インターフェイスの街角 (5)
予測型テキスト入力システム POBox

増井俊之

前回まで、曖昧検索システムと予測インターフェイス・システムについて紹介してきました。今回は、曖昧検索と入力単語の予測によりペン計算機での文章入力を効率化するシステム「POBox」を紹介します。

キーボード中心主義の限界

計算機上で文章やメールを書いたり編集したりするとき、ほとんどの人がキーボードを使っていると思います。優れたエディタや漢字変換システムのおかげで、キーボードによる文書入力/編集はたいへん楽になりましたし、慣れれば紙に書くよりもはるかに速く文書を作成できます。

しかし、これはタイピングに慣れた人が机の前に坐ってキーボードを打つ場合にかぎった話であり、キーボードが使えない状況や、キーボードに不慣れな人の場合は、なかなか高速に入力できないのが現状です。

モバイル・コンピューティングが普及してきたとはいうものの、計算機はいまのところ机の上で使われることが多いようですし、そもそも計算機の利用者にはキーボードを打つのが苦にならない人が多いため、キーボード中心の考え方が疑問に思われていないようです。しかし、ユビキタス・コンピューティング (ubiquitous computing)[1]やウェアラブル・コンピューティング (wearable computing)[2]などに代表されるような、実世界における計算機の将来的な使われ方を考えると、キーボードが利用できない状況でも文字や文章を効率的に入力するための手法が大きな課題になってくると考えられます。電車で立っているときにメールの返事を書いたり、電子黒板に字を書いたり、風呂のなかで Web のアンケートに答えたりするためには、キーボード以外の入力手法が必要です。

キーボード以外の文書入力手法では、音声認識がよく話題になります。しかし、実用化にはまだまだ時間がかかりそうですし、電車のなかなどで音声認識でメールを書くのが一般的になるとはあまり思えませんから[3]、今後は小さくて片手でも扱える装置が普及する可能性が高いのではないでしょうか。片手キーボードやダイヤル、ペン型入力装置などが候補になるでしょう。

ポインティング・デバイスによる文字入力

現在、キーボードが利用できない状況で使われるもっとも一般的な入力装置はマウスやペンのようなポインティング・デバイスでしょう。事実、PDA (Personal Digital Assistants) の多くはペンのみで操作するようになっていますし、街角の情報端末や自動販売機などではタッチパネルがよく使われています。机の上以外の場所で計算機を使うためには、ペンなどのポインティング・デバイスでも文字や文章を簡単に入力できなければなりません。

ポインティング・デバイスで文字や単語を入力する場合は、手書き文字認識かソフトキーボード[4]のいずれかの方法が使われることが多いようです。しかし、手書き文字認

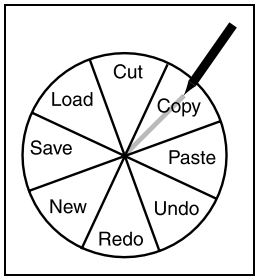
1 あちこちに計算機を配置したり持ち歩いたりすることにより、生活のあらゆる場面で計算機を活用しようという考え方 [4]。

2 計算機を服のように身につけて四六時中活用しようという考え方。最近、MIT (http://lcs.www.media.mit.edu/projects/wearables/) やジョージア工科大学 (http://mime1.marc.gatech.edu/wearable_computing/) などでさかんに研究が進められており、1997 年末には初めて学会も開催されました (http://computer.org/conferen/proceed/8192abs.htm)。

3 「周りのお客様のご迷惑になりますので車内での音声認識はご遠慮ください」とアナウンスされることになるでしょう。

4 画面上にキーボードを表示し、実物と同じように操作できるようにしたもの。

図 1　パイメニュー



識の場合は、すべてのストロークをある程度正確な筆順で入力する必要があるので手間がかかりますし、大量の文字を書くと手が疲れてしまいます。ソフトキーボードにしても、表示されたキーボード画面を正確に押さねばならないので、目や神経が疲れてしまいます。このため、ペンやマウスの簡単な操作で文字を入力する手法として、T-Cube や Unistroke といったシステムが考案されています。

T-Cube

T-Cube[3] は、文字入力に〝パイメニュー〟を使用する方式です。パイメニューとは、ポップアップ・メニューの各項目を矩形ではなく円内の楔形領域で表現するものです。たとえば、マウスボタンを押すと、押した位置を中心とした円形のメニューが表示され、マウスを各方向にドラッグすることによりそれぞれの項目が選択されます（図 1）。普通のメニューとは異なり、マウスの移動方向によって選択項目が決まるため、マウスボタンを離す位置を目で正確に確認しなくてもよいという特徴があります[5]。

