原著論文

Vol.7 No.2, 2005

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

加藤 直樹 *1 小國 健 *2

PARA-PARA Window: An easier window switching interface

Naoki Kato*1 and Tsuyoshi Oguni*2

Abstract – **This paper proposes a new user interface to solve irritating trouble when switching windows on overlapped multi-window system. By using a 1-dimensional input device, a user can directs thickness of unnecessary windows from the top, and makes them hidden or translucent. Software that can be used the proposed user interface with commercial devices for DirectX and wheel mouse was developed and evaluation experiments was conducted. As a result of the evaluation experiments, it is shown that the operation times to locate back windows frontward and to move icons from one window to another could be shortened when using the new interface.**

Keywords : **input device, multi window, overlapped window, GUI**

1. はじめに

マルチウィンドウシステムの出現によって，一つの物理画面上に複数の仮想画面を表示することが可能になり，たとえば，複数のアプリケーションソフトウェアが表示する情報を同時に閲覧することができるようになった．複数のウィンドウを表示する方式としては，すべてのウィンドウに重なりがないように表示するタイルウィンドウ方式と，重なりを許すオーバーラップウィンドウ方式がある．タイルウィンドウ方式は，多くのウィンドウを表示すると，一つ一つのウィンドウの大きさが小さくなってしまう問題点があり，現在主流なのはオーバーラップウィンドウ方式である．

ところが，このオーバーラップウィンドウ方式にも，ウィンドウを重ねて表示することを許していることから，多くのウィンドウを表示すると，その一部または全体が見えないウィンドウが出てくるという問題点がある．奥行き方向において下側に位置するウィンドウ全体を見えるようにするためには，ウィンドウの一部をポインティングデバイスでクリックするのが一般的であるが，ウィンドウ全体が見えていない場合には，それを隠しているウィンドウを移動するなどの操作をして，一部を見えるようにする必要がある．

文章書きなど一つのウィンドウだけを長時間利用している場合は問題ないが，複数のアプリケーションソフトウェアを駆使した作業やファイルの整理を行うときには，ウィンドウの切り替えを頻繁に行わなければならない．しかし，現状では，直観的に，容易にウィンドウの切り替えを行うための操作方法が確立しているとは言えず，ユーザは日々の作業でストレスを溜めている．

そこで，本稿では，既存のウィンドウの切り替え手法が直観的でない理由を明確にし，そして，この問題を解決する操作方法を提案する [1], [2]．具体的には，1次元の変移を入力することができる入力デバイスから変移を入力すると，最上位のウィンドウから変移に応じた枚数のウィンドウが非表示または半透明化され，下に位置するウィンドウの閲覧と操作ができるようになるユーザインタフェースを提案する．また，評価実験を通して，その有効性を検証する．

2. ウィンドウ切り替えインタフェースの現状

2.1 基本インタフェースの問題点

はじめににも記したように，既存のウィンドウシステムで，ウィンドウを最上位にして，ウィンドウ全体が見えるようにするには，ウィンドウの一部をマウスなどのポインティングデバイスでクリックする．しかし，この操作方法だけでは，次のようなことが起こる．

- ウィンドウの一部や全体が隠れているウィンドウを一時的に参照したい場合，上に位置するウィンドウを一時的に動かしたり，最小（アイコン）化したりしなければならない．さらに元の作業に戻るときには，移動や最小化したウィンドウを元に戻す操作が必要となる．
- 二つのウィンドウを交互に閲覧したり，二つのウィンドウ間でドラッグ＆ドロップなどを行ったりしたい場合，両方のウィンドウが見えるように，両方の

*1 : 東京学芸大学
*2 : NTT データ
*1 : Tokyo Gakugei University
*2 : NTT Data Corporation

ウィンドウの大きさや位置を調整しなければならない．

多くのウィンドウシステムでは，ウィンドウの一覧を表示し，そこから選択したウィンドウを最上位に表示する機能が用意されている．たとえば，米国マイクロソフト社の Microsoft Windows では，最小化したウィンドウがボタンとして格納されているタスクバーが常に表示されていて，そのボタンを押すことで対応するウィンドウを最上位に表示することができる．また，Alt キーを押しながら Tab キーを押すことで，すべてのウィンドウがアイコンとして表示されたダイアログが開き，そのまま Tab キーを繰り返し押すことで最上位に表示するウィンドウを選ぶことができる．しかし，これらの機能では，どのボタンやアイコンが目的とするウィンドウに対応しているのかわからず，手当たり次第にウィンドウを表示しなければならないことがある．また，ウィンドウ間のドラッグ＆ドロップなどの操作に関しては，前者はウィンドウの大きさや位置の調整が必要であり，後者はマウスを操作しながら反対の手で複数のキーを操らなければならないという問題点も生じる．

