絵はねずみなしでも描けるか?

久野　靖† 角田博保‡ 大木敦雄† †筑波大学大学院経営システム科学専攻
‡電気通信大学情報工学科

概要

計算機を用いて作画を行なうソフトウェアの大部分はマウスを利用する設計になっているが、これは必ずしも最善のやり方とは限らない。本稿では著者らが開発した、キーボードのみを使用して作画を行なうツールの設計および使用経験について述べる。また併せて実施した、マウスによる作画作業とキーボードのみの作画作業の比較実験について報告し、用途によってはキーボードのみで作画する方式が優れている場合も存在することを示す。

1　はじめに

高解像度の画面表示装置や印刷装置が普及するにつれ、計算機を用いて作成する文書も文字だけでなく絵 (ないし図) を含んだものが多く見られるようになっている。これらの絵は写真、画像、その他従来の媒体から変換装置を用いて計算機に取り込まれたものである場合もあるが、文書作成者自身が計算機上で稼働する作画ソフトウェアを用いて描いたものである場合が圧倒的に多い。これには例えば次のような理由があると考えられる。

1. 計算機上でワードプロセサやエディタを用いてテキストを打ち込むのと並行して作画ソフトウェアを使用して絵を作成する方が、時間や場所を改めて別の方法で絵を作成するより効率がよい。
2. 計算機で文書を作成するような利用者にとっては作画ソフトウェアの類を使うのに抵抗が少ない。
3. テキストと絵を同じ計算機上のファイルとして保管できるので散逸の心配がなく、管理が楽である。
4. 計算機で文書を作成するような利用者はフリーハンドで絵を描くのが下手なので計算機の助けを借りる方を好む。
5. 作画ソフトウェアによって作成されたファイルは論理的な図形の配置を記すので、変換装置によって読み込まれた絵よりもコンパクトである。
6. 計算機上で文書を作成するメリットの 1 つは後日内容を改訂するのが容易な点であるが、作画ソフトウェアによる絵にも同様の利点がある。

なお、作画ソフトウェアについて見ると、その方式により次のように分類できる。

a. 直線、長方形、楕円など予め用意された種類の図形を様々な大きさで生成し、組み合わせ配置して絵を構成するもの。例えば MacDraw がこの代表である。以下では「ドロー方式」と呼ぶ。
b. 直線や円などを描くツールも用いるが、基本的には絵を構成する各ピクセルの色を制御することを通じて絵を構成するもの。例えば MacPaint がこの代表である。以下では「ペイント方式」と呼ぶ。
c. 絵は決まった大きさの文字の集まりで構成され、その文字の種類として (罫線素片などのように) 絵を構成するのに適したものを活用する。多くの日本語ワードプロセサに見られる罫線機能や、一部のテキストエディタに存在する作画モードで絵を描く場合が相当する。以下では「罫線方式」と呼ぶ。

前掲の利点を全て備えているのは実はドロー方式の場合であり、ペイント方式については 1〜3、罫線方式については 1〜5 のみがあてはまる。ペイント方式ではフリーハンドの絵が描け表現の自由度は非常に大きい代わりに 1 枚を作画するのに要する手間は一般に大きい。また直線などの図形も一度描いてしまうとピクセルの並びになってしまうので後から位置や色を変更するのは大変である。一方、罫線方式はワードプロセサなど本来文字のみを扱うソフトウェアに組み込むのが容易であるが、限られた文字の組み合わせのみなので

[0] "Drawing without Mice" by Yasushi KUNO†, Hiroyasu KAKUDA‡ and Atsuo OHKI†, †Graduate School of Systems Management, University of Tsukuba, Tokyo and ‡ Department of Computer Science, University of Electro-Communications.

表現の自由度が小さく、一度描いた図形は文字の並びになってしまうため位置などの修正が大変な点はペイント方式に近い。

以上のような理由から計算機を用いて作成する文書に絵を含める場合にはペイント方式が使われることは少なく、ワードプロセサやエディタの枠内だけで描くという制約がある場合には罫線方式、そうでない大部分の場合にはドロー方式が用いられる。以下本稿でもドロー方式による作画を対象として取り上げる。

ドロー方式による作画ソフトウェアは非常に多数存在するが、そのほとんどは作画のためにマウス、トラックボール、タブレットなどの位置指示装置を使用する。これは、作画における主要操作の 1 つが各図形の画面上の配置指定であることを考えれば一見当然のように思える。

しかし、作画ソフトウェアにおいて必要とされる操作は他にも多数存在し、画面上の配置指定はあくまでもその 1 つに過ぎない。具体的には操作群として以下のものが考えられる。

- 図形を配置したり移動したりする位置を指定する。
- 配置する図形の種類を選択する。
- 図形に使用する線の種類や太さを選択する。
- 図形の塗りつぶしのパターンや色を選択する。
- 図形が「テキスト」の場合、その内容を入力する。
- テキストのフォントや大きさなどを選択する。
- 既に配置されている図形の中で次にどれを操作するかを指定する。
- その他、ファイルの読み書きやモードの設定などの補助的な操作全般。

こうして見ると、これらの操作全般についてマウスを使用することが適切かどうかは疑問がある。

特にこれらの大部分を占める「選択」操作について考えてみる。マウスによる (各種のメニューまたはパレットを使用した) 選択操作は直感的に分かりやすく、初心者でも迷わないという利点を持つが、反面次のような弱点も持つ。

- パレットやプルダウンメニューの場合、現在作画している位置からパレットやメニューバーまでマウスを移動させるのに時間が掛かる。
- どの種類のメニューでも、メニュー表示操作を開始し、メニューが表示されるまで待ち、選択対象の位置を認識し、そこまでポインタを移動するという手順が必要なため、全般に時間が掛かる。

