特集●インタラクティブソフトウェア

「アイコン投げ」ユーザインタフェース

久野 靖　大木 敦雄　角田 博保　粕川 正充

図的ユーザインタフェースの一種として，アイコンで様々な対象を表現し操作するWIMP [1]インタフェースがある．WIMPインタフェースでアイコンに対する操作を指示するやり方(操作選択方式)としてメニュー，ドラグ＆ドロップ，キー操作などがあるが，これらの間には，操作時間が短いものは柔軟性や分かりやすさに欠け，柔軟で分かりやすいものは操作時間が長いというトレードオフがある．筆者らはドラグ＆ドロップの改良版として「アイコン投げ」(ドラグの途中でマウスボタンを離してもアイコンがこれまでの速度と方向で目的地に向かって移動し続ける方式)に注目し，その操作時間について実験と検討を行った．実験の結果，「アイコン投げ」インタフェースはメニュー，パレット，ドラグ＆ドロップなどのマウスによる選択方法より有意に高速であり，誤り率も他の方法と比べて遜色ないため，図的インタフェースの基本操作として有望である．

"Icon Throwing" User Interface

Yasushi Kuno, Atsuo Ohki, 筑波大学大学院経営システム科学専攻, Graduate School of Systems Management, The University of Tsukuba, Tokyo.

Hiroyasu Kakuda, 電気通信大学情報工学科, Department of Computer Science, University of Electro-Communications.

Masaatsu Kasukawa, お茶の水女子大学情報科学科, Department of Infomation Science, Ochanomizu University.

コンピュータソフトウェア, Vol.13, No.3 (1996), pp.38–48.
[論文] 1995年5月30日受付.

1 はじめに

近年において著しい変化と発展が見られた分野の1つとして，計算機システムの図的ユーザインタフェース(Graphical User Interface, GUI)の分野が挙げられる．GUIにも様々な形態があるが，ファイル，ディレクトリ，ウィンドウなどのオブジェクトをアイコンによって表し操作させるという方式もその1つである．本稿ではこのような枠組みを「WIMP (Windows, Icons, Menus, Pointing) インタフェース」と呼ぶ[†1]．WIMPインタフェースはMacintoshのFinderやWindowsのProgram Managerなど，ユーザが長時間使うプログラムのインタフェースとして採用されており，その操作性はユーザに大きな影響を与える．

WIMPインタフェースでは各種の操作対象(オペランド)はアイコンとして表されるが，そのアイコンに対する操作(オペレーション)を指定する方法(本稿では「操作選択方式」と呼ぶ)には

- メニュー(プルダウン，ポップアップ)
- ドラグ＆ドロップ
- 押しボタンないしツールパレット
- キー操作(キーボードショートカット)
- マウスボタンと修飾キーの組み合わせ

などがある．操作選択方式がWIMPインタフェースの操作性に及ぼす影響は大きいので，よりよい選択方式を探求することは重要である．

WIMPインタフェースの利点として初心者にも分か

†1 WIMPインタフェースという用語をもっと広い範囲のものと考える研究者もいる．

りやすいことが挙げられるが，その大部分は圧倒的に多く使われる操作選択方式であるメニューの特性(文脈への適合性，多数の選択肢の提示，階層構造)に負っていると思われる．反面，メニューの選択はかなり複雑なプロセスであり，操作時間の観点からは不利である．

ところで，旧来の文字端末ユーザインタフェースでは，次のような枠組みが代表的である．

- 指令を文字列として打ち込む．
- 画面に書式を表示させ，空欄に値を打ち込む．
- メニューを表示して選択させる．

これらのうち，メニューは初心者でも習得が容易だが操作時間が長く，逆に指令方式は最も高速だが習得に時間が掛かる(つまり熟練者向けである)．

操作時間と習得しやすさの同様なトレードオフはWIMPインタフェースの操作選択方式にも見られ，メニューが初心者向きなのに対し，キーボードによる選択は熟練者向けといえる．ここでもし，直観的に分かりやすく，しかも高速に操作できる操作選択方式が設計できれば，多くのユーザが使うと考えられる．

筆者らはこのような考えに基づき，「アイコン投げ」と呼ばれる操作をWIMPインタフェースの操作選択方式として使用することを検討してきた．以下本稿では第2節で各種の操作選択方式の得失を検討し，第3節でそれらと対照して「アイコン投げ」の特徴について論じる．続く第4節では筆者ら自身が被験者となって実施した，各種操作選択方式の操作時間を測定する実験とその結果について述べ，第5節では実験時に採取したタイミングデータをキーストロークレベルモデルにあてはめ分析を行う．第6節でこれらの結果に基づき議論を行い，最後に第7節でまとめを述べる．

2 既存の操作選択方式

WIMPインタフェースにおいてアイコンをポインティングデバイスで指定した後，そのアイコンが表現している実体に適用すべき操作を計算機に指示する方法を本稿では「操作選択方式」と呼ぶ(操作を指示した後で対象を指定する方法もあるが少数派であり，一般には対象を先に指定するほうが使いやすいとされる)．

ポインティングデバイスとしては電子ペンやトラックボールなども使われるが，本稿では最も多く使われているマウスを前提とする．以下では既存の代表的な操作選択方式について検討する．