1 つのパイメニューにはせいぜい 10 種類ぐらいの項目しか置けませんが、T-Cube では、マウスボタンを押した位置に応じて異なるパイメニューを表示させ、数十文字を選択できるようにしています（図 2）。

Unistroke

Unistroke[1] は、普通の文字の代わりに図 3 のような一筆書き記号を使って文字を入力する手法です。

a、e、i などのよく使う文字には簡単なストロークが割り当てられているため、字形とストロークはあまり似ていません。したがって、利用する際は事前にストロークを憶

5　メニューを表示せずにペンの軌跡のみを表示して同様の効果を得る〝マーキング・メニュー〟や、ボタンを押したあと、上にドラッグしてから右にドラッグするというふうに、マウスを 2 段階に動かして選択可能な項目数を増やす〝階層型マーキング・メニュー〟も提案されています [2]。

図 2　T-Cube による文字入力

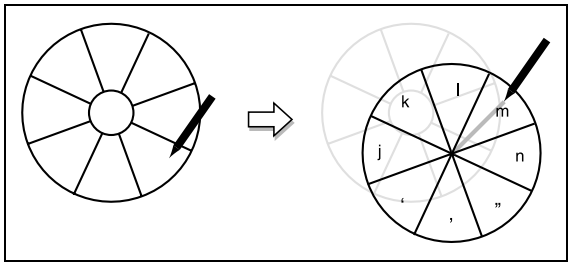


図 3　Unistroke

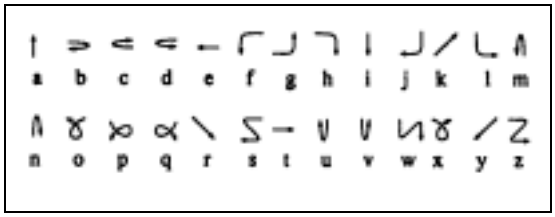
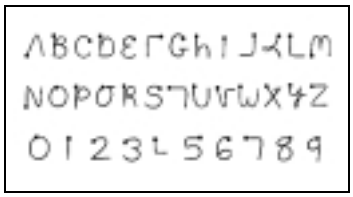


図 4　Graffiti



えておく必要があります。しかし、慣れればかなり高速に文字を書けるようになります[6]。

Graffiti

Unistroke はかなり高速な入力手法ですが、すべてのストロークを憶えなければ使えないので、ややとっつきにくいところがあります。1997 年 12 月号でも紹介しましたが、携帯端末 PalmPilot などでは、Unistroke と考え方の似た Graffiti という一筆書き方式が採用されています。

Graffiti のストロークは Unistroke のものより複雑ですが、普通のアルファベットの字形に近いため、憶えやすいという利点があります。

◇◇◇

T-Cube や Unistroke などでは、1 文字ずつの入力方法をより簡便にして入力の効率化を図っています。しかし、1 文字ずつ入力しなければならないことに変わりはありませんし、操作や認識の誤りなどによって間違った文字

6　聞くところによれば、Unistroke の開発者の David Goldberg 氏は 1 分間に 40 ワードも入力できるそうです。

図 5　初期画面

が入力された場合は、再度入力しなければならないので、どうしても時間や手間がかかってしまいます。日本語では、これに漢字変換の手間が加わるのでさらに大変です。

ポインティング・デバイスによる文字入力は、キーボードとくらべるとどうしても遅く不正確になりがちです。そこで、すべてを正確に入力しようとするのはやめて、

- **すべての文字を入力しなくてもよい工夫**
- **正確に文字を入力しなくてもよい工夫**

を施せば、入力効率を改善できる可能性があります。

2 月号で紹介した〝予測インターフェイス〟を使って次の入力文字や単語を予測すれば前者を実現できますし、1 月号で紹介した〝動的曖昧検索〟を応用すれば、多少誤った読みや綴りからでも正しい単語を検索して入力単語とすることによって後者を実現できますから、これらを融合すれば入力効率が向上するのではないでしょうか。