2.2 既存の解決ツール

こうしたウィンドウ切り替えに伴う問題を解決しようと，数々のツールがフリーソフトなどの形態で流通している．次にその例を示す．

- 下に位置し隠れているウィンドウを見えるようにするために，ウィンドウ上部のタイトルバーを右クリックすると，そのウィンドウを最下位に送り込むツール（たとえば枠ピタ [3]）
- 仮想的にデスクトップを増やしてウィンドウの切り替えの負担を減らそうとするツール（たとえばスイッチ XP [4]）
- デスクトップに簡単にアクセスできるようにするため，画面左端に置かれたバーを動かすと，動かした部分にはデスクトップが表示されるツール [5]
- 使用していないウィンドウが占める表示領域を小さくするために，ウィンドウを立体的に表示するツール [6]

こうしたツールが便利な場合もあるが，必ずしも使い勝手が良いとは言えない．操作体系・画面表示が複雑化し，屋上屋を重ねた感は否めない．

2.3 関 連 研 究

ウィンドウの切り替えを扱っている研究として，タスクの切り替えを容易にするユーザインタフェースの研究がある．Rooms [7] は複数の仮想的なデスクトップを切り替えられるようにすることで，一度に多くのウィンドウをデスクトップ上に表示せずに済むようにした．先のスイッチ XP などはこの類である．Data Mountain [8] では使用していないウィンドウをアイコンではなく縮小（サムネイル）表示することで，Forager [9]，Task Gallary [10] は仮想的な三次元空間に配置することで，それらのウィンドウの一覧性と検索性を高めた．先の文献 [6] はこの類である．しかしこれらは普通に表示されているウィンドウの切り替え方法は扱っていない．

普通に表示されているウィンドウの閲覧問題を解決する研究としては，久納ら，杉山ら，神原らの研究があげられる．久納らは，首を傾げてウィンドウの後ろを覗き込む動作を行うと，頭の動きを検出して，前側のウィンドウが首を傾けた方向とは逆方向へ，後側のウィンドウは首を傾けた方向へ移動し，後側のウィンドウが閲覧できるユーザインタフェースを提案している [11]．また，杉山らは視線の動きから覗き込みを検出する方法 [12] を，神原らはジョイスティックで覗き込みを指示する方法 [13] を提案している．これらの方法は，首の動作や視線の動きだけで目的のウィンドウを閲覧することができるが，ウィンドウの配置を予めある程度決めておく必要があるという問題点がある．これに対して提案するユーザインタフェースでは，今までのウィンドウの配置の方法を変える（ウィンドウの配置に気を配る）必要はない．

また，本論文で提案するユーザインタフェースは，作業を効率よく行うために複数のデバイスで操作を可能としたものと捉えることができる．Buxton らは，本来一つのデバイスで独立して行う操作を，二つのデバイスで同時に行う方法について操作性を調べ，二つのデバイスを用いることの有効性を報告している [14]．中村らは，二つのマウスを利用すると，いくつかの操作では効率が向上するとの報告を行っている [15]．二次元 GUI の操作において，複数デバイスを同時に操ることで操作できるシステムの提案としては，Toolglass [16]，Bricks [17]，T3 [18] などがあるが，これらはウィンドウの切り替えに応用した事例ではない．中村らは両手にマウスを持つ環境において，片方のマウスのホイールを操作することで，ウィンドウを選択することができる方法 [15] を提案しているが，あらかじめウィンドウを登録しておくなどの準備が必要である．

さらに，ウィンドウの重なり順序を変えずに下のウィンドウの確認やアクセスを可能とする，アイコンのドラッグなどのマウス操作とウィンドウ操作を独立／平行に行える，という本論文で提案するインタフェースの特長を提供しているものはない．

3. 新インタフェースの提案

3.1 次元のミスマッチ

オーバーラップウィンドウの世界は，平板なウィンドウが複数枚積み重なった三次元空間であるとみなせる．一方，現在もっとも普及しているポインティングデバイスであるマウスは，二次元上の点を指し示すために考案

されたデバイスである．

これまで，ウィンドウ切り替えの問題を解決しようと，さまざまなアイディアが考案・実装されてきたが，満足のいく結果が出ていないのは，この点が明確に意識されないまま，試行錯誤が行われてきたからではないかと考えられる．つまり，次元のミスマッチが発生していたのである．たとえば，画面上に表示したボタンを二次元の位置を示すためのマウスでクリックすることによって，ウィンドウを最上位に表示するなどの三次元目の操作を行わせていた．