2.1 マウスボタン

すべてのマウスにはボタンが備わっており，アイコンの選択に使用する．ここで，複数のボタンを備えたマウスであれば，押したボタンに応じて異なる操作が選択されるようにできる．また，1ボタンマウスを使用するMachintoshで行われているように，マウスボタンを連続して押したり(ダブルクリック)，修飾キー(Shift, Optionなど)を押しながらマウスボタンを押すことで複数の操作を区別することもできる．

マウスのボタンに対応させた選択では，操作対象の指示と操作選択が1動作でできるため，操作時間の短さは最も優ると思われる．しかし反面，ボタンや修飾キーの少なさから多数の操作を区別するには向かない．このため，マウスボタンによる操作選択は，頻繁に使うごく一部の汎用的な操作を起動する場合にのみ使われるのが普通である(例えばMachintoshではダブルクリックを多くの場面で「開く」という操作に対応させている)．また，マウスボタンによる選択でメニューを起動する(ポップアップメニュー)のも一般的である．

2.2 メニュー

本稿ではメニューを「普段は見えないが，何らかの操作によって画面に表示され，項目の選択を許すような機構」の意味で用いる．メニューは操作選択方式の中では最も一般的で広く使われる．その理由として次のことが挙げられる．

- 普段は見えないので，画面上の領域を浪費しない．
- 逆に，提示時には比較的広い領域を使える．
- 文脈に合わせて構成を変化させられる．

メニューはその提示方法によりポップアップメニューとプルダウンメニューに大別できる．

ポップアップメニューはマウスカーソルがどこにあっても起動できるが，起動操作はアイコンの選択と区別できなければならない(通常，別のボタンや修飾キーを用いる)．プルダウンメニューはそのような区別は不要だが，メニューバーなどメニューを出すための領域に，マウスカーソルを移動する必要がある．

選択は，メニューが表示された後，さらにマウスカーソルを選択したい操作に対応する領域まで移動させて行う．したがって，メニューによる操作選択は

1. メニューを出すための動作を行う．
2. メニューが出たことを認識する．
3. 選択したい場所までカーソルを移動する．
4. 選択動作 (通常はボタンを離す) を行う．

という段階から成るかなり込み入った操作だといえる．

メニューが多数の選択肢を持つことは可能だが，数が多くなると (目指す項目を見つけたりサブメニューをたどるのに) 時間が掛かるようになる．さらに，メニューが表示されるまでは，どこに目指す項目があるか分からないので，他のマウス操作に多く見られる「マウスを移動しながら目指す対象を探す」という最適化が難しい (頻繁に使うメニュー項目について，メニューの何番目にあるか覚えてしまうことはあり得る)．

2.3 パレット

メニューと異なり，選択のための領域を常に表示しておく方法もある．これには描画ツールなどによく見られる「ツールパレット」，一般の GUI に見られる「押しボタン」，メニューを常時表示しておく「引きちぎりメニュー」「ピンメニュー」などがあるが，ここでは用語として他の概念と混同しにくい「パレット」を使う．

パレットの長所は，表示されるのを待たずに済み，操作手順が単純なことである．反面，常に画面を占有すること，そのため画面の周辺に置くことが多く，かつ領域を小さくしがちで，結果的にマウスカーソルを領域内へ移動させるのに時間が掛かること，などが欠点である．

2.4 キーボード

キーボードは物理的には多数のボタンが並んだものなので，これを操作選択に使うことは自然であり，特に (マウスを右手で操作するとして) 左手の受け持ち範囲にあるキーは高速に操作できる．ただし，どのキーがどの操作選択に対応するかを覚える負担はある．特に選択肢が多くなると，ニモニックが割り当てにくくなるので負担が増す．

多くのシステムではメニューと同等の操作をキーでも行えるようにし (キーボードショートカット)，メニューにキー割り当てを表示して，頻繁に使われる項目を高速なキー操作に移行するよう配慮している．

2.5 ドラグ & ドロップ

ドラグ & ドロップは，アイコン自身のみによって操作を指定するという点が他の操作選択方式と異なる．すなわち，操作対象と操作の両方をアイコンで現し，対象アイコンの上でマウスボタンを押し，その状態で操作アイコンのところまで移動してきて (drag)，重ねた状態でマウスボタンを離す (drop) ことで操作を起動する．ドラグ & ドロップの操作系列は

1. 対象アイコンの上でマウスボタンを押す．
2. 操作アイコンの上までそのまま移動する．
3. マウスボタンを離す．

であり，1 が対象アイコンの選択操作を兼ねるため，残りの 2 と 3 が操作選択に必要な部分となる．

もう 1 つの特徴として，操作アイコンもアイコンであり，それを好きな位置に配置したり機能を変更するなどのカスタマイズが同じ枠組み中で行えることがある．ユーザが自分のよく使用する操作を効率よく行えるよう，インタフェースをカスタマイズできることは重要である．他の方式でもカスタマイズはできるが，大抵は設定ファイルのようなものを編集しなければならず，可能だとしても敷居が高い．

3 マウス操作時間と「アイコン投げ」

筆者らは，ドラグ & ドロップは操作系列が単純であり，操作時間の点で有利だと考えたが，予備的な実験の結果，必ずしもそうではないと分かった．本節では，その理由をマウス移動時間に関する検討に基づいて考察し，ドラグ & ドロップの改良操作としての「アイコン投げ」について述べる．

3.1 マウス移動時間

Card らの研究に基づくキーストロークレベルモデル (以下 KL モデル) [4]によれば，マウスを特定の目標に移動させるのに掛かる時間 (ミリ秒) は，移動距離を d，目標の大きさを s とした時，次の式で表される．

$$T_p = T_W + I_M \log_2 \left(\frac{d}{s} + 0.5 \right)$$

ここで T_W はマウス操作に付随するオーバヘッドで値は 800 msec, I_M は人によって異なるが 100 msec 程度の値の定数となっている.