日本語の場合には、

- **漢字変換の手間を省く工夫**

も重要です。

POBox (Pen-Operation Based On eXample) は、これらの工夫をとりいれたペンによる高速文書入力を支援するシステムです [5]。

POBox による文章入力例

まず、POBox で高速に文書を入力する例を示します。

次単語の予測

「日本語を高速入力」という文を入力する例を以下に示します。図 5 が POBox の初期画面です。

図 6　〝に〟を指定した状態

図 7　ドラッグにより〝ほ〟を指定した状態

図 8　ドラッグにより〝日本語〟を選択した状態

ペンやマウスで 50 音ソフトキーボードの〝に〟を指定すると、〝に〟で始まる単語のうち、出現頻度の高いものがプルダウン・メニューとして表示されます（図 6）。

ここでは、〝日本語〟という単語はメニューに表示されていません。しかし、ペンやマウスをそのまま〝ほ〟のところまでドラッグすると、今度は〝にほ〟で始まる単語がリストされ（図 7）、〝日本語〟を選択することができます（図 8）。ペンを押してから離すまでを 1 操作と考えると、1 操作で〝日本語〟が入力できたことになります。

図 8 ではプルダウン・メニューで単語を選択しましたが、図 7 の状態でペンを画面から離すと、候補がソフトキーボードの下に表示されてボタンのように選択可能になります（図 9）。

図 8〜9 のようにして〝日本語〟を選択すると、〝入力〟〝の〟〝を〟〝版〟といった、〝日本語〟の後ろに続く可能

図 9　ペンを離した状態

図 10　〝日本語〟を選択した直後

図 11　〝を〟を選択

図 12　〝こう〟を指定

図 13　〝に〟を指定して〝入力〟を選択

図 14　〝にふんこ〟を指定した場合

性が高い単語が候補として表示されるので（図 10）、次の文字〝を〟を選択します（図 11）。

同様に、図 12〜13 のようにして〝高速〟〝入力〟と入力していくことができます。

このように、〝日本語を高速入力〟という文字列をわずか 4 操作で入力できます。

いったん選択した単語は、次回はメニューの先頭に移動するので、似たような文章を入力する場合はメニューが適応していきます。

単語の曖昧検索

画面が小さい場合はソフトキーボードを正確に操作するのが難しいので間違ったキーを押しがちです。これに対し、POBox では指定された単語がみつからない場合、1 月号で紹介した動的曖昧検索をおこなって候補を表示するため、図 14 のように、〝日本語〟と入力しようとして間違えて〝にふんこ〟などとしても、予測された候補のなかから正しい文字列を選択できます。

ペン計算機と UNIX

POBox の内部について解説する前に、UNIX をペン計算機で使う方法について説明しましょう。

ペン計算機で UNIX を使う方法

最近は、ペン入力に対応した各種の PC が発売されており、FreeBSD や Linux なども比較的簡単に動かせるようになりました。しかし、これらの UNIX を普通にインストールしただけでは、ブート後にコンソールのログイン画面が出るようになっているため、キーボードがなければログインすらできません。そこで、

図 15　XTEST のおもな関数

XTestFakeKeyEvent(Display *display, unsigned int keycode, Bool ispress, unsigned long delay) キー操作をシミュレート
XTestFakeButtonEvent(Display *display, unsigned int button, Bool ispress, unsigned long delay) マウスボタン操作をシミュレート
XTestFakeMotionEvent(Display *display, int screennumber, int x, int y, unsigned long delay) マウスの絶対位置移動をシミュレート
XTestFakeRelativeMotionEvent(Display *display, int screennumber, int x, int y, unsigned long delay) マウスの相対位置移動をシミュレート
XTestGrabControl(Display *display, Bool impervious) Server Grab の扱いを変える

1. コンソール・ドライバをペン対応にする
2. X サーバーをペン対応にする
3. X のクライアントとしてペン入力プログラムをつねに動かしておく

といった工夫が必要になります。

しかも、ペン型 PC におけるペンのハードウェアは機種によりまちまちなので、機種ごとに対応しなければなりません。最近発売された筑波大学の大木敦雄さんによる『FreeBSD カーネル入門』[6] では、FreeBSD のカーネルを改造して三菱電機のペン型計算機 Amity 上でペン入力をおこなう方法が紹介されています。この本では、シリアルドライバとコンソール・ドライバを改造し、T-Cube を使用可能にする手法（上記の 1）と、X サーバーをペン対応に改造し、入力拡張機能で Unistroke 認識プログラムを使う手法（上記の 2）が詳しく解説されています。

1 の手法ではコンソール画面でもペンが使えますが、カーネルを再構築しなければならないのがやや面倒です。2 の手法は、X が動いていればマウスと同様にペンを使えるメリットはありますが、Accelerated-X のような商用の X サーバーは利用できません。これに対し、3 はペン入力プログラムに不具合が生じると手も足も出なくなってしまいますが、カーネルや X サーバーの改造が不要なので、もっとも手軽な手法といえるでしょう。