一方、もう 1 つの標準的な入力装置であるキーボードについて考えると、上述の操作のうちテキスト本体の打ち込みに適するのはもちろんであるが、指令やオペランドの形式を記憶することさえ厭わなければ、それらを打ち込むことによって多数の選択肢から 1 つを選ぶ操作を高速に行うことができる。従って、図形の配置や選択をキーボードで効率よく行うことさえできれば、全体としてマウスより効率よく作画を行えるようになる可能性さえある。

実際、筆者らは以前に、ウィンドウシステムにおける窓操作をキーボードのみで行う方式を提案・実装し、普段手がキーボードにあるという前提のもとではキーボードによる窓操作は一般に行なわれているマウスによる窓操作より有利であることを示した [1][2]。もちろん、窓を選択・配置することと図形を選択・配置することは同列には論じられないが、少なくともキーボードのみによる作画ソフトウェアが有効である可能性はあると思われた。

以上のような考えに基づき、筆者らはキーボードのみで全ての操作が行える作画ソフトウェア `k2d` [1] を開発し、評価を行なった。

本稿ではその設計、使用経験、評価実験ための実験および結果について報告する。以下、2 章では k2d の設計方針とコマンド体系について述べ、その発展の経緯を初期の使用経験と併せて報告する。続く 3 章では k2d を従来のマウスによる作画ソフトウェアとして使用した場合とキーボードで操作した場合の優劣を調べるための実験について報告し、4 章でまとめをおこなう。

2 k2d の設計とコマンド体系

2.1 基本方針

本節では筆者らが開発した、キーボードのみで操作可能な作画ソフトウェア k2d の設計とコマンド体系について述べる。まず、k2d の開発に当たっては、まったく新規に作成するのではなく、既存の作画ソフトウェアを下敷きにキーボードによる操作機構を組み込むことにした。その理由は次の通りである。

[1] kdraw[3] をキーボードのみで操作できるようにしたものなので最初は keyboard kdraw と呼んでいたのだが、長々しいのでこう呼ぶようになった。

- 既に広く実用に使われているソフトを下敷きにすれば、少なくともそのソフトによる絵が役に立つようなものであることは認めてもらえる。
- 下敷きとなるソフトはマウスで操作するものなので、キーボード操作版ともとの版で比較実験を行うことで比較対象の差異をマウス操作とキーボード操作の差異のみに限定できる。
- 下敷きとなるソフトが広く実用に使われているものであれば、品質の劣るマウス操作ソフトと比較したのではないかという疑念を受けないで済む。
- そしてもちろん、ゼロから新規に作るよりも労力が節約できる。

具体的には、Unix 上の X-Window(Release 4, 5) で稼働する作画ソフトである idraw[4](interviews[5] version 2.6 付属のもの) の漢字版 (kdraw[3]) を改造の対象とした。これは、ドロー方式ソフトの代表である MacDraw に準じた設計になっていること、筆者らが日頃使用していて十分実用的だと考えていること、実際に多くの利用者がいること、フリーソフトウェアであるため改造の前提であるソースが入手できること、X-Window 上で動くため [1][2] の時と同様の方法で時間計測が行えること、C++で書かれているためモジュール構造が明確で改造が容易であると判断したこと、などによる。

2.2 指令の設計 (1)

2.2.1 マウスで行われる操作の検討

k2d の仕様設計については次のように進めた。まず下敷きとなった idraw は (パターンや色などの選択を除く) すべてのメニューコマンドおよびパレット選択はキーボードショートカットを持ち、キーボードから 1 文字入力することで起動できる。そこで、これら以外の操作でマウスでなければ行えない操作を検討し次のものを列挙した。

- 図形の選択。マウスの右ボタンで図形をつつくと、その図形が「選択」された状態になる。またシフトキーを押しながら同様にすると、これまでの選択にその図形が追加される。または、図形のない所で右ボタンを押し、そのままマウスを動かして長方形の対角線を描くことにより、その長方形に含まれる全図形をまとめて選択することもできる。
- 図形の移動。マウスの中ボタンで図形を「つかんで移動」することでボタンを離す位置までその図形を移動することができる。多数の図形が選択されていればそれらがまとまって移動する。
- 図形の変形など。変形、スケール、大きさ変更などのパレットを選んだ状態でマウスの左ボタンにより対象とする図形の頂点や辺や制御点を「つかんで移動」する。[2]
- 図形の作成。パレットでそれぞれを選んだ状態で左ボタンにより行う。直線、長方形、円などはその一端、一頂点、中心でボタンを押し、そのままでマウスを移動して他端、対角をなす頂点、円周上の点でボタンを離すことにより指定する。折れ線、スプライン、多角形ではその各点を左ボタンのクリックで指定し、最後の点は中ボタンのクリックで指定する。
- ショートカットのないメニュー選択。線の種類、文字フォント (漢字、英字)、パターン、色 (前景、背景) の 5 種がある。

そこで全体的な方針として、まずこれらをともかく一応キーボードのみで行えるように改造し、それを試用することでよりよい方法の着想を得る、というサイクルを反復することで仕様を定めるやり方を採った。以下では比較的初期の段階の上記各種操作に関する仕様をまとめる。

2.2.2 図形の選択

最初は図形の選択であるが、これをキーボードで行うためには「どの図形を」+「選ぶ/選択に追加」、という情報を打ち込む必要がある。そこで次の 2 つの指令を用意した。

番号 `s` -- 番号で指定される図形を選択
番号 `a` -- 番号で指定される図形を選択に追加

「番号」の意味は、図形の重なり順で最も上にあるものが 1、次が 2、... ということにした。従って、新しい図形を作るごとにこれまでの 1 番は 2 に、2 番は 3 に、という風に繰り下がることになるが、その代わり最近に作った 3 個を選択したければ

[2]図形の選択や移動もパレットにあり、それらを選択した状態で左ボタンを使って同様に操作してもよい。

```
1s2a3a
```

で行える。なお、番号は当然数字で入力するのだが、もとの idraw では数字もコマンドのショートカットになっていた。それでは困るので、数字についてはコマンドには対応せず、これまで打ち込んだ数字が累積され (これを数値プレフィクスと呼んでいる)、その値がツールの窓の右上隅に表示されるように改造した。s や a など番号を参照する指令はその値を読み出した後、プレフィクスを 0 にリセットする。

