インターフェイスの街角 (45) – 効率的かな入力

増井 俊之

日本語の入力方法

現在、日本語の〝読み〟をキーボードから入力するために使われている方式は、

- ローマ字入力
- JIS かな配列
- NICOLA（親指シフト）配列

の 3 つに絞ることができるでしょう。入力効率を考えると、JIS かな配列や NICOLA 配列のように 1 キーで 1 文字を入力できる〝かな入力〟のほうが優れています。しかし、私にはこれらのキー配列を憶える根性がなく、ずっとローマ字入力を使用しています。

ヘボン式や訓令式のローマ字は、英語などを母国語とする人にとって発音しやすい記法になっていますが、かならずしも日本語の入力に適しているとはいえません。日本語でよく出現する促音（っ）や拗音（ゃ、ゅ、ょ、ゎ）は標準的なローマ字では直接表現できないので、たとえば「うっひゃぁー」であれば、

```
u h h y a x a -
```

などと入力する必要があります。これは、あまり直感的とはいえないでしょう。これは極端な例かもしれませんが、ローマ字に馴染みのない年輩の方などは、促音や拗音の入力に苦労するという話をよく耳にします。

ローマ字は、キーストローク数の面でも効率がよくありません。たとえば、「中小企業」と書きたければ、

```
c h u u s h o u k i g y o u
  または
t y u u s y o u k i g y o u
```

のように 14 キーも入力しなければならず、タイピングに不慣れな人の場合は相当な時間がかかってしまうでしょう。

かな入力にすれば状況は多少改善されるはずですが、やはり配列を憶えるのが大変そうです。それに、拗音や促音を含む文字列は、頻出するにもかかわらず入力しにくいという問題はあまり変わりません。

ローマ字入力の場合でもかな入力の場合でも、ソフトキーボードやテンキーなどを使用する携帯機器では、拗促音や濁音の指定によぶんな手間がかかることが多いので、かなを効率よく入力することができません。

日本語の漢字の単語は、「きょう」や「しゅう」のように拗音を含む読みをもつものが多いという特徴があります。また、語尾が「い」「う」「ん」「く」「つ」などの単語が多いという特徴もあります。一般的な日本語入力手法では、これらの特徴をほとんど考慮していません。ローマ字入力の場合は、むしろこれらの読みが入力しづらくなっています。

このような問題は従来から指摘されており、改善するための各種の手法が提案されています。思いつくままに、そのいくつかを簡単に紹介しましょう。

拡張ローマ字記法

ローマ字による日本語の標準的な記法では、l や q といった文字は使われませんし、促音を表現する場合を除けば、子音キーを連続して入力することもありません。このようなキー列を特定のかな文字列に割り当てれば、より効率的にかな入力ができる可能性があります。

AZIK

AZIK[1]は、尚絅女学院短期大学の木村 清さんが開発した拡張ローマ字入力手法です。AZIK は一般的なローマ字入力法を基本としていますが、標準のローマ字で使われない l や q などのキーを使ったり、連続した子音文字を使って撥音や拗音を表現することにより、日本語の読みを効率よく入力できるようになっています。

AZIK では、訓令式のローマ字表現に、いくつかの拡張が加えられています。おもなものを以下に紹介します。

促音：〝っ〞はつねに〝;〞で入力する。

しゃ/ちゃ：〝しゃ〞行を入力する場合は〝sy〞もしくは〝x〞と母音を、〝ちゃ〞行を入力するには〝ty〞もしくは〝c〞と母音を組み合わせる。
例：xa →しゃ、ca →ちゃ

撥音拡張：2 文字目に〝ん〞がくる場合、子音に続けて z、k、j、d、l（エル）を入力し、それぞれ an、in、un、en、on を表現する。
例：kk → (kin) →きん

二重母音拡張：子音に続けて q、h、w、p を入力し、それぞれ ai、uu、ei、ou を表現する。
例：kh → (kuu) →くう

互換キー：同じ指で連続打鍵が必要な場合などに別のキーで代用する。
例：通常のキー入力では〝hya →ひゃ〞だが、AZIK では〝hga →ひゃ〞とする

この方式であれば、〝ちゅうしょうきぎょう〞という読みを、

```
c h x p k i g y p
```

というキー列で入力できます。

AZIK では多くの規則が定義されているので、そのすべてを憶えるのは容易ではありません。ただし、訓令式ローマ字入力を拡張するかたちになっているため、それらの表現を完全に暗記しなくても使えます。

