向へ移動するというユーザにとって馴染みのない操作が，聴覚的効果によって理解しやすくなったものと考えられる．同様に，潜りこむ機能自体の便利さについても高い評価が得られた．これは裏を返せば，従来手法による操作が，提案手法と比較してより不便に感じられたと理解することもできる．

6.3.3　実験全体に関する考察

操作時間は多くの被験者が提案手法で最短となったが，同時にキーボード操作に秀でたユーザにとっては大きな打開策ではないこともわかった．今回は操作時間に限定してデータを取得したため，マウス・キーボード併用操作が最速だった被験者が他の 9 名とは異なる特殊な操作をしていたのか，あるいは純粋にキータイプが速かったのかなどの判断はできないが，今後はその点を分析する必要もあると考えている．

タスクにおけるウィンドウの初期配置は，「同サイズのウィンドウを同一方向にずらして配置する」としたが，これは提案手法で解決したい状況として 4 章で例示した図 4，5，6 を単純化したものである．また，タスク終了時のウィンドウの状態は不問にするという条件だったので，ウィンドウを閉じたり，スナップ機能で画面の左右に配置したりした被験者がいたが，このままではその後の作業に支障をきたすと思われるウィ

ンドウの配置になっていることがあった．実際の PC 利用時にはより複雑なウィンドウやアイコンの配置が起こり，また操作終了時に「このウィンドウは開いたままにする必要がある」などという制約が発生することもある．こういったより現実的な状況設定で実験を行う場合には，複数のウィンドウが重なりあう条件下でウィンドウやアイコンがランダムに配置されたり，「ファイルの移動を終えたら，ウィンドウが完全に元の状態になるようにすること」などの条件を課したタスクを設定する必要があると考えられる．しかしながら今回の実験条件は，ファイルエクスプローラを複数起動すると自動的にカスケードするように，「一定のずれ幅で規則的に開いたフォルダ間でファイルを移動する」という現実の作業環境で起こりうる状況をある程度取り入れたものである．またアンケート結果からは，従来手法では困難だった操作が可能になることの効果を実感させられたことや，カーソルがレイヤ移動する際のフィードバックに関する好意的意見を得られたことから，一定の評価ができたと考えている．

7. 議論と今後の課題

本論文では主にウィンドウ操作に要する手間や時間について述べてきたが，これらの他にカーソルが 3 次元方向に移動することに対する認知的負荷の問題がある．認知的負荷とは，脳の注意力には限度があり，それを様々な対象に配分しているとするものである．従来手法では 2 次元方向にのみ移動していたカーソルを 3 次元方向にも移動可能にすることで，提案手法ではカーソルやウィンドウの Z オーダを把握しつつ操作しなければならなくなった．つまり，提案手法では従来手法よりも認知的負荷が高いことになる．従来手法であってもポインティング以外の作業をしているとカーソルを見失ってしまう問題がよく生じるが，提案手法ではさらに奥行き方向の位置まで把握しなければならない．この認知的負荷を下げるために，ウィンドウとカーソルの位置するレイヤを提示するビューワを用いる解決策もあるが，意識を向けなければならない対象が増えることになり，屋上屋を架すことになるであろう．したがって，「デスクトップのアイコンをダブルクリックしたらすぐに手前に戻す」といった奥のレイヤへの一時的なアクセスに用い，ユーザ自身が利用場面を適切に選ぶようになるのが望ましいが，そのような習熟は可能なのか，長期運用を通して調査したい．

潜りこむ機能をマウスボタンの両押しによって発動させる設計にしたのは，広く使用されているマウスで利用でき，かつ従来の操作と競合しないようなトリガを設定するためであった．だが近年サイドボタンを備えたマウスも広く用いられるようになったことから，

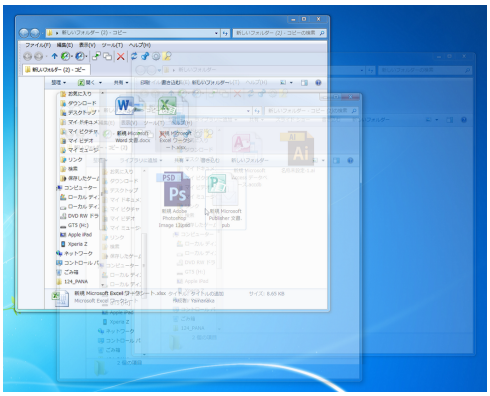


図 16　アイコンが配置されたレイヤの判別が難しい場面．
Fig. 16　It is difficult to detect the layer where the icons are located.