2.2.3 図形の移動

次に移動であるが、これは

```
m
```

という指令で移動モードに入り、その中で h、j、k、l の各キーを押すと現在選択されている図形 (群) がそれぞれ左、下、上、右に 1 グリッドずつ移動するものとした。なお、常に 1 グリッド (初期設定では 8 ドット) ずつ動くのではカーソルキーと同じでまだるっこしく、また微調整のためより細かく動かすこともあると考えたので、修飾キーを使用して次のように指定できるものとした。

```
なし      -- 1 グリッドずつ
Shift    -- 4 グリッドずつ
Control -- 1/4 グリッドずつ
Shift+Control -- 1/16 グリッドずつ
Meta      --- 1 ドットずつ
```

これらのキーにより目指す位置まで来たら、[RET] キーを押すと位置が確定して移動モードを終わる。

2.2.4 図形の変形等

図形の大きさ変更の際には、まず

```
r
```

という指令で大きさ変更モードに入ると図形を囲む四角形 (Bounding Box、以下では bbox と記す) が表示される。その後は移動モードと同様に h、j 等のキーを使用すると囲む四角形の右下隅が移動し、[RET] キーで終了するとその時点での四角形の形に応じて選択されていた図形の大きさが変化する。移動する点を四角形の右下隅に固定したため、マウス操作と比べて図形の下辺や右辺を固定したままでの大きさ変更ができないという不利はあるが、実用上はほとんど問題ない。

もう 1 つの変形方法として、図形の制御点 (長方形や多角形では頂点、直線や折れ線では端点や折点) のうちどれかを移動するというものがある。この場合は「どの点か」という情報を与える必要があるので、

```
番号 q
```

により、点の番号を指定し、あとは m と同様のキーでその点の位置を移動する。点の番号付けは直線では始点が 0 で終点が 1、長方形では左下から時計回りに 0、1、2、3 となっているが、特に表示される訳ではないのでやや苦しいところである。しかし実際に使用してみると比較的素直に数えて当たるようである。

idraw ではこれらの他に回転や縦横比を変えないでの大きさ変更なども行えるのだが、筆者らの経験ではめったに使用しないのでこれらのための指令を増やすことはしなかった。(もともとメニューコマンドの中にスケール値や回転角度をキー入力させてこれらを行う指令があるので、それらをショートカットで呼び出せばキーだけでできてしまうことも理由であった。)

2.2.5 図形の新規作成

図形の新規作成は、その図形の位置と形をともに指定する必要があるので、これまでの指令よりだいぶ複雑である。まずどのような図形を生成するかはパレットで選択するわけだが、これはもともとショートカットにより行える。その後実際に生成を行う指令は

```
n
```

で、これにより「とりあえず」適当な位置に作成される図形の bbox が表示される。この「適当な位置と大きさ」は当面次のようにした。

- 現在選択中の図形がない場合 — 画面の左下隅に、幅高さとも 4 グリッドの大きさ。
- 現在選択中の図形がある場合 — その図形 (群) の bbox と同じ大きさで、下と右に 1 グリッドずれた位置。

ただし直線、折れ線、多角形などの場合は bbox の代わりに bbox の左上隅から右下隅への直線が「最初の線」として現れる。その後、まず移動モードに入って図形の位置を移動し、配置したい場所に持っていく。この

キー操作は上記の移動モードと同じである。[RET] で場所を確定したあとの操作は図形の種類によって異なり次の通りである。

- 直線 — 変形モードと同様にして終点の位置を動かし、[RET] を打つと作成が完了する。
- 長方形、楕円、角のまるい長方形 — 大きさ変更モードと同様にして囲む四角の左下隅を指定し、[RET] を打つとその大きさで作成される。
- 折れ線、スプライン曲線、多角形、スプライン閉曲線 — 最初の折れ点 (制御点) までは直線と同様で、その後 [RET] を打つごとに折れ点 (制御点) の位置が定まるとともに新しい折れ点 (制御点) がその場所に重なってできるので、再び移動キーで位置を指定する。任意の点で [ESC] を打つとそこが最後の点になって図形が作成される。

この他にテキストがあるが、テキストの場合は n を打つと位置指定は行なわずいきなり選択箇所の近辺でテキスト入力の状態になる (テキストは数行に渡ることができるので、打ち込みの終了は [ESC] で示す)。これは、他の図形があるべき場所にないと描きにくいのに対し、テキストの場合には打ち込む内容は場所と関係なく決まっている場合が多く、とりあえず適当な場所にできて後から移動する方が使いやすいためである。

2.2.6 ショートカットのないメニュー選択

これは簡単で、例えば線種メニューの場合だと

番号/

でメニューの上から n 番目の線種を選ぶことができる、という風になっている。

2.3 使用経験と改良

2.3.1 最初の使用経験と問題点

このようにして一応全ての作画ができるようになったので、とりあえずこれを用いていくつか図を描いてみるなどの試用を行ったが、この段階ではマウスを使う場合にくらべて非常に使いづらく感じた。具体的には次のような問題点があった。

- 図形の選択をすべて番号で指定するのがとても煩わしい。新しい図形を加えるととたんに番号がずれてしまうので特にそうである。
- 位置を指定するのにキーを何回も使用するため、時間が掛かりいらいらする。修飾キーの組み合わせが多数あるが、混乱するのでほとんど使われない。

そこで、最初の改良として以下に示す機能を追加した。

2.3.2 図形への名前つけ

まず、図形の番号が変わってしまうことに対する対策として、図形に文字列の名前をつけることにした。対案として図形の番号を常にできた順にすることも考えたが、これだと「最も最近できた図形」「その次の図形」というこれまで使えていた指定方法が使えなくなること、複数をグループとしてまとめて 1 つの図形にするとそれはまた別の新しい番号になるので混乱すること、などから採用しなかった。具体的には名前をつけたい図形を選択した状態 (普通は生成した直後) に