効率的なローマ字かな入力手法

九州大学の近藤哲哉氏も、ローマ字方式を基本としつつ、日本語の読みを効率的に入力できるように拡張したシステムを発表しています[2]。この手法では、ローマ字表現に以下のような拡張が加えられています。

う段：子音に続く u は省略可能とする。
例：tkr →つくる、kakka →かくか

促音拡張：〝っ〞は〝x〞を使う。
例：kaxka →かっか

拗音拡張：あ段〜お段の拗音は、子音に続く j、v、q、l、f で表現する。
例：sj →しゃ

長音拡張：長音は母音を重ねて入力する。
例：koohii →こーひー、rooma →ろーま

この方式では、〝ちゅうしょうきぎょう〞を次のようにして入力できます。

```
c q u s f u k i g f u
```

特殊なかなキーボード

ローマ字によるかな入力は、かな入力にくらべるとどうしても打鍵回数が増えます。そもそも、ローマ字にさえ馴染みのない人にとっては、拡張ローマ字を使うのはかなりの負担になるかもしれません。

一方、JIS かな配列や NICOLA 配列のキーボードでは読みがばらばらに配置されているので、慣れるまでに時間がかかるという問題があります。このような問題を解決するために、各種の日本語用かなキーボードが提案されています。

TRON キーボード

東京大学の坂村 健教授が主導する TRON プロジェクトでは、BTRON サブプロジェクト[3]において、日本語入力の負担を軽減する「TRON キーボード」（図 1）の仕様を決めています。

この TRON キーボードは以前に製品化されたこともあって、熱狂的なファンも多いようです。しかし、現在は製造・販売されていないため、再度製品化を望む声が高まっていました。最近、ユーザーの要望が〝商品化お頼みサイト〞の tanomi.com[4]を通じて結実し、TRON キーボー

1 http://hp.vector.co.jp/authors/VA002116/azik/azikindx.htm

2 http://cc.vbl.kyushu-u.ac.jp/cc/cc2000/romaji/

3 http://tron.um.u-tokyo.ac.jp/TRON/proj95/BTRON.html

4 http://www.tanomi.com/items/tron/

図 1　TRON キーボード配列

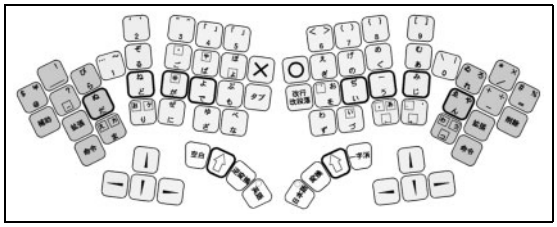

図 2　M 式キーボード

ドの限定製造販売がおこなわれることになったようです。

M 式キーボード

元・日本電気の森田正典氏は、長年にわたり独自の「M 式キーボード」（図 2）を提唱しています[5]。この方式にもとづいた製品も、日本電気から販売されています。

M 式キーボードは、子音を右手で、母音を左手で入力することで、日本語の読みの効率的な入力を実現しようというものです。さらに、シフトキーを使って、拗音や二重母音を簡単に入力できるようになっています。

ナラコード

ナラコムの奈良總一郎氏は、拗音を含む読みをとくに入力しやすくする「ナラコード」というかなキーボード配列を提唱しています[6]。

ナラコード配列のキーボード（図 3）は、五十音順にキーが配置され、初心者に分かりやすいという特徴があります。さらに、〝しょう〞や〝にゅう〞などの出現頻度の高い読みが、そのままキーに割り当てられているため、さきほどから例に挙げている〝ちゅうしょうきぎょう〞であれば、4 回の打鍵で入力することができます。

奈良氏は、全国 160 万の企業のうちの 50 万社では PC

5 http://121ware.com/apinfo/content/mworld/
6 http://www.naracom.co.jp/naracode/what.html