3.2 ぱらぱらウィンドウ

そこで我々は，三次元目を操作するための入力デバイスとして，一次元の変移を入力することができるものを用い，その入力デバイスから変移を入力すると，最上位のウィンドウから順番に非表示，または半透明になり，下に位置するウィンドウが見えるように，そして，操作できるようになるユーザインタフェース（ぱらぱらウィンドウ）を提案する．

具体的な例を示すと，図1のように，スライドボリューム型のデバイスのつまみを押し込むと，押し込んだ量に応じて上位のウィンドウから非表示になるユーザインタフェースである．

これで，次元のミスマッチが解消でき，また，平面はマウス，奥行きは新デバイスと，役割分担を明確にすることで混乱が避けられる．三次元入力デバイスで同様の操作を行えるようにする方法も考えられるが，この場合，三次元同時に入力しなければならなく，また，三次元空間の動きは不安定となりがちで，操作が難しい．これに対して提案する方法では，従来の二次元の入力とは独立して三次元目を入力でき，かつ，単純な動きだけで入力できるため，操作が簡単である．

3.3 基本設計

次に，提案するユーザインタフェースの基本設計を示す．

3.3.1 入力デバイスはセルフリターン式を基本とする

たとえば，図1のスライドボリュームであれば，つまみを押し込んだ指を離したり，力をぬいたりすると，自然に初期位置につまみが戻るものとする．このようにすることで，非表示にしたウィンドウを再表示して，元の状態に戻したいときには，指を入力デバイスから離すだけで済む．

ただし，セルフリターン式の場合，いくつかのウィンドウを非表示にした状態で留めたいときには，つまみを押さえ続けなければならない．セルフリターン式と非セルフリターン式のどちらが使いやすいかはユーザの好みや，使い方によって異なると考えられる．そこで，非セルフリターン式も考慮することとする．

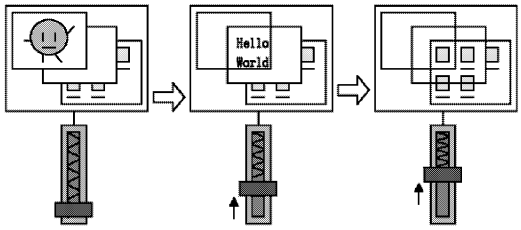


図1 ぱらぱらウィンドウの基本動作
Fig. 1 Basic operation of PARA-PARA Window.

3.3.2 従来の操作を妨げない

提案する操作方法でウィンドウを非表示にしている間も，他の操作は普通に行えるようにする．このようにすることで，次節3.4.2，3.4.3項に示すような操作が行えるようになる．

3.4 操作方法の例

本ユーザインタフェースを用いたウィンドウ操作の例を示す．なお，ここではウィンドウシステムとしてMicrosoft Windows（以下Windowsと記す）を例とし，また，入力デバイスとしては，前出のスライドボリューム型のものを用いて説明する．

3.4.1 基本操作：ウィンドウのブラウズ

前記したように，スライドボリュームのつまみを押していくと最上位のウィンドウから消えていき，押す力を抜いてつまみを戻すと消えたウィンドウが順に表示されていく (図1)．このインタフェースを用いることで，目的のウィンドウを探し出すために，作業中のウィンドウを無意味に移動したり最小化したりする必要がなくなる．また，つまみを押し続けると，最終的にすべてのウィンドウが消えてデスクトップが表示されるため，デスクトップへのアクセスも容易となる．さらに，セルフリターン式であれば，つまみを放すだけで元の状態に簡単に戻せる．

3.4.2 ウィンドウの切り替え

あるウィンドウを最上位にして作業を行いたい場合（Windowsではアクティブウィンドウにすると言う）には，3.4.1項の操作によってウィンドウをブラウズし，目的のウィンドウが見つかり，ウィンドウ上部のタイトルバーが操作できるようになったら，タイトルバーにマウスカーソルを合わせてマウスボタンを押下する．続いて，つまみを放してすべてのウィンドウを表示した後に，マウスボタンを離せばよい．この操作方法を応用することで，任意のウィンドウを任意の奥行きに配置することも可能である．

3.4.3 ウィンドウ間のドラッグ＆ドロップ

ウィンドウ間でアイコンをドラッグ＆ドロップするときも，3.4.1項の操作で移動元のウィンドウを表示させ，

アイコン上にマウスカーソルを合わせてマウスボタンを押下してドラッグを開始する．続いて移動先のウィンドウを見えるようにし，そのウィンドウの中にドロップをすればよい (図 2)．同様の手順で，文字列などのコピー & ペーストなども行える．ドラッグ& ドロップのために 2 つのウィンドウの位置を整える，という無駄な作業が必要なくなる．