しかし, このモデルはキーボードとマウスの操作が適宜混ざった環境での実験に基づいて構成されたと思われ, 予備的な計測の結果, 今回のようにずっとマウスを持ったまま操作する場合には T_W の値が過大であるように思われた. そこで後述する実験のためのプログラムを応用し, 様々な距離で並んだアイコンを順次クリックしていく実験を行い, 逆算して T_W の値を求めた. その結果に基づき, 以下では 300 msec という値を使用する.

この値を用いると, マウスの移動時間はおおよそ 350 ～ 750 msec となり, これにマウスボタンを押す時間 (キーボードの打鍵時間と同じと考える) を加えると, 全体の時間はキーボード打鍵時間 (100 ～ 300 msec 程度) の 3 ～ 5 倍とかなり大きくなる.

また, キーボードの打鍵時間は熟練するにつれて短くなるが, マウスの移動時間は熟練してもさほど短縮されない. そして, メニュー, パレット, ドラグ & ドロップのいずれもがこのマウスによる位置指定を必要とする. 多量の操作をしていて「いらいらする」という感覚を持つのは, そのせいもありそうである.

3.2 「アイコン投げ」の検討

ドラグ & ドロップのマウス操作に限って考えると, テキストの編集位置や図形の位置を指定する場合と異なり, 「どれに重ねようとしているのか」さえ計算機に伝われればよい. したがって, 対象アイコンが目標アイコンに重なるまで待たなくても, 対象アイコンが動き始めれば, その経路を延長することでどの目標アイコンを目指しているかは判断でき, その後の操作はなくても済む (経路上に複数のアイコンがあるような場合は後で扱う).

そのような方式をとった場合, 利用者の操作はドラグ & ドロップの前半だけ, すなわち対象アイコンの上でマウスボタンを押し, そのまま移動を開始し, 移動の途中でボタンを離す, というものになる. これを我々は「アイコン投げ」と名づけた.

アイコンを投げる場合にどれくらい時間が掛かるかは KL モデルでは計算できないが, 予想としては「動き出せばよい」のだから, 目標アイコンまでの距離に依存せずに済み, なおかつ通常のマウス操作で時間が掛かるのは目標位置に正確に合わせるための微調整であるので, それがない分大幅に速くなるのではないか, と考えた.

4 実験による「アイコン投げ」の評価

「アイコン投げ」の操作時間その他の特性を調べるため, 筆者ら自身が被験者となり小規模な実験を行った. 実験には IBM-PC 互換機 (飯山電機製, 486DX66, メモリ 16 MB, ISA バス, 17 インチカラーディスプレイ MF-8217J, 機械式 3 ボタンマウス) を使用し, OS は BSD/386 1.1, X サーバは BSD/386 付属の X11R5 サーバを解像度 1152 × 900 ドットとして使用した. 実験プログラムは C++ で約 900 行あり (API は Xlib), BSD/386 付属の g++ により翻訳した.

4.1 実験の設計

実験の設計は次の通りである.

- 「マウスにより対象アイコンを指定し, 続いてそのアイコンに対する操作選択を行う」という動作を 1 サイクルと考え, 各種の操作選択方式について所要時間を比較する.
- 選択肢の数は人間にとってひと目で認識できる数として, 今回は 4 を採用した. より多くの選択肢については後で扱う.
- 操作選択方式としてドラグ & ドロップ, パレット, ポップアップメニュー, キーボード, および「アイコン投げ」の 5 種類を取り上げた. マウスボタンによる選択は用途が限られると考え, 除外した.

図 1 に実験用プログラムの画面の様子を示す (ポップアップメニューを出しているところ).

ここで, メニューのすぐ上にある丸いものが対象アイコンである. 選択肢は 4 つなので, 各アイコンには A ～ D までの文字が描かれ, 被験者はそれと同じ操作 (つまり操作名も A ～ D となっている) を選ぶことになる. したがって, 操作を現す目的アイコンにも, パレットにも, ポップアップメニューにも A ～ D の文字が記してある.

具体的な操作は次の通り.

- 投げる: 対象アイコンの上でマウスボタンを押し, そのまま同名の目的アイコンに向かって動かし, 動

表1 各方式による操作サイクルの平均所要時間

被験者	投げる	ドラグ	パレット	メニュー	キー
A	1.002 (0.36)	1.676 (0.42)	1.885 (0.60)	1.799 (1.43)	0.637 (0.14)
	1.00	1.67	1.88	1.79	0.64
B	0.717 (0.17)	1.311 (0.36)	1.582 (0.47)	1.258 (0.47)	0.633 (0.20)
	1.00	1.82	2.20	1.75	0.88
C	0.990 (0.24)	1.633 (0.33)	1.694 (0.47)	1.325 (0.24)	0.667 (0.11)
	1.00	1.65	1.71	1.34	0.67
D	0.884 (0.21)	1.650 (0.30)	1.769 (0.28)	1.271 (0.35)	0.686 (0.17)
	1.00	1.87	2.00	1.44	0.78