=英数字から成る文字列 [RET]

でその文字列が図形の名前として登録される。また

’名前の文字列 [RET]

によりその名前の図形を選択状態にすることができる。

2.3.3 「ひっかかり」による移動

これは kwm[1][2] の窓移動において採用していた方式であり、例えば図 1 のようにある図形を右に移動するとして、移動する図形の左端、右端、中央および他

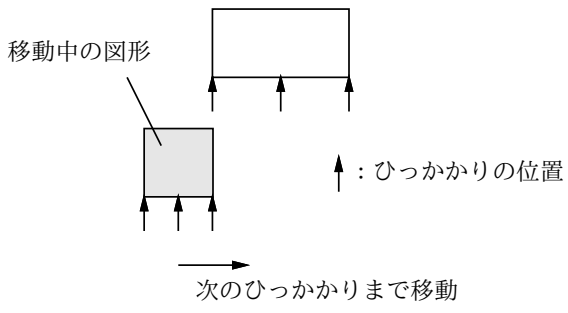

図 1: ひっかかりによる移動

の図形の左端、右端、中央に互いに「ひっかかる部位」があり、移動方向で最初にその「ひっかかり」が起きた所で止まる、という風にする。これにより図形どうしを互いに接したり、揃った位置に容易に置くことが

できるわけである (ただし窓どうしを中央揃えしたい、ということはまずないので、kwm では中央のひっかかりはなかった)。

また、修飾キーの組合せを多くしても結局使えないという反省から、組み合わせを次のように整理した。

```
なし      -- 1 グリッドずつ
Shift     -- 4 グリッドずつ
Meta      -- 1 ドットずつ
Control -- 「ひっかかり」による移動
```

2.4 引き続く使用経験と問題点

この段階で予備的な実験として、筆者らそれぞれが後述の実験に使用したような絵をキーボード操作とマウス操作の両方で描いてみて、所要時間をストップウォッチで測ることを行なってみた。しかし、結果はキーボード操作とマウス操作の所要時間の比率はならしておおよそ 3:2 でキーボードの方が遅い、というあまり思わしくないものであった。この際に指摘された問題点としては次のようなものが挙げられた。

- せっかく「図形の名前を定義する」という機能をつけたのだが、実際に使ってみるとあまりにも煩わしくほとんど価値がない。
- 図形の選択方法として、「現在選択中の図形の上/下/右/左の図形」、「同じく、重なり順で次/前」というのが欲しい。その方がエディタなどで編集位置を指定するのに類似しているから。
- 図形を移動する際、移動モードに入るのを忘れていきなり移動キーを押すという誤りが多発する。
- ひっかかり方式を採用したものの、図形の数が少し多いとひっかかる位置が多過ぎて何回もキーを押すことになり、効果がない。さらに 1 回押すとどこまで進むか予測しにくいため結局単純に 1 グリッドずつ進む方が使いやすい。
- このため、移動の際に打鍵数が多くて時間が掛かるという問題点が依然として解消されていない。
- 同一の図形を複数描く場合には最初の 1 個を複製 (duplicate) 指令でコピーしてはそれぞれの場所に配置するのが普通だが、複製すると元のものから 1 グリッドずつ右下にずれた位置にコピーができる。マウスで操作する場合はこれで問題ないが、キーボードだと「元の位置から n グリッドどちらへ」配置しようと思うと一担元の位置まで戻してから改めて移動することが多く煩わしい。
- 図形を新規作成する際、現在選択されている図形の位置と大きさを基準にする、という方式でちょうどよい場合もなくはないが (たとえば図形の下に説明を付加する場合) それはごくまれで、とりあえず出来た状態から求める位置と形になるまで修正するのがひどく大変である。

これらの経験をもとに、次節で述べるように指令の機能を大幅に変更した。

2.5 指令の設計 (2)

2.5.1 移動指令の追加と改良

まず、移動に際して m 指令を打たずにいきなり h、j などの各移動キーを押してしまうことが多かったので、m 指令とは別にこれら 4 つを「その方向専用の」移動指令として追加した。

```
h -- 選択中の図形を 1 グリッド左へ移動
j -- 選択中の図形を 1 グリッド下へ移動
k -- 選択中の図形を 1 グリッド上へ移動
l -- 選択中の図形を 1 グリッド右へ移動
```

またこれらの指令の前に数値プレフィクスをつけた場合にはそのグリッド数だけ移動するようにした。m 指令の中での h、j などでもその前に数字を打った場合にはそのグリッド数だけ移動するよう改めた。

次に、m 指令での「ひっかかり」モードでひっかかりの数が多過ぎたのを、例えば左右の移動の場合で次のように整理した。

- 移動しつつある図形の左右端いずれかと他の図形の左右端いずれかが一致する所にひっかかりがある。
- 移動しつつある図形の中央と他の図形の中央が一致する所にひっかかりがある。

これは、「移動中の図形の端と別の図形の中央が揃う」などの場合はまず使われないと考えてこう改めたものである。逆にいうと、新しい方式でひっかかるのは移動中の図形と他の図形が「接する」または「中央揃えの関係にある」場合だけであり、後から考えてみれば実用的にはこれだけでほぼ十分なのは明らかであった。

2.5.2 現図形を基準とした選択方式

既に述べたように図形番号や図形名による選択には問題が多かったので、これらに加えて現在選択している図形を基準とした選択も行えるように次の指令を追加した。

```
^H  -- 現図形の左で最も近い図形を選択
^J  -- 現図形の下で最も近い図形を選択
^K  -- 現図形の上で最も近い図形を選択
^L  -- 現図形の右で最も近い図形を選択
^N  -- 図形番号が現図形の前のものを選択
^P  -- 図形番号が現図形の次のものを選択
```