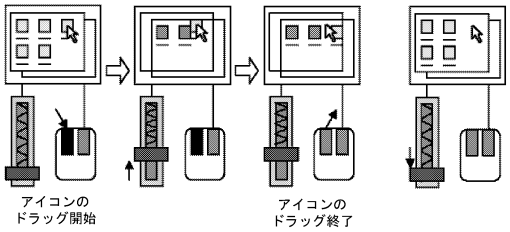


図 2　ぱらぱらウィンドウを利用しながらのドラッグ＆ドロップ
Fig. 2　Drag&Drop while using PARA-PARA Window.

4. 実　　装

提案したユーザインタフェースの実現可能性・有用性を確認するために，Windows 上で利用できるようにするソフトウェアを試作した．

4.1　入力デバイス

入力デバイスとしては，広く一般の人にも利用できるように DirectX 対応の市販デバイスを対象とした．また，セルフリターン式ではないが，ほとんどのユーザが所有するホイールマウスのホイール部でも操作できるようにした．

4.2　構　　成

本ソフトウェアは主に，デバイス監視部，ウィンドウ非表示部，ウィンドウ再表示部，システム監視部から構成される（図 3）．次に各部の動作を述べる．

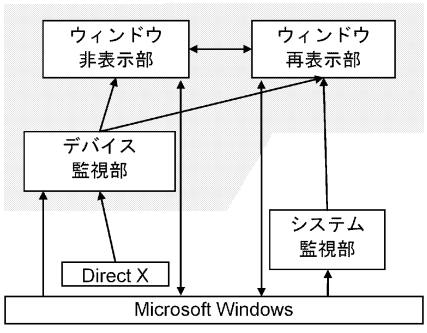


図 3　ソフトウェア構成
Fig. 3　Structure of software.

4.2.1　デバイス監視部

入力デバイスの変移を監視し，隠し枚数=((変移量-バイアス) ÷規定値) を計算する．ここでバイアスは，デバイスに触っていないときも若干の変移が検出されることが多いため，これを無効化する定数である．また，規定値はデバイスの変移量に対して非表示するウィンドウの枚数の割合を決める係数で，ユーザが自由に変更することができる．

また，ホイールマウスのホイールが回転された場合は，その方向に応じて隠し枚数を増減させる．通常 Windows 上のアプリケーションでマウスの動作を監視する場合は，ウィンドウシステムが通知するイベントを用いるが，この方法では，あるウィンドウが監視できるマウスの動作は，そのウィンドウへの入力だけに限られてしまう．本ソフトウェアでは常にホイールの動きを知る必要があるため，マウスの動きも DirectX 経由で取得する．

なお，DirectX からは 3 方向への変移，3 軸の回転，2 個のスライダ変移の 8 種類の入力を取得できるので，これらすべての入力に対応させる．

4.2.2　ウィンドウ非表示部・再表示部

隠し枚数が増加した場合，その時点で最上位に表示されているウィンドウから増加分の枚数だけ非表示にする．ただし，Windows では，見かけ上一つのウィンドウでも内部的には複数のウィンドウから構成されている．見かけ上のウィンドウ単位で非表示を行うために，Windows の標準的なウィンドウスタイルであるタイトルバー（ウィンドウ上部のタイトル名が表示される部分）があること，システムメニュー（ウィンドウ右上のボタン）があることを満たすウィンドウを非表示の対象とする．また，3.4.2 項に述べた操作を可能とするために，ドラッグ中のウィンドウは非表示にする対象とはしない．

一方，隠し枚数が減少した場合は，最近非表示にしたウィンドウから減少分の枚数だけ再表示を行う．再表示の際には最前面に表示されるようにするが，ドラッグしているウィンドウがある場合にこの処理を行ってしまうと，ドラッグしているウィンドウが再表示したウィンドウより下位になってしまい，3.4.2 項の機能が実現できなくなる．そこで，そのようなときには，強制的に最前面にする処理は行わない．

なお，再表示しようとしたウィンドウが既に破棄されていたときも，再表示を行った枚数として数えることによって，スライダの変移量と非表示になっているウィンドウの枚数の整合性をとる．また，新たなウィンドウの生成には特別な関与はしないため，操作中にウィンドウが生成されたときには，その時点の最上位に表示される．続いてさらに隠し枚数が増えた場合は，この生成されたウィンドウが最初に非表示され，逆に隠し枚数が減った場合は，この生成されたウィンドウの上に最も最近非表示されたウィンドウが表示されるという動きになる．