単位は秒数，かっこ内は標準偏差
下段は「投げる」時間を1とした時の比率

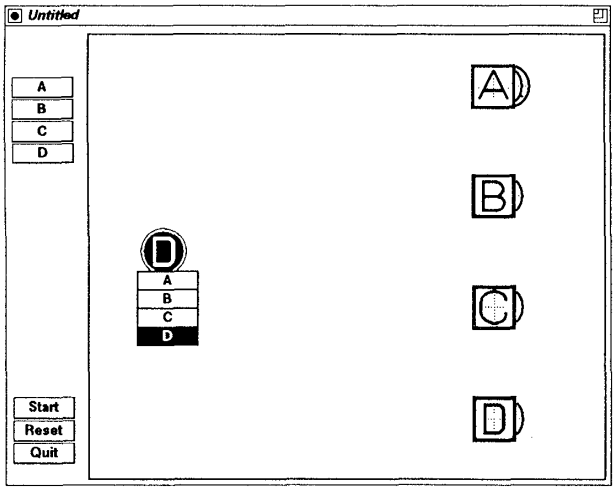

図1 実験用プログラムの画面

いた状態でボタンを離す．

- ドラグ & ドロップ: 上と同様だが，途中で離すのではなく，目的アイコンの上まで持っていって重ねた状態で離す (重なると目的アイコンの色が変わるようになっている)．
- パレット: 対象アイコンをクリックして選択した後，マウスカーソルをパレットの上に持っていって，A～Dのいずれかの上でボタンをクリックする．
- メニュー: 対象アイコンをクリックして選択した後，コントロールキーを押しながらマウスボタンを押すとメニューが表示され，その状態でマウスを動かし，A～Dのいずれかが反転した状態でボタンを離す．
- キー: 対象アイコンをクリックして選択した後，キーボードのA～Dのキーを押す．

対象アイコンは半径30ドット (実験に用いた画面では1ドット＝約0.27 mm) の円，目的アイコンは1辺60ドットの正方形，パレットとメニューの1項目は高さ25ドット，幅80ドットの長方形であった．

なお，「投げる」場合には，対象アイコンが目的アイコンに接しなくても，対象アイコンの中心が目的アイコンの中心から90ドット以内に入った時には，「引力」が働いて引き付けられるようにした．さらに，投げたアイコンが壁 (窓の四辺) にぶつかった場合には，停止するのではなく，ぶつかった壁に沿って「ころがる」ようになっている．このようにしたのは，投げた以上「何もない」ところを目指して投げることはないはずなので，多少それた方向に投げても，そばにある目的アイコンを選択したものと見なせるようにするためである．デザイン上の大きさは，目的アイコンも対象アイコンも同一である．

実験は20個の対象アイコンが同一位置に重なったものを1セットとし，5種類の選択方法を各9セット，合計45セット実行した．実験に際しては，できる範囲ですみやかに操作することとし，誤りに気がついた場合に訂正するかどうかは被験者に任せた．

4.2 実験の結果

表1に各方式毎の1サイクルの平均所要時間と標準偏差，および「アイコン投げ」を1とした時の比率を被験者毎に示す (1セットにはアイコンが20あるが，最後のサイクルは次のアイコンがないため計測できず，19サ

イクル分のデータとして使用した).

また，これらのデータについて順位和検定を行った結果，すべての被験者について，キー操作の時間は他のすべての操作時間より有意に短く，「アイコン投げ」の時間はキー操作を除くすべての操作時間より有意に短かった (有意水準 0.99). 残る 3 つの操作については，被験者によって順番がまちまちであり，操作時間に有意差が見られるか否かも被験者によってまちまちであった.

次に，表 2 に誤り数を集計したものを示す. これは実験で採取した事象の系列から時刻，マウス位置，位置報告イベントなどの情報を削除した操作系列を作り，それとあらかじめ用意した標準的な操作系列とで異なっている箇所を数えたものである (複数の連続した差異は 1 個に数え，ただしサイクルをまたがる場合は分けて数えた). さらに，これらのうちメニューの項目番号や目標アイコン番号のみが違っている場合を選択誤り (下段)，それ以外を操作の逸脱 (上段) として分類した.

表 2 各方式による操作の逸脱 (上) と選択誤り (下)

投げる	ドラグ	パレット	メニュー	キー
1 0.63%	5 3.13%	13 8.13%	11 6.88%	5 3.13%
5 3.13%	1 0.63%	0 0%	22 13.75%	0 0%

ただし全操作系列を検討するのは大変だったので，9 ラウンドの実験のうち最初の 2 ラウンド (40 操作) のみについて行い，誤り件数が多くないことから全被験者の誤りを合計して示した. これを見ると，「投げる」操作における選択誤り (「投げ間違い」) の比率は予想通りドラグ & ドロップよりは多いが，メニューの選択し損ないよりはだいぶ少ないことが分かる.

なお，メニューの選択し損ないは大部分が選択時にオーバーシュートして隣りの項目を選んでしまうというもので，パレットでこれが少なかったのは項目間にすき間が空いていたためと思われる (その代わり，オーバーシュートして項目外でマウスボタンを離すという逸脱誤りが多発している). 「アイコン投げ」は熟練を要するため，実験時にプレッシャーとなりマイナスに作用することも予想されたが，これらの結果を見る限り，プレッシャーが悪影響を与えたのはメニューやパレットに対してだったといえる.