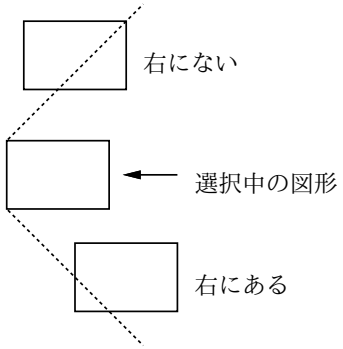

図 2: 右にあるかどうかの判定

図形の相対位置に関する判断は何通りかのやり方を試した結果、例えば「右にあるかどうか」の判定は図 2 に示すように「左辺が現図形の左辺より右にあり、なおかつ 45 度の範囲内にあるかどうか」のようにしている。また、図形番号について否定的な判断を述べた後で^N と^P を追加しているのは矛盾しているように思われるかもしれないが、例えば^P は「現図形より 1 つ前に作った図形」[3]なので、ある図形を作った後ラベルなどをそれに付随させ、その後またもとの図形を選択して作業を続けたい時などに有用である。また、図形を作った直後で^N を使うと (最後から最初へ回るようになっているので) 最初に作った図形を選択することができ、これも役に立つことがある。

2.5.3 現図形を基準とした複製方式

idraw で複製がもとの図形の位置からずれたところに置かれるのは、もとの図形と全く重なると複製できたかどうか確認しにくいためだが、前述のようにキーボードで操作する場合にはこれは必ずしも使いやすくない。そこで d 指令は標準ではもとの図形と同じ位置に複製を置くようにした。

これに加え、複製を「もとの図形のどちら側に隣接させて」と言えるようにした。「どちら側」は数値プレフィクスを使って、図 3 のように番号で指定する。従ってある図形を横にずらっと並べたい場合には (d 指令では複製された図形が新たな選択となるので) その図形を選択した状態で「1d1d1d1d...」と打てばよい。ま

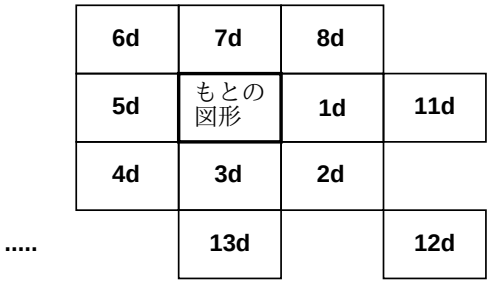

図 3: 複製指令のプレフィクスによる位置指定

た、実際の作図では隣接させるより間隔を持たせて並べることも多い。そこで数値プレフィクスの 10 の位も活用して、11d、13d などとした場合には図 3 のように現図形の大きさ分だけ空けて配置するものとした、さらに空けたい場合には 10 の位を 2、3、... としてもよい。これらの機能を入れたことにより、同じ図形を規則的に複数個配置するような場合の使い勝手は大幅に改良された。

2.5.4 現図形を基準とした新規図形生成方式

前述のように、新規に図形を作成する際にキー操作で位置と形を指定するのはかなり煩わしい。重複の際の位置指定が成功したので、図形の新規作成時にも同様の方式が適用できないかと考えた。基本的な考え方としては、現在選択中の図形群の bbox を考え、n 指令ではそれと同じ大きさの長方形に囲まれるような大きさで新しい図形を生成するが、加えてその位置も d 指令と同様に数値プレフィクスで指定する、というものである

また、直線や折れ線などの図形では図 4 に示すように、現在選択されている図形群から指定した方向 (上下左右および斜め) にとりあえずの位置と形が設定されるようにした。これは、これらの種類の図形は「ど

[3]図形番号が若い図形ほど後から作られたものであることに注意。

の図形からどの図形までを結ぶ」という形で使われることが多いと考えたからである。

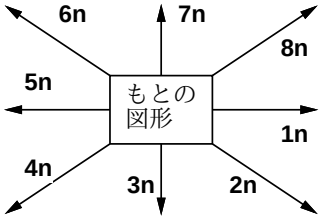

図 4: 直線のプレフィクスによる位置指定

これらの指定でできるのはあくまでも「とりあえずの位置と形」であり、その後そこから場所や形を変更するのが前提である。しかし「とりあえず」のままで済まない時も、本来目指すのに近い「とりあえずの位置と形」で現れてくれればその後の修正が心理的に非常に楽になるという効果が大きかった。

2.5.5 その他の改良

これまでの段階で k2d はすべての操作をキーボードのみで行なえるようになっていた「はずだった」が、実際にはあと 1 つだけ、どうしてもマウスを使ってしまう場合が残っていた... それは、「どのキーが何のショートカットだったか」を忘れた時にはマウスでプルダウンメニューを降ろしてみる必要がある、というものだった! (idraw ではメニューの一部としてショートカットが何であるかを表示するようになっている。) このままでは画竜点晴を描く、ではなく欠くので、最後の指令として

番号 `M` -- 左から n 番目のメニューを降ろす

という指令を追加した。なお、この指令はそのメニューを選択するような機能は一切なく、指定したメニューを表示したあと任意のキーが押されるまでそのまま待機し、キーが押されると終了する。

3 マウス操作とキーボード操作の比較実験

3.1 実験の方針と概要

前節までで説明したような改良が完成した段階で、描く図の種類によってはキーボード操作でもマウスに負けないだろう、という見通しが立ったので、マウス操作とキーボード操作を比較する実験を計画した。本来は多数の被験者を用いて厳密な比較を行なうのが望ましかったが、時間的/経済的制約もあったので、とりあえず筆者ら 3 人が被験者となり、[1][2] と同様の方法でタイミング計測を行ないながらマウス操作とキーボード操作で同じ図形を描いて比較する形で行なった。本節ではその概要を述べる。