図 3　ナラコード配列のキーボード

図 4　「花」配列

を導入していないが、これはローマ字によるキー入力ができないからであり、ナラコード配列のキーボードを使えば、この問題は乗り越えられると述べています。

「花」配列

冨樫雅文氏は、日本語の読みの出現頻度と指の動きの統計情報にもとづき、日本語入力の労力を軽減する「花」というキーボード配列（図 4）を提案しています[7]。

キートップの左側に表示されている文字は、ホームポジションの中指のキーを押したあとで別のキーを押す〝中指シフト〞で入力するようになっています。たとえば、ASCII キーボードの k と s を順に押すと、〝よ〞が入力されます。

ペンストロークの応用

ここまでに紹介した各種の手法は、そのままペン計算機のソフトキーボードでも使うことができます。しかし、ほとんどのソフトキーボードではシフト操作などが簡単にはできません。そもそもホームポジションという概念がないので、そのままのかたちで適用するのは賢明とはいえないでしょう。

一方、ペン計算機ではペンジェスチャー（いわゆる手描き入力）も使えるので、これとソフトキーボードとを組み合わせれば、かなを効果的に入力できるようになるかもし

7 http://member.nifty.ne.jp/togasi/hana_no_kuni/

図 5　Palm の Graffiti エリアに貼るシール

図 6　CLIE に貼ったシール

れません。

以下では、ペン計算機のかなソフトキーボードによる日本語の効率的な入力を目指して開発した「PalmKanaKB」を紹介します（末尾のリスト 1 もあわせてご覧ください）。

PalmKanaKB

Palm での日本語入力は、標準では Graffiti という英数字認識システムと、ローマ字かな変換の組合せを使うことになっています。しかし、通常のローマ字ではどうしても入力する文字が多くなるため、時間がかかってしまいます。さらに、入力エラーも多くなりがちです。キーボードボタンを押して五十音のソフトキーボードを表示させて使うこともできますが、毎回かなキーボードを表示させるのは面倒です。

そこで、PalmKanaKB では図 5 のシールを文字認識エリアに貼り付け、かなキーボードとして使う仕組みにしてみました。

今回紹介した各種のシステムの開発者たちも述べていますが、漢字の音読みは、二重母音で終るもの（かい、こう、など）や語尾に拗音を含む読み（しょう、きょう、など）がくるものが多数を占めています。これらをスムーズに入力できれば、全体としての入力効率もかなり上がるのではないでしょうか。

そのために、AZIK では二重母音や拗音を含む読みを子音に割り当て、ナラコードでも〝しょう〞や〝にゅう〞などの出現頻度の高い読みに特殊なキーを割り当てていました。一方、PalmKanaKB では、五十音のソフトキーボードをタップしたあとでペンを上下左右に動かすことにより、二重母音や拗音を含む読みを語尾に追加できるソフトキーボードを作ってみました。

PalmKanaKB は、以下のような機能をもっています。

ジェスチャーによる語尾追加

語尾に頻出する〝ん〞〝い〞〝う〞などを簡単に入力できるようにするため、以下の規則を適用します。

- かなキーのタップ後にペンを右に動かすと〝い〞を追加
- かなキーのタップ後にペンを下に動かすと〝ん〞を追加
- かなキーのタップ後にペンを上に動かすと〝う〞を追加

たとえば、〝か〞のキーをタップ後、右にペンを動かしてから離すと〝かい〞と入力され、下にペンを動かしてから離すと〝かん〞と入力されます。

次に、〝きゃ〞や〝きょ〞といった拗音を入力しやすくするために、以下の規則を適用します。

- あ段のキーをタップ後にペンを左に動かすと〝ゃ〞を追加
- う段のキーをタップ後にペンを左に動かすと〝ゅ〞を追加
- お段のキーをタップ後にペンを左に動かすと〝ょ〞を追加

たとえば、〝か〞をタップしてから左にペンを動かして離すと〝きゃ〞が、〝こ〞をタップしてから左にペンを動かして離すと〝きょ〞が入力されます。

また、〝きゅう〞や〝きょう〞のように、〝ゅう〞または〝ょう〞で終る読みはとくに多いので、以下の例外規則も設けました。

- い段のキーをタップ後にペンを上に動かすと〝ゅう〞を追加
- え段のキーをタップ後にペンを上に動かすと〝ょう〞を追加

図 7　KanaKB による入力

ちゅう	゛ち゛をタップして上にドラッグ
しょう	゛せ゛をタップして上にドラッグ
き	゛き゛をタップ
ぎょう	゛け゛をタップし、一瞬おいてから上にドラッグ

たとえば、゛き゛をタップしてから上にペンを動かして離すと（゛きう゛ではなく）゛きゅう゛が入力され、゛け゛をタップしてから上にペンを動かして離すと（゛けう゛でなく）゛きょう゛が入力されます。