5 操作時間の分析と検討

各操作のより細かい分析を行うため，各実験における 1 サイクル分の操作系列を抽出してタイミングチャートを作成した (図 2〜図 6). 抽出に際しては，エラーの分析を行った第 2 ラウンドのデータを用い，エラーを含むサイクルと他のサイクルより明らかに長いサイクルを除外し，残ったものの平均をとった. 以下では，各操作系列の KL モデルによる時間予測も合わせて検討を行う.

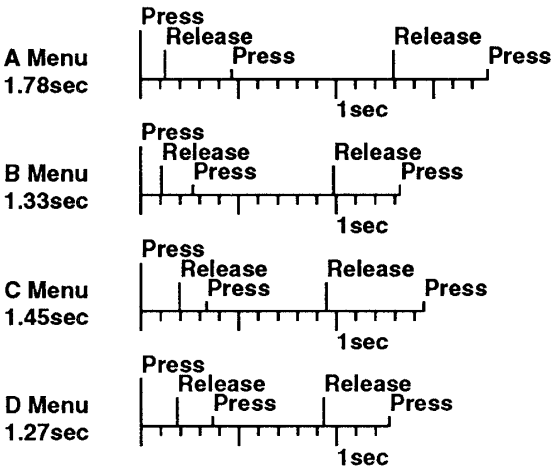

図 2 メニュー操作のタイミング

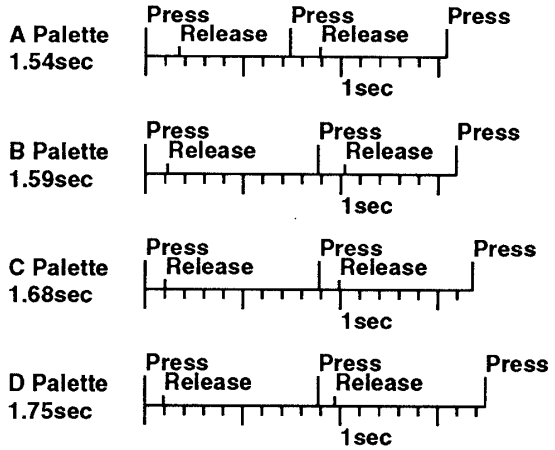

図 3 パレット操作のタイミング

5.1 メニュー操作 (図 2)

まず対象アイコンを選択するためボタンを押してすぐに離し，次にメニューを出すため再度 (コントロールキーとともに) 押す. ここで被験者 A だけ時間が掛かっているが，これは被験者 A は対象アイコンの真上にメニューが

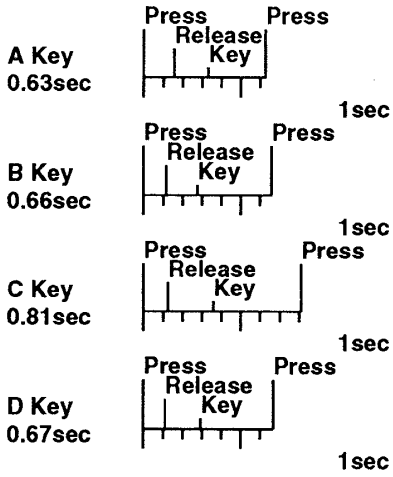

図4 キー操作のタイミング

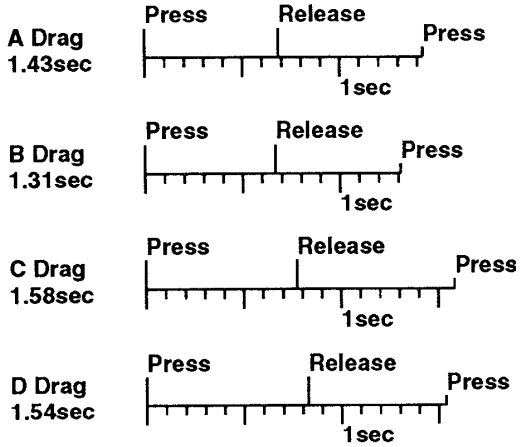

図5 ドラグ & ドロップ操作のタイミング

出るのを好まず，少しマウスを移動してからメニューを出していたためである．次にボタンを押したままマウスを移動し，メニューを選択するまでは比較的時間が掛かるが，その後次の対象アイコンの位置 (最初の位置) までカーソルを戻すのはさほど掛かっていない．次に，KL モデルによる操作系列は次の通り．

$$KKP_{10.5,7}KP_{10.5,6}K$$

最初の 2 つの K は，メニューを出すための Control キーとマウスボタンの押下げに対応している．次の $P_{d,s}$ は距離 d 離れた大きさ s の目標への移動を現す (距離の単位は mm)．KL モデルではマウスボタンを押して離すのを 1 つの K としているが，この実験の場合はボタンを押したまま別の操作をし，改めて離すことからマウスボタンを離す操作も独立した K(3 番目のもの) として考えた．次の P は次の対象アイコンまで戻る操作，最後の K はその対象アイコンを選択するものである．