実験には SparcStation SLC (8MB メモリ) 上で X-Window(Version 11、Release 4、サーバにタイミング記録機構組み込み済み) と k2d をともに動かす形で実施した。実験に使用した図は図 5〜図 9 で、比較的簡単だが互いに (特に図を描く操作として) あまり重複しないように選んだつもりである。実験は被験者が交替でマウス、キーボード双方を用いてそれぞれに好きなやり方で図形を作成し、十分習熟したと判断したところで 2〜3 回、時間計測しながら実験を行なった。ある被験者が練習/実験を行なっている間他の被験者はそれを傍らから観察しているので、結果的に各被験者の操作手順は (完全に同じではないが) 似通ったものとなった。しかし、全般に被験者間の所要時間の差は同一被験者でのマウス操作とキーボード操作の差よりも大きかった。そこで本稿では全般的な傾向を示すため、とりあえず被験者のうち 1 名のみのタイミングデータから、マウス操作、キーボード操作ともそれぞれ最速だったものを取り上げて比較検討する。

図 10 に 5 つの実験のタイミングチャートをまとめて示す。各チャートは上から下に向かって時間が進む方向を示し、縦棒の左側に秒単位の時間目盛りを記した。また右側にはキーの押し下げ (1 本線) およびマウスボタンの押しと離し (2 本の線をコの字形に結んである) を示した。秒数はキーボード操作の実験の左側のみに記入し、マウス操作の方は左側には押されているマウスボタンが左 (L)、中 (M)、右 (R) のどれかを記した。どちらも右側にはキーボードのキーを記入した。コントロールキーは分けて示さず、例えば `^[` であればその両者を上下に重ねて記した。また、横向きの三角は `[RET]`、黒丸は `[DEL]` を示す (本当はエラーフリーな例だけ掲載したかったが、実験数が十分でなく、誤りを含むものを掲載せざるを得なかった)。

以下では各実験ごとにその図の概要、マウスとキーボードそれぞれによる操作の概要、およびタイミングチャートに基づく分析をこの順で記す。

3.2 実験 1

概要: 箱を 5 つ描き、その間を矢線で結ぶ。箱の中にはそれぞれタイトルが入る。

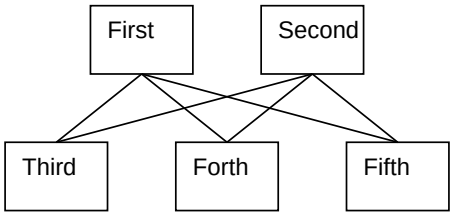

図 5: 実験 1 に用いた図

マウス操作: マウスで長方形のパレットを選択してから最初の箱を 1 つ作る。それを重複し、適切な位置へ移動することを 4 回繰り返して 5 個の箱を配置する。ショートカットで文字入力パレットを選択し、それぞれの箱の中をクリックしてタイトルを打ち込むことを 5 回繰り返す。マウスで直線パレットを選択し、計 6 本の直線を描く。

キーボード操作: `bn` で最初の箱を作る。場所は最初に出来た位置でよいことにしてすぐ `[RET]` で確定する。大きさは右に 2 グリッドのばして確定。`tn` でテキスト作成モードに入りタイトルを入力し、`^[`を押して終る。`^P` で先の箱を選択し、`d` で重複し、できた箱を 10 グリッド右へ移動する。`tn` で先と同様にタイトルをつけ、`^P` で箱を選択し、`d` で重複し、8 下へ移動し、5 右へ移動する。今度はすぐにそれを `d` で重複し、10 左へ移し、またすぐ `d` で重複し、10 左へ移す。これで箱は全部できたので、今度は下左の箱から順にタイトルをつけてゆく。右下の箱まで来た状態で`^K` で右上の箱を選び、`i3n` でその箱から下へ向かう線を作りはじめる。その始点は最初のままでいいのですぐ確定し、終点は `l` を繰り返して右下の箱の上辺中点まで動かし確定する。`d` でその線を重複し、`1q` で終点の位置を変更し始め、`10h` で中下の箱の位置へ移動して確定。同様にこれをすぐ重複して変形し終点を左下の箱の位置へ動かす。`^K` で左上の箱を選択し、先と同様にして 3 本の直線を描く。

分析:

- 総所用時間ではキーボードが 42 秒、マウスが 44 秒でキーボードがやや速い。さらに細かく見ると、次のような点がわかる。
- 最後に直線 6 本を引く部分がキーボード 16 秒、マウス 12.5 秒でマウスの方が優っている。これは主観的にもキーボードでは「この位置」というのを指定するのが直接的でなくいらいらする、という感覚があるのでうなずける。それでもこの程度の差で済むのは図に規則性があって移動量などを直接数値で打ち込むことができるためである。
- タイトルのテキストを打ち込む時間はどちらでもほぼ同じである。これは、マウスとキーボードの間で手が移動する方が不利だと予想していた筆者らには予想外であった。その原因は、キーボードのみの操作では終わりに`^[`を打ち込み、次の箱へ移動したりする打鍵そのものに時間が掛かるのに対し、マウスを併用する場合には単に次の位置でマウスをクリックすることで前のテキストの終了と次のテキストの開始を同時に行える有利さがあるためかと思われる。キーボードの場合でもテキスト入力中に図形選択指令 (`^P` など) を打つとただちにテキスト入力が終るようにすればさらに時間を短縮できると思われる。
- 時間的には同じでもマウスの場合には「図形はマウス、テキストはキーボード」という心理的な区別と物理的な区別が一致する快適さがあるのに対し、k2d ではキーボードが図形操作とテキスト入力で重複して使われるという点に習熟しなければならない抵抗感が (少なくとも最初のうちは) ある。これに対して、指令をすべてコントロールキー等の組合せで表わし、通常のキーをすべてテキスト入力用に割り当てる、というアイデアもあったが、キーバインディングの数が足らなくなること、図形の操作が通常キーでできる方が (お絵描きツールである以上) 重要であると考えたことからそうはしなかった。テキストエディタでいえば k2d は vi 流であり、emacs 流ではない、ということである。
- テキスト入力が互角で線引きが負けているのに全体で (わずかだが) 速いのはそれ以外の部分、つまり最初の箱を作る時と規則的に配置する時が速いためである。マウスでは箱の大きさや位置を正確に指定するのはかなり慎重に操作しなければならず、それに時間が掛かる。これに対し、キーボードでは 1 ストロークが 1 グリッド、という具合いに対応が明確だから「何回押せば済む」(回数が多くなれば数値で打ち込む) になり、慣れて来ると高速に行える。