これをトリガにして潜りこむ機能を発動させることも検討したい．サイドボタンとはマウスの側面に備わっているボタンであり，主に親指で押しこむ．ユーザが任意の機能を割り当てることができ，潜りこむためのトリガとしても問題なく操作できると思われる．また現在のシステムでは，ファイルのドラッグ操作と潜りこむ機能を併用するには，左ボタンを押下してドラッグを開始し，次いで右ボタンを押下して両押しにする必要があり，押す手順を認識しなければならないためやや煩雑である．サイドボタンを使用すれば，ドラッグやクリック操作とは分離したボタンで潜りこむことができるため，こういった問題も解消することができる．これらの改善案の採用はマウスのサイドボタンが一般的か否かの議論にもよるが，不透明度や最大化ウィンドウの縮小サイズなどと同様に，ユーザがカスタマイズ可能な項目に含めることも可能である．

カーソルよりも手前のウィンドウを半透明化するシステムデザインは第 5.2.1 項のおわりに述べたようにウィンドウの概形を把握可能にするためだが，奥にあるカーソルの視認性問題に加え，オブジェクトが配置されているレイヤを把握しづらい問題がある．これは文献 [5] でも述べられているが，ウィンドウとそれに含まれるファイルアイコンなどのオブジェクトを画一的に半透明化すると，図 16 のように手前から何番目のウィンドウにどのアイコンが配置されているか一見判別しづらい．ゆえに目的のオブジェクトに辿り着きにくくなり，提案手法の操作性を下げる懸念がある．半透明化ではなく，ウィンドウの外周線のみの表示にし，手前のウィンドウの概形を把握可能にしつつ奥の視認性を向上させる解決策も考えられる．また，カーソルより手前にあるウィンドウの枚数に対する適切な不透明度が求められれば，現状の半透明化であっても問題なく操作できる可能性があるため，これについて

も調査したい．

非公式にではあるが，実験から数日経て「実験をしてから普段のウィンドウ操作が面倒に感じるようになった」という旨の感想が被験者から得られた．つまり，実験を行うまではクリックや移動，サイズ変更などのウィンドウ操作を行うのは当然だと受け止めており，それらを煩雑とは思わなかったのだと理解できる．提案手法によって従来手法の本来行いたい操作に付随して求められるウィンドウ操作の面倒さを発見し，普段の PC 操作を通して筆者らの提案手法の意義を感じるようになったのだと我々は考えている．一般的に使用されている GUI の不便な点に慣れてしまい，ウィンドウシステムの問題に気付かなかった，ということをユーザ自身が発見できたことは，定量化できないものの提案手法が持つ 1 つの意義であったと考える．

8.　関連研究

ウィンドウが重なることで発生する問題については様々な解決方法が試みられている．ウィンドウ切り替えの煩雑さを解決するために，加藤らはスライダやペダルを用いてウィンドウを手前から順に非表示にするぱらぱらウィンドウを提案した [6]．特殊な機器を使用せずマウスホイールを利用するバージョンも存在するが，その場合にはアプリケーションに対するホイール操作が制限されると考えられる．Ishak らは，ウィンドウを表示内容に応じて透過処理し，ウィンドウの奥を見えるようにすることでウィンドウ操作回数を軽減するシステムを開発した [5]．このシステムではウィンドウ内の白色部分を透過させているが，ウィンドウ全体に情報が表示されている場合には透過される箇所がなく，有効に機能しない場面が生じうる．Faure らは，マウスボタンを一定時間以上押してからドラッグすることで，任意のレイヤにあるウィンドウ（デスクトップ含む）を最前面に出すシステムを開発した [7]．これは同一レイヤにある複数のウィンドウ切り替えを一括して行うことで操作量の削減も実現しているが，特定のウィンドウを最前面のままに保ちたいといった要求には対応していない．小林らのシステムもこれと同様に，ドラッグ操作中にマウスホイールを回転させることで，カーソル直下のウィンドウを手前から順に非表示にするものである [8]．神原らのちらりウィンドウは，ジョイスティックを操作することで全てのウィンドウを移動させ，前面のウィンドウに隠された部分も閲覧可能にするシステムである [9]．久納らはこれと同様の操作を実世界での視点移動で実現するため，カメラを用いて首の動作をウィンドウ移動に反映させるシステムを提案した [10]．この 2 つのシステムは，最大化ウィンドウは画面外に出てしまうなどの不都合が生じるため，事前に問題が生じないようにウィンドウ配置をある程度工夫する必要がある．これに対し提案手法では，ウィンドウの配置やサイズがどのような場合であっても対応可能である．Dragicevic は，紙のようにウィンドウをめくることで，奥にあるウィンドウへのドラッグアンドドロップを可能にするシステム fold-and-drop を提案した [11]．これはめくる操作をドラッグ中に限定しているが，本論文の提案手法は奥のフォルダを探索したり，デスクトップ上のショートカットを起動したりと，より多岐にわたる場面での利用が可能である．