ここでポインティング時間は 3.1 節の式により，打鍵

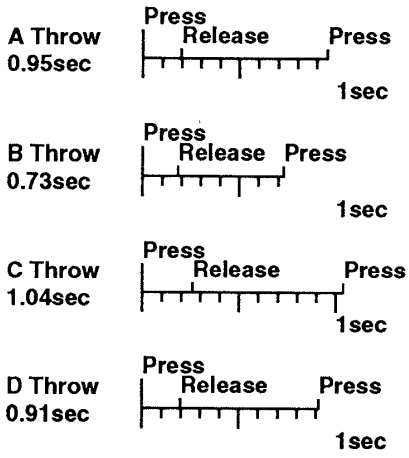

図6 「投げる」操作のタイミング

時間を一律 200 msec とすると，予測は 200 + 200 + 300 + 200 + 320 = 1220 msec であり実測値とよく合う (被験者 A では「よける」分の移動 $P_{16,16}$=358 msec を加えると，これもおおむね合っている)．

5.2 パレット操作 (図 3)

対象アイコンの上でマウスボタンを押して離し，続いてパレットの上に移動してそこでマウスボタンを押して離し，その後元の位置に戻って次の対象アイコンの上でマウスボタンを押す．操作系列は単純だが，アイコンとパレットの間の移動に時間が掛かるため速くはない．KL モデルによる操作系列は

$$P_{78,15}KP_{78,16}K$$

であり，予測される時間は 550 + 200 + 540 + 200 = 1490 msec となる．これもよく合うといえる．

5.3 キー操作 (図 4)

対象アイコンの上でマウスボタンを押して離し，続いてキーを押し，再び次の対象アイコンの上でマウスボタンを押す．KL モデルによる操作系列は

$$KK$$

だけなので 400 msec となり，過少である．

これは思考時間に関係していると思われる．すなわち，今回の実験では，設定が「A を A に重ねる」という単純なものであるので，思考時間を考慮していない．しかし，実際にはこの思考時間はマウス移動操作とオーバラップして隠されているだけであり，KK のような短い系列になるともはや隠されることができずに現れてくる

と考えればつじつまが合う．思考時間のモデル化については今後の課題であるが，例えば木村らによる「少数の選択肢からの選択」オペレータ [5]のようなものが適用できるかもしれない．

5.4 ドラグ & ドロップ操作 (図 5)

対象アイコンの上でマウスボタンを押し，そのまま移動して目標アイコンの上で離す．操作系列は

$$P_{128,16}KP_{128,16}K$$

であり，予測時間は 610 + 200 + 610 + 200 = 1620 msec とやや大きいが，まあ合っている．

5.5 「アイコン投げ」操作 (図 6)

操作の系列はドラグ & ドロップと同じだが，途中でボタンを離すため，ボタンを押してから離すまでの時間がずっと短い．「投げる」操作は KL モデルに含まれないため KL モデルでは予測できないが，これを仮に一定時間を要する T オペレータと考えれば，操作系列は

$$TP_{d,16}K$$

となる．ここで T の時間はボタンが押されてから離されるまでの時間だと考えると，図 6からは 200 ～ 250 msec であると読み取れる．この値はキーボードの打鍵時間とさほど変わらない値であり，「投げる」操作の効率性を表しているものと考える．

6 議論

本節では，前掲の実験結果と分析に基づき，「アイコン投げ」インタフェースの有効性について検討する．

6.1 操作時間の優劣

まず，表 1 にあるように「アイコン投げ」操作はキー操作に続いて高速であり，メニュー，パレット，ドラグ & ドロップのいずれと比べても有意に速く操作できる．また，今回の実験設定ではキー操作は

- マウスを全く動かす必要がない．
- 選択内容とキーが表すシンボルが一致している．
- 使用したキーが A ～ D であり，左手をホームポジションから動かすことなく操作できる．

という点で有利であり，「アイコン投げ」がそれとさほど遜色ない操作時間を達成したのは印象的である．

なお，すべての対象アイコンが同じ位置に重なっているという実験設定は (マウス移動が最小限で済むため) メニュー操作にも有利なはずであるが，実際にはメニューはパレットやドラグ & ドロップと同程度かややよい程度の速度しか達成できていない．これら 3 つについても，KL モデルによる予測との適合性から見ても，これ以上の高速化は難しいであろう．

6.2 実験条件に関する議論

まず，今回の設定では選択肢の数が 4 という比較的少ない数だったが，もっと選択肢を多くした場合についても検討する必要がある．とりあえず，参考のために選択数を 8 にして被験者 C のみで予備的な実験を行ってみた．この実験では「アイコン投げ」のみを連続して行ったので，必ずしも表 1 と同等に比べられないが，めやすにはなる．その結果は，サイクル所要時間の平均が 1.167 秒，標準偏差 0.11 であり，時間的には「やや遅くなった」程度である．この遅延が思考時間によるかどうかの検討は今後の課題である．選択誤り率は 15.6% で，4 個の場合の 3 倍に増えた．これは予想よりやや多かったが，この程度であれば実用に耐えると考える．

次に，上とも関連するが，アイコン群の配置が比較的簡単化されていて，現実のデスクトップインタフェースとはだいぶ異なるという点も問題である．今回の実験ではアイコンの大きさや距離を同一にすることで解析を容易にしたが，これを補う意味で後述の「アイコン投げシェル」のような実用インタフェースを用いた評価を今後計画している．