長タップによる濁点追加

ソフトキーボードでは、濁点の入力が意外に面倒です。そこで、ペンタップの時間によって濁点を指定できるようにしてみました。たとえば、゛き゛をタップしたまま、しばらくしてからペンを離すと゛ぎ゛と入力されます。

超長タップによる促音追加

ソフトキーボードでは、促音の入力もそれほど容易ではありません。そこで、ペンタップの時間によって促音も指定できるようにしてみました。たとえば、゛き゛をタップしたまま、かなり時間をおいてからペンを離すと゛っき゛と入力されます。

■ ■ ■

これらの手法を組み合わせると、゛ちゅうしょうきぎょう゛という文字列は 4 回の操作で入力できます（図 7）。

KanaKB の使用感

現在、PalmKanaKB を POBox と組み合わせて使っていますが、ジェスチャーを用いた語尾追加により、漢字熟語の入力が格段に高速になりました。

Graffiti でローマ字を使うか、かなキーボードを使うかは好みが分かれるところだと思います。しかし、すくなくとも私の場合は Graffiti の認識精度があまりよくないので、かなキーボードを使うほうが高速に入力できるようです。

おわりに

初心者でも抵抗なく使えて、習熟するにしたがって高速に日本語が入力できる手法は、今後ますます重要になってくるでしょう。現在はローマ字入力やかなキーボードが主流ですが、これらがかならずしも最善のものとは思えないので、新たによい方法が出てくる余地はまだまだ残っているのではないでしょうか。

なお、今回紹介した AZIK や花などの入力手法は、AZIK を開発した木村氏の Web ページ[8]の Java アプレットで体験できます。

PalmKanaKB は私の Web ページ[9]で公開していますので、ぜひお試しください。

（ますい・としゆき　ソニー CSL）

8 http://hp.vector.co.jp/authors/VA002116/otamesiindex.html
9 http://www.csl.sony.co.jp/person/masui/PalmKanaKB/