ウィンドウとカーソルの次元の相違を解決するために，3 次元デスクトップ環境を導入する研究もある．Robertson らの Task Gallery [12] は，仮想的な 3 次元空間表示を利用することでウィンドウの重なりをなくし，さらに配置を記憶しやすくしたシステムである．同様に Card らの Rooms [13] は，複数のデスクトップを切り替えられるようにすることで，一度に表示するウィンドウ数を減らし，重なりあいを軽減したシステムである．大内らは，3 次元デスクトップ環境においてマウスが 2 次元でしか操作できないことを問題として，3 次元入力が可能なデバイスを使用する手法を提案した [14]．これらは一般的なオーバーラップウィンドウ方式におけるウィンドウ切り替え手法を扱っていない．

本論文の提案手法と同様に，マウスカーソルに通常とは異なる挙動をさせ，それによってマウスの操作量を低減させるシステムが存在する．Grossman らの Bubble Cursor [15] は，カーソルを円形に表示し，ポインティングを一点ではなく広範囲の円にすることでカーソルの移動距離を短縮するシステムである．同じくマウスの移動量を低減するシステムに，カーソルを目標のオブジェクトまでワープさせる Delphian Desktop [16] や，目標のオブジェクトをカーソル付近まで引き寄せる Drag-and-Pop [17] がある．Kobayashi らは，複数のカーソルを表示することで移動距離を短縮させる Ninja Cursors [18] を提案している．また中村らは 2 つのマウスを用いてウィンドウを操作する環境を提案している [19]．この 2 つは，目標のオブジェクトに近い方のカーソルを利用することでマウス移動量の軽減が可能である．

9.　おわりに

本論文では，オーバーラップウィンドウ方式におけるウィンドウ操作の問題について述べ，カーソルをウィンドウの奥へと潜りこませる操作手法による解決を提案した．これによって本来目的とする操作に付随するウィンドウ操作量を軽減できるようにした．評価結果から，実験環境においては従来手法よりタスクを速く

完了させられることを明らかにした．ただし，本論文では実験のために用意したタスクを完了するまでの操作時間を比較したものであり，実際の利用状況ではより複雑なウィンドウやオブジェクトの配置もありうる．そこで，様々なウィンドウの枚数，サイズ，配置，ウィンドウ同士のずれ幅などで実験を行い，提案手法の有効範囲を求めたい．また，普段の作業中において提案手法を長期利用した場合の習熟状況も調査したい．なお，本システムの実際の動作場面については筆者らのウェブページに掲載した動画を参照されたい[20]．

参考文献

[1] 山中祥太, 宮下芳明: 重なりあったウィンドウ間を移動可能なマウスカーソル操作手法の提案; 情報処理学会研究報告, Vol.2011-HCI-144，No.13，pp.1-8 (2011).

[2] 山中祥太, 宮下芳明: スイッチバックカーソル : 重なりあったウィンドウ間を移動可能なマウスカーソル操作手法; 第 19 回インタラクティブシステムとソフトウェアに関するワークショップ論文集, pp.66-71 (2011).

[3] Tashman, C. and Edwards, W.K.: WindowScape: Lessons learned from a task-centric window manager; *ACM Transactions on Computer-Human Interaction*, Vol.19, No.1, Article 8 (2012).

[4] 平岡茂夫: ぼかしの効果を活用したインターフェースデザイン; 福岡工業大学情報科学研究所所報, Vol.12, pp.7-13 (2001).