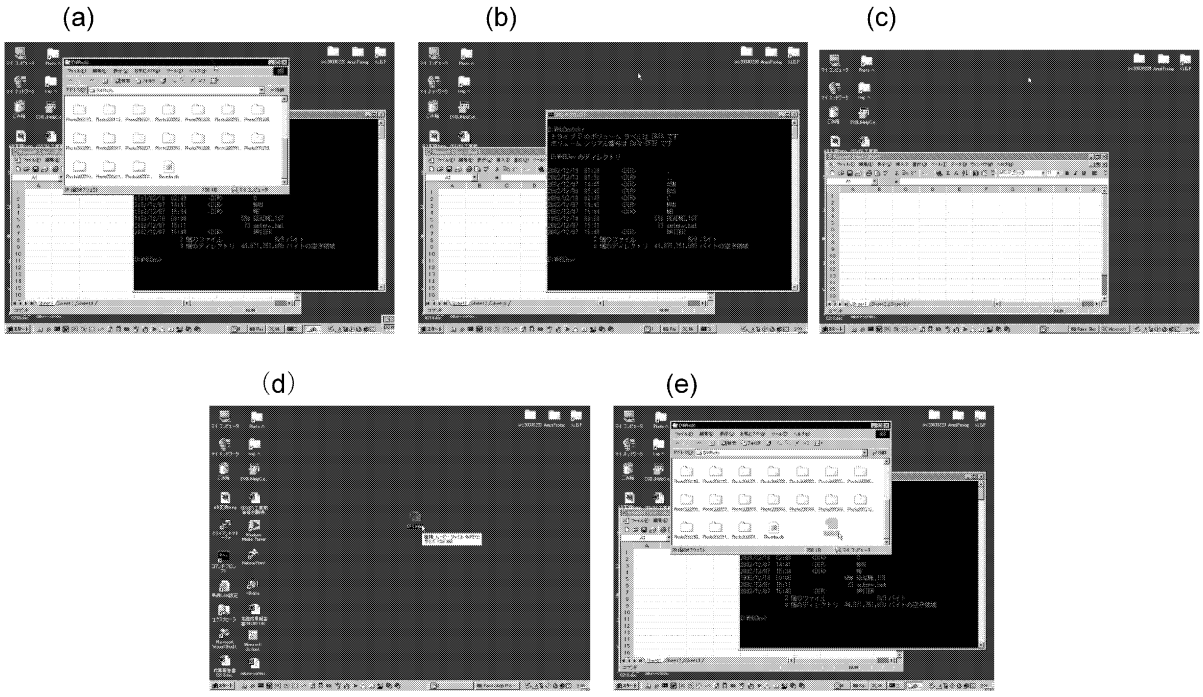


図 4　利用しているときの画面
Fig. 4　Screenshots of use scene.

4.2.3　システム監視部

ウィンドウ非表示部と再表示部では，ドラッグ中のウィンドウを特別扱いすることを述べた．このために，ウィンドウがドラッグ中であるかどうかの監視をする必要がある．

また，あるウィンドウが非表示にされた後に，そのウィンドウを持ったアプリケーションソフトウェアが終了されるなど，ウィンドウ自体が存在しなくなることが考えられる．このようなウィンドウをウィンドウ再表示部で再表示するわけにはいかない．そこで，ウィンドウの破棄も監視する必要がある．

Windows では，これらの現象が発生すると，そのウィンドウや親子関係にあるウィンドウに対してイベントが送られる．しかし，マウスのイベントと同様，関係のないウィンドウがそのイベントを受けることはできない．そこで，システム監視部ではウィンドウシステムに対するイベントフックと呼ばれる技術を利用し，イベントを覗き見する．なお，すべてのウィンドウのイベントをフックするための条件に従って，システム監視部は DLL (Dynamic Linking Library) として実装する．

4.3　実　　現

ぱらぱらウィンドウを利用しているところの実行画面を図 4に示す．左上から順に，(a) 初期状態，(b) スライダのつまみをすこしだけ動かして最上位のウィンドウが消えたところ，(c) さらに動かして 2 枚目が消えたところ，(d) さらに動かしすべてのウィンドウを消し，デスクトップ上のアイコンをドラッグし始めたところ，(e) つまみを離してすべてのウィンドウを表示させ，最上位のウィンドウ上でドラッグ中のアイコンをドロップしたところである．また，利用しているところの模様を撮影したビデオを web ページ [1] 上で公開している．