XFree86 のデフォルト設定や Accelerated-X には、X11 の拡張モジュール XTEST が含まれています。これを使うと、クライアント・プログラムからマウスやキーボードの動きを制御することができます。

XTEST

XTEST は、もともとサーバーの機能テストを簡便におこなうための拡張モジュールですが、マウスやキーボードの操作をクライアント側からシミュレートすることができます。たとえば、クライアントから XTestFakeButtonEvent() を呼び出すことによって、ユーザーのマウスボタン操作と同様の効果が得られます[7]。

XTEST を使えば、X11 上でのマウスとキーボードの操作をシミュレートできます。ペン入力をこれらの操作に対応させれば、X11 をペンで扱えるようになります。XTEST のおもな関数を図 15 に示します。

Pen daemon

ここでは、大木さんの本と同じ環境（Amity + FreeBSD）について説明しますが、ペン入力の処理以外はほかの環境でも同様です。

まず、ペン入力をマウス入力に変換するデーモン **pend** を作成します。Amity SP の場合、ペン情報は 19,200bps、7 バイト単位でシリアルポートに送られます。データ形式は、表 1 のようになっています[8]。

pend は/dev/ttyd2[9]から 7 バイトずつペン情報を読み取り、位置情報とボタン情報を解釈して、XTEST ライブラリを用いて等価なマウス操作に変換します。

ペンのボタンが押された場合は、XTestFakeButtonEvent() を使ってマウスボタン操作をシミュレートし、ペンの位置が移動した場合は XTestFakeMotionEvent()

7 XTEST の関連資料は、ftp://ftp.XFree86.org/pub/XFree86/current/、ftp://ring.aist.go.jp/pub/unix/XFree86/current/ (Ring Server)、CD-ROM などの X11 配布メディアの untarred/xc/doc/specs/Xext/xtestlib.ms、untarred/xc/doc/hardcopy/Xext/xtestlib.PS.Z で入手できます。

8 これはメーカー資料にもとづいたものではなく、データの観察結果から推測したものです。当然、誤りを含んでいる可能性があります。

9 FreeBSD では、各デバイスの I/O ポートと IRQ をブート時のコマンドで設定できます。Amity SP の場合は、I/O アドレス 0x208 のシリアルポートに IRQ 11 を割り当てることにより、/dev/ttyd2 などからペンの入力を取得できるようになります。

表 1　Amity のペン信号

データ		意味
byte[0]	bit7(MSB)	1
	bit6	
	bit5	ペンがタブレットに近いかどうか
	bit4	
	bit3	
	bit2	
	bit1	ボタン 2 (ペン横のボタン)
	bit0(LSB)	ボタン 1 (タブレット接触)
byte[1]		ペン X 座標 上位
byte[2]		ペン X 座標 下位 7 ビット
byte[3]		ペン Y 座標 上位
byte[4]		ペン Y 座標 下位 7 ビット
byte[5]		ペン圧力
byte[6]		0x01 または 0x11

を使ってマウス移動をシミュレートします。pend のソースファイルを末尾に示します。

キーボード入力

X11 の上で pend を動かしておけば、ペンをマウスの代わりに利用できます。したがって、あとはペンをキーボード代わりに使える仕組みを用意すれば、ペンで X11 を完全に操作できるようになります。

さきほどの POBox の使用例では、日本語を入力する様子を示しましたが、ウィンドウの右上の [直接] ボタンをクリックすると、POBox の画面は図 16 のように変わり、単純なソフトキーボードとして使えるようになります。たとえばシェルを使う場合は、ウィンドウ・マネージャーを使ってまずキーボード・フォーカスを kterm などに設定し、POBox のソフトキーが押されたときに XTestFakeKeyEvent() を呼び出してキーボード入力をシミュレートできます。

おわりに

今回は、予測型入力システム POBox の概要とペン計算機で UNIX を使う方法を紹介しました。次回は、POBox の予測方式などについてもうすこし詳しく説明します。

3 月号で紹介した Reactive Keyboard や POBox に似た各種のシステムは、障害者向けインターフェイスとして研究されたり紹介されたりしていることが多いようです。しかし、障害の程度や入出力機器の習熟度は十人十色

図 16　単純なソフトキーボードとしての使用

ですし、電車のなかではキーボードが使いづらかったり、運転中はディスプレイが見づらかったりするように、状況によって機器が使えたり使えなかったりします。障害者向けといった扱いはせずに、つねにいろいろな状況に対応できるインターフェイス手法を用意しておくことが重要なのではないでしょうか。