[5] Ishak, E.W., Feiner, S.K.: Interacting with hidden content using content-aware free-space transparency; In *Proc. of UIST '04*, pp.189-192 (2004).

[6] 加藤直樹, 小國健: ぱらぱらウィンドウ : ウィンドウの切り替えを容易にするインタフェース; ヒューマンインタフェース学会論文誌, Vol.7, No.2, pp.81-88 (2005).

[7] Faure, G., Chapuis, O., Roussel, N.: Power tools for copying and moving: useful stuff for your desktop; In *Proc. of CHI '09*, pp.1675-1678 (2009).

[8] 小林正朋, 五十嵐健夫: 活用 : マウスホイール; インタラクション 2005 論文集, pp.175-176 (2005).

[9] 神原啓介, 安村通晃: ちらりウィンドウ : 隠れたウィンドウを覗き見る; インタラクション 2004 論文集, pp.47-48 (2004).

[10] 久納章寛, 岡本壮平, 武藤直美, 中島誠, 伊藤哲郎: 層構造の作業環境におけるユーザ意図の把握; FIT2002 情報科学技術フォーラム情報技術レターズ, pp.199-200 (2002).

[11] Dragicevic, P.: Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows; In *Proc. of UIST '04*, pp.193-196 (2004).

[12] Robertson, G., Dantzich, M.v., Robbins, D., Czerwinski, M., Hinckley, K., Risden, K., Thiel, D., Gorokhovsky, V.: The task gallery: a 3D window manager; In *Proc. of CHI '00*, pp.494-501 (2000).

[13] Card, S.K., Henderson, A.H. Jr.: A multiple, virtual-workspace interface to support user task switching; In *Proc. of CHI '87*, pp.53-59 (1987).

[14] 大内勇佑, 西野浩明, 宇津宮孝一, 賀川経夫: 触感提示機能を有する 3 次元デスクトップ環境の開発; 火の国情報シンポジウム 2011 (2011).

[15] Grossman, T., Balakrishnan, R.: The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area; In *Proc. of CHI '05*, pp.281-290 (2005).

[16] Asano, T., Sharlin, E., Kitamura, Y., Takashima, K., Kishino, F.: Predictive interaction using the delphian desktop; In *Proc. of UIST '05*, pp.133-141 (2005).

[17] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., Zierlinger, A.: Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems; In *Proc. of Interact '03*, pp.57-64 (2003).

[18] Kobayashi, M., Igarashi, T.: Ninja cursors: using multiple cursors to assist target acquisition on large screens; In *Proc. of CHI '08*, pp.949-958 (2008).

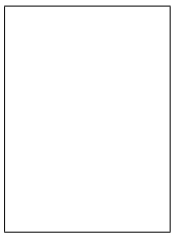
[19] 中村聡史, 塚本昌彦, 西尾章治郎: 2 つのマウスを用いたウィンドウ操作機構の設計と実装; 情報処理学会研究報告, ヒューマンインタフェース研究会報告, Vol.99, No.35，pp.1-6 (1999).

[20] スイッチバックカーソル : 重なりあったウィンドウ間を移動可能なマウスカーソル操作手法 [動画]; http://miyashita.com/2012/04/switchbackcursor-j.html (2012 年 11 月 18 日参照).

（2012 年 11 月 20 日受付，2013 年 4 月 20 日再受付）

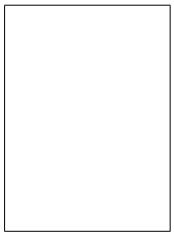
著者紹介

山中　祥太



2013 年明治大学大学院理工学研究科新領域創造専攻ディジタルコンテンツ系博士前期課程修了．2013 年より同大学博士後期課程に在籍するとともに理工学部助手，現在に至る．ユーザインタフェース研究，特にポインティングインタフェース研究に興味を持つ．

宮下　芳明



千葉大学工業部卒業（画像工学），富山大学大学院で音楽教育（作曲）を専攻，北陸先端科学技術大学院大学にて博士号（知識科学）取得，優秀修了者賞．2007 年度より明治大学理工学部に着任．2009 年度より准教授．2013 年より同大学総合数理学部先端メディアサイエンス学科所属，現在に至る．ヒューマンインターフェース学会，日本ソフトウェア科学会，情報処理学会，ACM 各会員．