最後に，筆者らが被験者を兼ねたことへの疑問も当然あろうが，第 1 筆者の環境では十分な被験者を得ることが難しく，とりあえず作成者らによる評価を行ったものである．その後，第 1 筆者の所属する社会人大学院で学生 7 名を対象に同様の実験を行ったが，「アイコン投げ」はメニュー，ドラグ & ドロップ，パレットのいずれよりも速いという結果を得た (キー操作よりも速かったが，これは学生がタッチタイプできていないためと思われる)．この実験は制御が十分でなかったため詳細な分析は行えなかったが，筆者らの独断と偏見ではない一般性をかいま見ることができよう．

6.3 「アイコン投げ」のトレードオフ

上述の比較は純粋に操作時間のみを比較したものであり，公平さのためには「アイコン投げ」と他の操作選択方式の特性の違いについての考察も必要である．具体的には次の2点について検討する．

- メニューやドラグ＆ドロップでは最後までマウスカーソルの制御を保持するので，階層メニューや，マウスカーソルが入ると自動的にフォルダが開いてその中の要素へのドロップを許す，などの形で複数レベル選択が可能だが，「アイコン投げ」では途中でカーソルの制御を放棄するためこれができない．
- メニュー，パレット，キーによる選択は，一度選択したアイコンに対して複数の操作を連続して指定する際に操作の一部を省略できるが，「アイコン投げ」はそのような性質がない．

まず前者について，例えば3レベルの階層メニュー選択の操作系列はKLモデルで次のように書き表せる．

$$PKKPPPK$$

ここで3つの連続するPが3レベルの各メニュー内の位置づけに対応する．その所要時間はメニューの配置によって変わるが，仮に今回の実験設定と同様だとすると，各レベルについて150～300 msecのオーバーヘッドとなる．これに対し，「アイコン投げ」では3回「投げる」必要があるが，階層メニューのように次の選択が必須であるものへ投げた場合には，アイコンが自動的にマウスカーソルの位置に戻り，直ちに次の選択へ向かって投げられるようにできる．その場合の操作系列は

$$PKTTT$$

となるものと予想され，Tの所要時間が200～250 msec程度であれば，階層メニューに比べて特に不利ではない可能性が大きい．

次に後者についてであるが，連続して操作選択を行う場合のメリットは対象アイコンを選択するP操作が省略できることであり，そのために節約できる時間は今回の実験設定では200 msec程度である．今回の実験結果では「アイコン投げ」はキー操作以外に対しては200 msec以上速いので，これによって優劣が逆転することはない．これらの議論はあくまで予測に基づくもので，今後実験による評価が必要なのは当然である．

6.4 「アイコン投げ」の特性

分析時に気づいたことの1つに，投げる時にボタンを離す位置までの移動距離が被験者A，C，Dで60 mm程度，被験者Bで25 mm程度と異なっていたことが挙げられる．これと被験者Bの所要時間が他の被験者より短いことは何らかの関連があると予想でき，利用者が短い移動距離で投げられるように修練を積むことで所要時間も短縮できる可能性がある．

ただし，「投げる」場合にはマウスを動かした状態でボタンを離すので，投げ終わった時のマウス位置はボタンを話した位置よりも目標アイコンに近付いている．したがって，Tの正味の所要時間は先に述べた値よりもう少し長いと考えるべきであろう(今回の実験ではマウスボタンを押していない状態でのマウス位置は記録しなかったため，実際にどこまでマウスが行ったかは後からは分からなかった)．

反面，今回の実験では投げた後必ずマウスを元の位置に戻していたが，実用の場面では投げた後「向こうの方に」次の対象アイコンがある場合もあり，その場合には全体の時間は短縮される．

表2を見る限り，「アイコン投げ」の誤り率は，メニューの選択間違いより少ない．ただしメニューは間違えては困る場合にはゆっくり慎重に操作できるが，「アイコン投げ」ではかなり熟練しても一定の投げ損ないが避けられないであろう．これについてはundo機能など周辺のサポートでカバーすべきと考える．

なお，投げるインタフェースはそれ自体がアニメーションなので，メニューやボタンのように選択したものを点滅させるなどの工夫を設けなくても間違いの認識は容易である．また，どうしても間違えたくない場合には，そのままドラグ＆ドロップに移行できる．

6.5 「アイコン投げシェル」の構想

今回の実験では「アイコン投げ」操作自体の効率について検討したが，それが実用のユーザインタフェースにおいて有効かどうかは別の問題である．そこで次の段階として，筆者らが過去に関わった[2]，[3]などの設計を下敷きに，X-Window上でMacintosh Finderのような汎用の作業をこなすツール(「アイコン投げシェル」と呼んでいる)の開発を計画している．

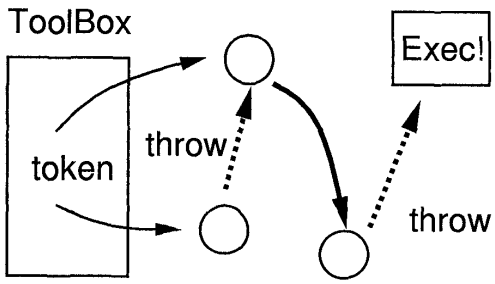

図7 「アイコン投げシェル」の概念

その基本的なアイデアは図7のようなものであり，「道具箱」と呼ばれる領域に多数のトークン(語)が蓄えられ，それをデスクトップにドラグすると(実線)，自立したアイコンとなる．そしてアイコンを別のアイコンに投げつけると(点線)，複数のトークンが接続された新しいアイコンが生成される(太線)．これを繰り返して，例えばUnixのコマンド行のようなものを組み立て，完成したコマンドを「実行」という特別なアイコンに投げつけて実行させる．コマンドアイコン自体は元の位置に戻るので，繰り返し実行可能である．