今回紹介したプログラムとペンで UNIX を使うときの情報、POBox 全般の情報は、私の Web ページ（http://www.csl.sony.co.jp/person/masui/UnixMagazine/）から入手できます。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] David Goldberg and Cate Richardson, Touch-typing with a stylus, In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems (CHI'93)*, pp. 80–87, Addison-Wesley, April 1993

[2] Gordon Kurtenbach and William Buxton, The limits of expert performance using hierarchic marking menus, In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems (CHI'93)*, pp. 482–487, Addison-Wesley, April 1993

[3] Dan Venolia and Forrest Neiberg, T-Cube: A fast, self-disclosing pen-based alphabet, In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pp. 265–270, Addison-Wesley, April 1994

[4] Mark Weiser, Some computer science issues in ubiquitous computing, *Communications of the ACM*, Vol. 36, No. 7, pp. 75–84, July 1993

[5] 増井俊之「ペンを用いた高速文章入力手法」、田中二郎（編）『インタラクティブシステムとソフトウェア IV：日本ソフトウェア科学会 WISS'96』、pp. 51–60、近代科学社、1996 年 12 月

[6] 大木敦雄『FreeBSD カーネル入門』、アスキー、1998 年 1 月

• pend のソースファイル

```
#include <stdio.h>
#include <fcntl.h>
#include <termios.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>

typedef struct {
    int state;
    int x,y;
    int pressure;
} Pen;

int penopen(char *ttydev) {
    struct termios tty;
    int fd;
    int flags;

    fd = open(ttydev, O_RDWR | O_NONBLOCK);
    if(fd < 0){
        fprintf(stderr,"Can't open %s\n",ttydev);
        return -1;
    }
    flags = fcntl(fd,F_GETFL,NULL);
    fcntl(fd,F_SETFL,flags & (~O_NONBLOCK));

    tcflush(fd, TCIFLUSH);
    tcgetattr(fd, &tty);

    cfsetispeed(&tty, B19200);
    cfsetospeed(&tty, B19200);

    tcsetattr(fd, TCSADRAIN, &tty);
    return fd;
}

int penread(int fd, Pen *pen){
    unsigned char b[7];
    int n;
    // 正しいデータまで読み飛ばす
    while(1){
        n = read(fd,b,1);
        if(n==1 && ((b[0] & 0xfc)==0x80 ||
                    (b[0] & 0xfc)==0xa0)){
            while(1){
                n += read(fd,b+n,7-n);
                if(n == 7) break;
            }
            if(b[6] == 0x01 || b[6] == 0x11) break;
        }
    }
    pen->state = b[0];
    pen->x = (b[1]<<7)|b[2];
    pen->y = (b[3]<<7)|b[4];
    pen->pressure = b[5];
}

main()
```

```
{
    Pen pen;
    int penfd;
    int bstate1, bstate2;
    int newbstate1, newbstate2;
    int posx, posy;
    int newposx, newposy;
    Display *xdisplay;
    char *display_name = NULL;

    penfd = penopen("/dev/ttyd2"); /* for FreeBSD */
    if(penfd < 0){
        fprintf(stderr,"Can't open pen device /dev/ttyd2\n");
        exit(0);
    }
    // サーバーと接続
    if((xdisplay=XOpenDisplay(display_name)) == NULL){
        fprintf(stderr,"Can't connect to X server...");
        if(getenv("DISPLAY") == NULL)
            fprintf(stderr,", 'DISPLAY' environment variable not set.\n");
        else
            fprintf(stderr," %s\n", XDisplayName(display_name));
        exit (-1);
    }
    // Server Grabを無効にする (for twm/fvwm)
    XTestGrabControl(xdisplay,True);

    while (1) {
        // ペンデータを読んで、XTestでサーバーに依頼
        penread(penfd,&pen);
        // タブレットと画面の解像度を適合させる
        newposx = pen.x * 640 / 0xc00;
        newposy = pen.y * 480 / 0x900;

        if(newposx != posx || newposy != posy){
            XTestFakeMotionEvent(xdisplay,-1,
                newposx, newposy,
                CurrentTime);
            posx = newposx;
            posy = newposy;
        }
        newbstate1 = pen.state & 0x01;
        if(newbstate1 != bstate1){
            XTestFakeButtonEvent(xdisplay,Button1,
                newbstate1 ? True : False,
                CurrentTime);
            bstate1 = newbstate1;
        }
        newbstate2 = pen.state & 0x02;
        if(newbstate2 != bstate2){
            XTestFakeButtonEvent(xdisplay,Button2,
                newbstate2 ? True : False,
                CurrentTime);
            bstate2 = newbstate2;
        }
        XFlush(xdisplay);
    }
}
```