3.3 実験 2

概要: 4 つの箱が等間隔で並び、それらの間が直線で結ばれている。各箱にはタイトルが入っている。

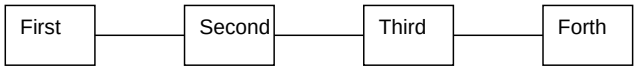

図 6: 実験 2 に用いた図

マウス操作: パレットから長方形を選択し、最初の箱を作る。これを重複し、次の箱の位置に移動することを 3 回繰り返して 4 個の箱を作る。直線のパレットを選択し、箱の間の 3 本の直線を引く。次にショートカットでテキストのパレットにし、各箱の中をクリックしてタイトルを打ち込むことを 4 回繰り返す。

キーボード操作: bn で長方形を作り、位置は最初のままでよいのですぐ確定。次に ll と打ち横に長い形にして確定。次に 11d を 3 回行い、3 個の箱をそれぞれ箱の横幅ぶんあけて配置する。次に^H を 3 回打ち最初の箱を選択した状態で i1n でこの箱から右へ出る直線を作り始めるが、その始点も終点も最初にできたままで良いので 2 回 [RET] を打って確定する。また 11d を 2 回行い残り 2 本の直線を配置する。また^H を 5 回打ち再び最初の箱を選択。あとは実験 1 と同様にして各箱にタイトルを打ち込む。

分析:

- 総所用時間は、マウスで 33 秒、キーボードで 21 秒で、圧倒的にキーボードが速い。
- 再び、テキストを打ち込んでいる時間はともに 11 秒で同じなので、残りの図形を配置している時間で倍、k2d が速いことになる。(テキストを含めたのはその方が k2d に有利だからだと考えたのだが、裏目であった。)
- やはり、規則的に図形を配置する場合にはキーボードによる指令の方が適しているという点につきる。
- 図形を選択するのに^H などのキーを繰り返し打つのは不利ではないかと予想していたが、実際にはそれに要する時間はほとんど問題にならない位小さかった。

3.4 実験 3

概要: 折れ線ツールを使ってジグザグ形を描く。これは実験 2 のような k2d の指令が特に有利な場合と対比させ、普通に位置決めをしながら図を描く場合を想定して行ったものである。

図 7: 実験 3 に用いた図

マウス操作: 折れ線のパレットを選択し、各点をクリックしながら描く。

キーボード操作: 折れ線のパレットを選択し、最初の位置はそのままでいいので確定し、2 番目の点は hh で移動して確定する。3 番目の点からは、4kll[RET]、4jll[RET] の反復。最後は^[。

分析:

- キーボード 8 秒、マウス 9 秒でややキーボードが速いが、ほとんど差はない。
- 1 つの点から次の点まで移るのにマウスで 1.5 秒くらい掛かるのに対し、キーボードで 4kll[RET] の 5 打鍵は 1 秒程度なので優っている。
- その代わり、ある点から次の点に移るときやや「考えている」間があるため、全体としては同じくらいになる。結局、キーボードの場合は多用な指令が駆使できる反面、それに習熟するまでが大変で、習熟しても「次にどの指令を使うか」プランする時間が掛かる。

3.5 実験 4

概要: 正方形を 3×3 の形に並べ、その中に 0〜9 の数字を入れる。

マウス操作: 長方形のパレットを選択し、最初の正方形を作る。それを重複して移動することを 2 回繰り返し、一番上の列ができる。ショートカットで「全図形の選択」を行い、これを重複して下へ移動することを 2 回繰り返して箱が 9 個できる。あとは実験 1、2 と同様にショートカットでテキストのパレットを選んで入力。ただし、数字 1 文字ずつなので右手にマウス、左手にキーボードの状態で操作した。

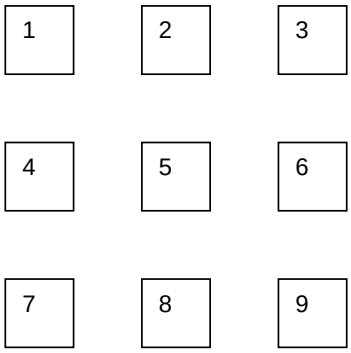

図 8: 実験 4 に用いた図

キーボード操作:　bn で正方形ができるので、位置も形もそのままで確定し、11d を 2 回行って最上列を完成する。次に A で全部を選択し、今度は 13d を 2 回行って箱が 9 個できる。あとは^K 等で箱を選択しては tn で数字を打ち込む。

分析:

- キーボードが 27 秒、マウスが 28 秒でほとんど差はない。今度はテキスト入力がキーボード 20 秒、マウス 18 秒でややマウスが速いが、これは上述のように右手でマウスを持ったまま数字を入力したことによると思われる。箱の配置はやはりキーボードが速い。

3.6　実験 5

概要:　4 状態のオートマトンふうの絵。円が 4 個等間隔で並び、それらの間が 2 本のスプライン曲線で環状に結ばれている。

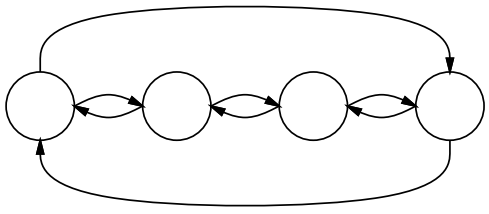

図 9: 実験 5 に用いた図

マウス操作:　円のパレットを選択し、円を作る。重複、移動を 3 回繰り返し 4 個の円を配置する。スプラインのパレットを選択し、最左の円から最右の円までを円の列の上を通って途中 2 個の制御点を作りながら結ぶ。これを重複し、90 度回転のショートカットを 2 回使ってひっくり返し、移動して円の列の下側に配置する。次に最左の円から隣の円への曲線を途中制御点 1 個で作り、また重複、回転して下側の曲線の位置に置く。シフトキーを押しながら上側の曲線を追加選択し、これの重複・移動を 2 回繰り返して 2、3 番目の円の間と 3、4 番目の円の間に配置する。