5.　評 価 実 験

提案したユーザインタフェースの有効性を確かめるために，単純なタスクを，マウスを用いて通常の操作方法で行うときと，提案したユーザインタフェースで行うときの操作時間を比較する評価実験を行った．次にこの実験について述べる．

5.1　実 験 方 法

5.1.1　タ　ス　ク

操作時間を比較するためのタスクは，3.4.2 に対応するタスクとして，下に位置するウィンドウは上に位置するウィンドウに完全に隠されるように重ねて置かれた 4 枚のウィンドウのうち，最下位のウィンドウを最上位にするタスク（実験 1:図 5上）と，3.4.3 に対応するタスクとして，最下位のウィンドウ内のアイコンをドラッグし，上から 2 枚目のウィンドウへドロップするタスク（実験 2:図 5下）とした．

5.1.2　操 作 方 法

このタスクをマウスで行う場合には，ウィンドウの移動またはサイズ変更だけを用いて行うこととした．ぱら

1: http://lab.bmoon.jp/parapara.html

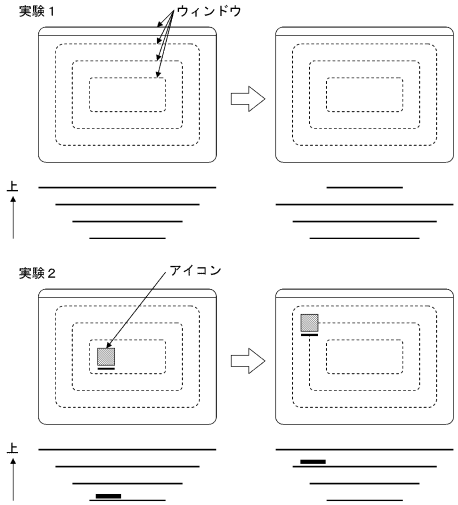


図 5　実験におけるタスク
Fig. 5　Tasks for experiments.

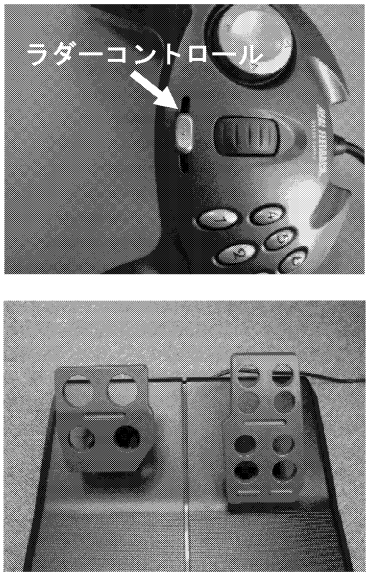


図 6　実験に用いた入力デバイス
Fig. 6　Input devices for experiments.

ぱらウィンドウを用いる場合の入力デバイスとしては，前にも例として示した指で操作するセルフリターン式のスライドボリューム型デバイスに近いものとして，サンワサプライ社製コントローラ Real Feedback のラダーコントロール部 (図 6上) を用いた．このデバイスはマウスを持たない手で操作してもらうこととした．ここで，手はキーボードの操作にも用いるため，実際の利用場面ではデバイスの持ち替えの煩雑さが増えることが予想される．そこでコンピュータを操作する際に用いることが少ない足で操作するデバイスとして，Logitech 社製コントローラ GT Force のフットペダル部 (図 6下) を用いた実験も行った．

表 1　平均操作時間と標準偏差（秒）
Table 1　Average task time and standard deviation (sec)

	実験 1	実験 2
ペダル	2.16(0.48)	4.38(0.70)
ラダー	2.07(0.42)	3.95(0.75)
マウス	4.02(0.51)	5.95(0.41)

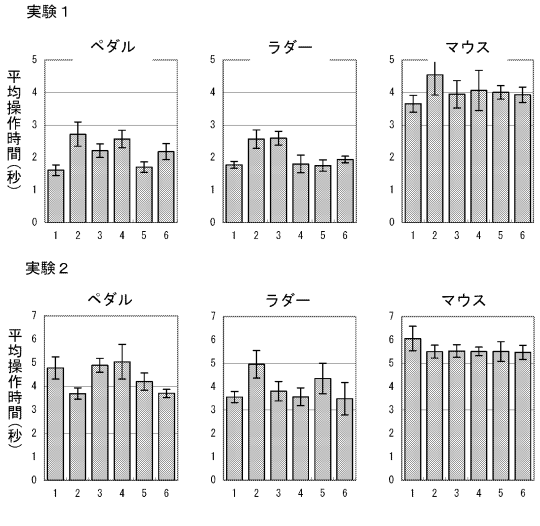


図 7　被験者別平均操作時間
Fig. 7　Average task time for each subject.