この枠組みでは，アイコン投げの特徴を活かすために，次のような工夫が取り入れられている．

- トークン同士を投げるのは至近距離同士で行えるので，途中に障害物があって投げられないということが起きにくい．
- 「実行」「ごみ箱」など特によく投げつけるアイコンは，画面中央に投げやすく配置できる．
- よく使うアイコンは自然にデスクトップ上の投げやすい位置に集まると予想される．

6.6 多数のアイコンを使用する場合の可能性

実用に使う場合であれば，常に目標アイコンが投げる位置から見てうまく横に並んでくれるとは限らない．そのため，手前のアイコンを「よけて」向こうにある目標アイコンに到達させる方法も必要であろう．そのためのアイデアとしては，

- 投げたアイコンが窓の周囲で「はね返る」ようにして，クッションを利用して到達させる．
- 投げる時の軌跡の方向ベクトルの角速度を記憶しておき，それに従って投げられたアイコンもカーブやシュートさせる．
- 画面上にも地上と同様の重力を仮定し，投げたアイコンが放物線を描いて飛ぶようにする．
- 飛んでいったアイコンが目標アイコンに「つかまる」ための操作を明示的に(キーボードなどで)行う．ただし，これを行うとオーバラップして投げることは難しくなる．
- 「摩擦」を導入して，投げたアイコンがだんだん遅くなるようにし，ある程度以上の速度であれば目標アイコンにつかまらずに通過するようにする．
- 速度がある値以上だと目標アイコンにつかまらずに通過するが，一定のエネルギーを吸収されて速度が落ちる．したがって，強く投げるほど途中のアイコンを通過して遠くのアイコンまで到達する．

などを考えている．これらのいくつかを試みに実現したが，操作した感覚では「重力」と「エネルギー吸収」が有望に感じられた．

また，実用的には「投げにくい」場合の不利さはそれほど問題にならない可能性もある．というのは，先にも述べたようにドラグ & ドロップではアイコンの配置を変更すること自体がカスタマイズであり，したがって，特によく使う操作に対応する目標アイコンを少数選んで投げやすい位置に置くという状況を想定すれば，あえて投げにくい時に投げる必要はあまりないかもしれないからである．「アイコン投げシェル」の評価の一環として，これらの事項についても検討していく予定である．

6.7 類似研究

今回提案した方式は純粋にソフトウェア的な実現を前提としていたが，文献[6]に提案されている慣性マウスのように，マウスのハードウェアを拡張することで「投げる」動作を実現しているものもある．これはマウスの下面にスイッチを設け，マウスを物理的に持ち上げると，持ち上げている間ずっと持ち上げる直前の方向/速度の移動が続いているかのような信号を送出するものである．この方式の利点はソフトウェアの変更が不要で任意のソフトウェアに適用できる点であるが，「引力」などのアイデアは適用できない．

一方，文献[6]によれば慣性マウスの利点は

- 遠くの対象を指示する時でもマウスを大きく移動しなくて済む．
- マウスをマウスパッドの中央付近に置いておける．

点に由来するとのことだが，これらはアイコン投げにおいても同様である．ただし，今回の枠組みではアイコンは投げられるがマウスカーソルは投げられないので，対象アイコンを選択する時には通常のポインティングになる．この点を改良してカーソルまで投げられるようにすべきかどうかも今後の検討課題である．

7 まとめ

GUIにおいて複数の選択肢からの選択を行うのに，メニューと並んで有力な方法であるドラグ＆ドロップの操作時間を改善するため，「アイコン投げ」と呼ばれる操作方式を提案した．その操作時間を評価するための実験の結果，「アイコン投げ」方式は条件によってはメニュー，パレット，ドラグ＆ドロップより高速であり，誤り率も少なく，実用に耐えるとの感触を得た．

今後の課題としては，「アイコン投げ」の操作時間を説明するようなもっと細密なモデルを組み立て検証すること，および前節で挙げた操作性を改善するアイデアとそれを組み込んだ「アイコン投げシェル」の実用性評価が挙げられる．

参考文献

[1] Graphical User Interfaces: The Next Generaiton, Special issue, CACM, Vol. 36, No. 4, 1993.

[2] 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.

[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 マシンのユーザインタフェース, 第 29 回プログラミングシンポジウム報告集, pp. 13–22, 1988.

[4] Card, S. K., Moran, T. P., and Newell, A.: The Psychology of Human-Computer Interaction, Lawrence Erlbaum, 1983.

[5] 木村泉, 粕川正充：ワープロ利用者の思考時間に関する統計的模型の精密化, 情報処理学会文書処理とヒューマンインタフェース研究会 14-4, 1987.

[6] 野中秀俊, 伊達淳：慣性機能を持つポインティング装置の開発, 情報処理学会論文誌, Vol. 31, No. 2, pp. 268–274, 1990.