キーボード操作:　円のパレットを選択し、最初にできた位置と形で確定する。11d を 3 回行い 4 個の円を配置する。スプラインのパレットを選択し、3n で最右の円から下向きに描き始める。最初の制御点はできたままの位置でいいので [RET]2 回で位置とともに確定し、次の制御点は 24h でいきなり最左の円の下に持ってきて確定。あとは k を 4 回打って最左の円に到達し、^[でスプラインを書き終る。(この後 d までが間違いで、4 秒ほど無駄な動作をしている。) 次に d でこの曲線を重複し、2 回回転して k を 8 回打って上側の弧の位置に置く。^J^H で最左の円を選び、1n で右に向かってスプラインを書き始める。最初の位置はそのまま確定し、khh で最初の制御点を指定して確定、続いて jll で終点を指定して^[で終る。再びこれを重複して回転し移動し下側の曲線の位置に置く。2a で先の曲線を追加選択し、d でこれら 2 つを同時に重複して 18 個で次の円の位置にもって来ることを 2 回繰り返す。

分析:

- キーボードが 34 秒、マウスが 36 秒だがキーボード操作では途中に 4 秒ほど誤りのため無駄な部分があるので、差し引くとキーボードの方が速い。
- 円を規則的に配置する部分ではこれまでの実験と同様に k2d が優れている。
- それ以外の部分でも全体にキーボードによる操作で遜色はない。
- k や l などを連続して移動している部分は回数を打ち込めばさらに速くなるが、心理的には k や l を繰り返して目的の位置に「だんだん近づく」方が安心感があるのでこうなっている。規則性があり、繰り返しが多くなってくれば回数指定に移行することが多いと思われる。

3.7　実験結果のまとめ

実験全体を通じて、図形を規則的に配置する場合には明らかに k2d のやり方がマウスで 1 個ずつ配置して

行くより優っている。従って本方式の有効性は描く図形にそのような規則性があるかどうかに大きく影響される。

これらの実験全体を通じて、マウス操作は同じ図形を描くことによる習熟が比較的小さかったのに対し、キー操作はかなり急激に所要時間が短くなる傾向が見られた。特に、これまでとは異なる指令列を発見し適用することによって大幅に時間が短縮される場合がそうであった。

しかし一方で、新しい図形を描き始めるときに、「どのように描こうか」と考えるので一瞬 (というよりかなり長時間) 手が止まることもあり、マウス操作に比べると習熟するための努力はかなり多く必要とするように思われる。(そのような努力をあまりしない場合には結局 `k` や `l` などの単純な指令を多く使うのであまり高速にならない。)

なお、ここに取り上げた実験は k2d の開発からそれほど時間を置かずに行なったものなので、実験時の被験者が十分習熟していなかったという可能性も否めない。実際、筆者らは実験時にはキーのみで操作することに対する違和感をかなり感じていたが、その後 k2d を日常的に使うように努力した結果、本稿執筆時点ではかなり「なじんだ」感じを持つようになっている。従って、今後さらに習熟した被験者を対象として追試を行なうことが必要であろう。

4 おわりに

本稿に報告した実験はあくまでも予備的なものであり、これだけから強い結論を述べることは適当でないが、少なくとも、被験者の適性、および図の性質によっては、マウスで描くよりもキーボードで描く方が高速に行なえるような場合が存在することは示せたと考える。

また、キーボードによる作画は慣れるまでは違和感が大きいが、連続して使って習熟してくると主観的にも「マウスで描くよりやりやすい」と感じるようになってくる、というのが筆者らの経験である。実際、本稿の図は (タイミングチャートを除いて) 全てキーボードのみで作画したものである。

さらに、キーボード操作の特性を活かす方向づけとして次のような可能性が存在すると思われる。

- キーボード操作の直列性 (指令が順にならんでいる) を活かして、エディタのキーボードマクロや定義コマンドのような機能を組み込み、カスタマイズ機能を通じてより効率のよい作画を目指す。
- マウスが壊れている場合や身体的障害、あるいは振動の激しい乗りもの上での作画など、マウスを使用するのが適さない場合の作画を可能にする。
- 電話など、図形を直接表示できないメディアを通じてともかくどんな絵であるかを指定する可能性を開く。[4]

また、本稿で述べた k2d のやり方はキーボード操作による描画方式のあくまでも一例であり、他により優れたキーボード操作方式がある可能性も十分大きいと筆者らは考えている。

参考文献

[1] 久野　靖, 角田博保, キーボードによる窓操作機構の作成と評価, 情報処理学会論文誌, vol. 31, no. 5, pp. 721-730, 1990.

[2] 久野　靖, 角田博保, 窓はねずみなしでも操れるか?, 第 30 回プログラミングシンポジウム報告集, pp. 93-104, 1989.

[3] 千葉　滋, 漢字対応 idraw (kdraw) patch kit (level 1-3), fj.sources.unix, 1990.

[4] J. M. Vlissides, M. A. Linton, Applying Object-Oriented Design to Structured Graphics, Proc. 1988 USENIX C++ Conference, pp. 81-94, 1988.

[5] M. A. Linton, J. M. Vlissides, P. R. Calder, Composing User Interface with InterViews, IEEE Computer, vol. 22, no. 2, pp.8-22, 1989.

[6] Y. Kuno, K2d Reference and Tutorial, Technical Report 91-07, Graduate School of Systems Management, University of Tsukuba, Tokyo, 1991.

[4] 実は図 5〜9 は実験の時の図を保存しておくのを忘れたので、図 10 のタイミングチャートを見ながら打ち込むことにより再現した。マウスで同様のことが行なえるとはとても思えない。