5.1.3　被験者と試行回数

被験者は Windows の操作に慣れた大学生 6 名で，各タスクについてマウスとペダルとラダーそれぞれについて 10 回行ってもらった．使うデバイスの順番が実験結果に与える影響を減らすために，被験者によって使用するデバイスの順番を変化させた．計測する時間は，開始を合図する音からタスクが完了するまでの時間とした．なお，タスク開始時にはマウスカーソルを画面中央に戻させ，また，できる限り早く正確に操作するように指示した．

5.2　実 験 結 果

計測した操作時間の平均値を表 1に示す．また被験者ごとの平均値を図 7に示す．上が実験 1 の，下が実験 2 の結果を示している．また，図内の棒は平均値を，線分は平均値±標準偏差を表している．なお，この平均値は，被験者 1 人につき 10 回計測した操作時間のうち，最長および最短操作時間を除いた 8 回から算出した．

5.3　考　　察

両方の実験において，提案したユーザインタフェースを用いた場合の操作時間は，マウスを用いて通常の操作方法で行う場合の操作時間より短いことが示された．分散分析（乱塊法）を行ったところ，有意水準 1%でデバイスの違いによる操作時間の平均値の差が認められた（実験 1 : $F(2,10) = 124.7$，実験 2 : $F(2,10) = 13.88$）．

そこで，多重比較（テューキーの方法）を行ったところ，実験 1 ではマウスとペダル，マウスとラダー間で操作時間に差があることが認められた（名義水準 0.3%）．実験 2 ではすべてのデバイス間で操作時間に差があることが認められた（名義水準 0.3%）．

どの方法（デバイス）が操作しやすいかを口頭で尋ねたところ，ペダル（3 人），ラダー（2 人），マウスを含めどれとも言えない（1 人）に分かれ，提案したユーザインタフェースが肯定的に受け入れられたことがわかる．

一方，ラダーが小さく操作しづらい，ペダルはある位置で止めるのが難しい，ウィンドウの非表示，再表示に合わせてクリック感があるとわかりやすい，との意見が得られた．また，展示会などで実際に利用してもらった際にも，連続的な値を入力するデバイスで離散的な操作をすることに違和感があることを理由として，マウスのホイールの方が使いやすい，ペダルを 1 回踏むたびに 1 枚のウィンドウが消える方式がよいのではという意見が得られた．また，ラダーやペダルを横にずらすとロックがかかるようにすれば良いのでは，常にロックがかかって戻らないようしてなんらかのボタンを押すと戻るようにすれば良いのではなどのアイデアも得られた．さらに，単純にウィンドウが消えるのでなく，半透明に徐々に消えていく形式や，上にスルスルと上って隠れる形式，ウィンドウが中心に向かって小さくなる形式などを利用してもらったところ，それぞれのよい点が認められた一方で，人によって好みが異なることが明らかになった．

デバイスの形状やフィードバックがユーザインタフェース自体の使いやすさに影響することは明らかであり，また，様々な好みや要求があることから，提案するユーザインタフェースに適する入力デバイスの検討を行うとともに，操作方法を選択可能なソフトウェアとして完成させることも重要な課題であると言える．

6. おわりに

本稿ではウィンドウの切り替えに伴う操作性の問題点を解決するための操作方法を提案した．このユーザインタフェースを利用することで，複数のウィンドウが重なっている状態でも，意図するウィンドウを容易に表示させ，操作することができる．そして，ウィンドウの重なり順序を変えずに下のウィンドウの確認やアクセスを可能とすること，アイコンのドラッグなどのマウス操作とウィンドウ操作を独立／平行に行えること，という効果も得られる．評価実験からは，マウスを用いる通常の操作方法に比べ，このユーザインタフェースを用いることで，下に位置するウィンドウを最上位にするタスクや，二つのウィンドウ間でアイコンを移動するタスクの操作時間が短くなることが示された．

ペンで操作するコンピュータでは，スタイラス（ペン）による細かい領域のクリックやドラッグ操作は行い難く，また，キーボードがついていないという条件から，マウスとキーボードを用いる環境に比べ，さらにウィンドウの操作が難しい．そこで，Tablet PC などにこのユーザインタフェースを適用する意義は非常に高いと考えられる．たとえば，Tablet PC を持つ際に親指が添えられる部分にスライダをつけて，ぱらぱらウィンドウの操作を割り当てることで，ペンをウィンドウの選択操作から解放することができる．