リスト 1　PalmKanaKB ソース

```
#include <PalmOS.h>
#include "pbinline.h"

#define LEFT 25
#define RIGHT 134
#define TOP 169
#define BOTTOM 222

#define Queue(c) EvtEnqueueKey(c,0,0)

enum {
        KANAKBMODE = 0,
        SCREENX = 1,
        SCREENY = 2,
        SYMBOLSHIFT = 3,
        TICKS = 4,
        TAPLATENCY = 5,
        KANAMODE = 6
```

```
};

typedef void EvtGetEventTrapFunc(EventType*, Int32);
typedef Err EvtProcessSoftKeyStrokeFunc(PointType *start, PointType *end);

void QueueAIUEO(int ycol);

Err KanaKB(PointType *start, PointType *end)
{
        EvtProcessSoftKeyStrokeFunc *trapfunc;
        UInt32 ftrvalue;
        Int16 xd,yd,xu,yu,x,y;
        UInt32 dist;

        UInt32 ticks;
        UInt32 msec;
        UInt16 slowtap;
        Int16 r;
        UInt16 xcol,ycol;
        Boolean valid;

        // システムの漢字変換はつねにOFFにする <a name="fepoff">
        TsmSetFepMode(NULL,tsmFepModeOff);

        xd = start->x;
        yd = start->y;
        xu = end->x;
        yu = end->y;

        x = xu-xd;
        y = yu-yd;
        y = -y;
        dist = x*x+y*y;

        r = -1;
        slowtap = 0;
        if(dist < 3){
                // ペン移動なし
                FtrGet(CREATOR,TICKS,&ticks);
                msec = (TimGetTicks()-ticks) * 1000 / SysTicksPerSecond();
                slowtap = (msec > 600 ? 2 : msec > 200 ? 1 : 0);

                r = 0;
        }
        else if(dist < 300){
                // ペン移動あり
                FtrGet(CREATOR,TICKS,&ticks);
                msec = (TimGetTicks()-ticks) * 1000 / SysTicksPerSecond();
                slowtap = (msec > 800 ? 2 : msec > 300 ? 1 : 0);

                // ペン移動方向の認識。右方向がr=1、あとは反時計回りにr=2〜4
                if(x > 0){
                        if(y > x){ r = 2; }
                        else if(y * -1 > x){ r = 4; }
                        else { r = 1; }
                }
                else {
                        x = -x;
```

```
                if(y > x){ r = 2; }
                else if(y * -1 > x){ r = 4; }
                else { r = 3; }
        }
}
if(r >= 0){
        xcol = (xd-LEFT) * 10 / (RIGHT-LEFT);
        ycol = (yd-TOP) * 5 / (BOTTOM-TOP);

        // POBoxのon/off指令 <a name="poboxonoff">
        if(xcol == 0 && ycol == 4){
                EvtEnqueueKey(TOGGLECHR,0,0);
                FtrSet(CREATOR,KANAKBMODE,0);
                return 0;
        }
        valid = true;
        // 特定の場所をタップした場合だけ記号シフト モード と解釈
        if(xcol == 2 && ycol == 3){
                valid = false;
        }

        if(valid){
                switch(xcol){
                case 0: if(r == 0){
                        switch(ycol){
                                case 0: Queue('w'); Queue('a'); break;
                                case 1: Queue('w'); Queue('o'); break;
                                case 2: Queue('n'); break;
                                case 3: Queue('.'); break;
                                default: break;
                                }
                        }
                        else {
                                Queue('w');
                        }
                        break;
                case 1: Queue('r');
                        break;
                case 2: if(r == 0){
                                switch(ycol){
                                case 0: Queue('y'); Queue('a'); break;
                                case 1: Queue('-'); break;
                                case 2: Queue('y'); Queue('u'); break;
                                case 4: Queue('y'); Queue('o'); break;
                                default: break;
                                }
                        }
                        else {
                                Queue('y');
                        }
                        break;
                case 3: if(slowtap > 1) Queue('m');
                        Queue('m'); break;
                case 4: if(slowtap > 2){ Queue('p'); Queue('p'); }
                        else { Queue(slowtap > 1 ? 'p' : slowtap ? 'b' : 'h'); }
                        break;
                case 5: Queue('n');
                        break;
```

```
                        case 6: if(slowtap > 1){ Queue('t'); Queue('t'); }
                                else { Queue(slowtap ? 'd' : 't'); }
                                break;
                        case 7: if(slowtap > 1){ Queue('s'); Queue('s'); }
                                else { Queue(slowtap ? 'z' : 's'); }
                                break;
                        case 8: if(slowtap > 1){ Queue('k'); Queue('k'); }
                                else { Queue(slowtap ? 'g' : 'k'); }
                                break;
                        default: break;
                        }
                        switch(r){ // 撥音、促音など <a name="bindingcode">
                        case 0: if(xcol != 0 && xcol != 2) QueueAIUEO(ycol); break;
                        case 1: QueueAIUEO(ycol); Queue('i'); break;
                        case 2: switch(ycol){
                                case 0: case 2: case 4:
                                        QueueAIUEO(ycol); Queue('u'); break;
                                case 1:
                                        Queue('y'); Queue('u'); Queue('u'); break;
                                case 3:
                                        Queue('y'); Queue('o'); Queue('u'); break;
                                }
                                break;
                        case 3: if(xcol != 0 && xcol != 2){
                                        switch(ycol){
                                        case 0: Queue('y'); Queue('a'); break;
                                        case 1: Queue('y'); Queue('u'); Queue('u'); break;
                                        case 2: Queue('y'); Queue('u'); break;
                                        case 3: Queue('y'); Queue('o'); Queue('u'); break;
                                        case 4: Queue('y'); Queue('o'); break;
                                        }
                                }
                                break;
                        case 4: QueueAIUEO(ycol); Queue('n'); break;
                        default: break;
                        }
                        return 0;
                }
        }

        // オリジナルのEvtProcessSoftKeyStroke呼出し
        FtrGet(CREATOR,1000,&ftrvalue);
        trapfunc = (EvtProcessSoftKeyStrokeFunc*)ftrvalue;
        return (*trapfunc)(start,end);
}

void QueueAIUEO(int ycol)
{
        switch(ycol){
        case 0: Queue('a'); break;
        case 1: Queue('i'); break;
        case 2: Queue('u'); break;
        case 3: Queue('e'); break;
        case 4: Queue('o'); break;
        default: break;
        }
}
```