また，今回提案したユーザインタフェースは，三次元目（奥行き）の操作を直感的に操作できる特徴を持ち，ウィンドウの操作以外にも，奥行きの指示が面倒であった場面，たとえば作図ソフトにおいて複数重なった状態にある図形オブジェクトの選択など，を解決する可能性を持っている．

今回の評価実験はあくまで決められたタスクの操作時間を比較したものであり，提案したユーザインタフェースが実際どのくらい利用される可能性があるのか，どのくらい便利なのか，および，タスクバーなどを用いるウィンドウ切り替え方法に比べて使いやすいかどうかなどは，長期的に常用してもらわなければわからない．そこで，実際に長期間に渡って利用してもらうことを通した現実的評価が今後の課題である．また，最適なデバイスの形状や構造の探求，ウィンドウ操作以外への応用を行っていきたいと考えている．

謝　辞

本研究は情報処理振興事業協会未踏ソフトウェア創造事業採択プロジェクト「スライドボリュームを利用したウィンドウ操作体系の構築」の一環として行われたものです．

参考文献

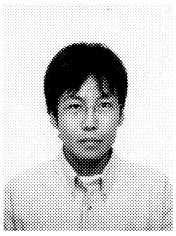
[1] 小國健，加藤直樹：ウィンドウの切り替えを容易にするインタフェースの提案；情報技術レターズ, Vol.1, pp.201-202 (2002)
[2] 加藤直樹，小國健：ぱらぱらウィンドウ：ウィンドウの切り替えを容易にするインタフェース；インタラクション 2003 論文集, pp.123-130 (2003)
[3] 枠ピタ：http://www.01-tec.com/WakuPita/
[4] スイッチ XP：http://www.geocities.co.jp/SiliconValley-PaloAlto/8654/tools.html
[5] RWiper: http://www.kit.hi-ho.ne.jp/modified-r32/SoftGallery/softgal.html
[6] 窓立て：http://www.ksky.ne.jp/ seahorse/mtate2/
[7] Stuart K. Card and Austin Henderson, Jr.: A multiple, virtualworkspace interface to support user task switching; Proceedings of the CHI+GI, pp. 53-59 (1987)
[8] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel and Maarten van Dantzich: Data Mountain:Using Spatial Memory for Document Management; Proceedings of the UIST '98, pp.153-162 (1998).

[9] Stuart K. Card, George G. Robertson and William York: The WebBook and the Web Forager: An information workspace for the World-Wide Web; Proceedings of the CHI '96, pp. 111-117 (1996)
[10] George Robertson, Maarten van Dantzich, Daniel Robbins, Mary Czerwinski, Ken Hinckley, Kirsten Risden, David Thiel and Vadim Gorokhovsky: The Task Gallery:A 3D Window Manager; Proceedings of the CHI2000, pp.494-501 (2000)
[11] 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；情報技術レターズ, Vol.1, pp.199-200 (2002)
[12] 杉山覚，渋谷雄，倉本到，辻野嘉宏:マルチウィンドウ環境における覗き込み動作を利用した情報閲覧手法;情報処理学会研究報告 (HI-107), Vol.2004, No.14, pp.1-8 (2004)
[13] 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)
[14] William Buxton and Brad A. Myers: A Study in Two-Handed Input; Porceengs of the CHI '86, pp.321-326 (1986)
[15] 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；ヒューマンインタフェース学会論文誌, Vol.2, No.4, pp.309-321 (2000)
[16] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton and Tony D. DeRose: Toolglass and magic lenses: the see-through interface; Proceedings of the SIGGRAPH '93, pp.73-80 (1993)
[17] George W. Fitzmaurice, Hiroshi Ishii and William A. S. Buxton: Bricks: laying the foundations for graspable user interfaces; proceedings of the CHI '95, pp.442-449 (1995)
[18] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel and Bill Buxton: The design of a GUI paradigm based on tablets, two-hands, and transparency; Proceedings of the CHI '97, pp.35-42 (1997)

（2004 年 11 月 1 日受付，2005 年 3 月 7 日再受付）

著 者 紹 介

加藤　直樹　（正会員）



1969 年生．1998 年東京農工大学大学院情報工学専攻博士後期課程修了．1997 年より日本学術振興会特別研究員，1999 年より東京農工大学工学部助手を経て，2004 年より東京学芸大学教育実践研究支援センター助教授．ヒューマンインタフェース（特にペン入力），教育へのコンピュータ利用に関する研究・教育に従事．ACM，情報処理学会，教育システム情報学会各会員．工学博士．

小國　　健

1998 年東京農工大学大学院情報工学専攻博士前期課程修了．同年 NTT データ通信株式会社（現・株式会社 NTT データ）入社．現在に至る．