インターフェイスの街角 (8)
ズーミング・インターフェイス

増井俊之

4 月末にロサンゼルスで開催された、ACM (Association for Computing Machinery) の CHI '98 コンファレンス[1]に参加してきました。CHI (Computer-Human Interaction) コンファレンスはユーザー・インターフェイスに関する世界最大の国際会議で、今回は 2,500 人もの参加者がありました。

会議の冒頭でキーノート・スピーチに立った University of Maryland の Ben Shneiderman[2]は、PowerPoint やスライドではなく、前回紹介したズーミング視覚化システム「Pad++」を用いてプレゼンテーションをおこないました。スライド画面に慣れた聴衆のあいだで、ダイナミックにズーミングする画面やアニメーションを駆使した発表が人気を集めていました。

今回は、この Pad++システムについて解説します。

ズーミング・インターフェイス

小さな画面上に多くの情報を配置するための手法として、現在は複数のウィンドウを画面上に重ねて配置するウィンドウ・システムがひろく使われています。これに対し、画面を重ねて配置するのではなく、拡大・縮小によって大量の情報を見たり操作したりする手法をズーミング・インターフェイス (Zooming Interface) といいます。現在のウィンドウ・システムでも、アイコン化のような画面の拡大・縮小手段をある程度は支援していますが、ズーミング・インターフェイスをもつシステムではすべての対象を連続的に拡大・縮小することによって画面を有効に活用できます。

1 http://www.acm.org/sigchi/chi98/
2 http://www.cs.umd.edu/users/ben/index.html

図 1　Shneiderman のプレゼンテーション (一部)

上図の白枠部分をクリックすると、〝Memex-Genex〟の中間部が下図のようにアニメーションで拡大されて、隠れていた部分が現れる。

ズーミング・インターフェイスの特徴

滑らかな拡大/縮小が可能なズーミング・インターフェイスシステムは、大量の情報を扱えるだけでなく、あらゆる操作が連続的/可逆的になるという大きな特徴をもって

います。現在普及しているウィンドウ・システムでは、ウィンドウをアイコン化するときとアイコンをウィンドウに戻すときの操作が違いますし、ウィンドウを連続的に縮小してアイコン化するようなことはできません[3]。これに対し、ズーミング・インターフェイスの場合は連続的/可逆的な拡大・縮小操作を用意するだけでよいわけです。

日常生活で使われている機械や道具類の大半は、連続的かつ可逆的に操作できます。たとえば自動車の場合、ハンドルの回転量に比例して連続的に向きを変えていくので、ハンドルを逆に回せば方向を戻せます。だからこそ、直感的かつ容易に操作できるのです。一方、現在の計算機は、前述のようにほとんどの操作が可逆的になっていないことが使いにくいと感じさせる大きな要因の 1 つといえるでしょう。画面上のボタンを押すと何が起こるか分からず、しかも元に戻せない可能性もあるというのでは、とても気軽には扱えません。

こういう状況を改善するためにも、ズーミング・インターフェイスのような連続的/可逆的な操作ができるインターフェイス手法は今後普及していく可能性があるのではないでしょうか。

Pad++システム

Pad++[1] は、New York University の Ken Perlin が 1993 年に提案した 2 次元ズーミング・システム「Pad」[2] を、University of New Mexico の Ben Bederson[4]が拡張したシステムです。

現在、Pad++は Tcl/Tk を拡張したツールキットとして実装されて公開されています[5]。Tk の Canvas オブジェクトを連続的に拡大・縮小できるようにした Pad オブジェクトを用いて Tcl のプログラムを動かすことができます。Pad++では、通常の Tcl/Tk で使われるシェル wish の代わりに **padwish** というシェルが用意されています。

ツールキットと padwish のほかに、Pad++には描画エディタ **Paddraw** が付属しています。これは、Pad++ツールキットを利用した約 26,000 行の Tcl スクリプトです。Paddraw は任意のズーミングが可能なドロー系エディタとして使えるのはもちろん、ハイパーリンク機構も備えているので、画面上のオブジェクトをクリックして視点を別のオブジェクトに移動するといった指定を簡単におこなえます。たとえば、画面上にスライド画像を並べ、それらをリンクで結んでおけば、市販のプレゼンテーション・ツールのようにマウスクリックにより順番にスライドを切り替えていくことができます。

Paddraw は、任意の Tcl プログラムを読み込めるだけでなく、Paddraw で作成したデータも Tcl プログラム形式で保存されるので、データやプログラムの読込みや再利用も簡単です。たとえば、矩形を 100 個描く Tcl プログラムを読み込めば、Paddraw の画面に 100 個の矩形が描けます。Paddraw は、いわば描画編集機能付きの PostScript ビューアのようなものといえるでしょう。

Pad++プログラミング

まず、Tcl から Pad++ツールキットを扱う方法を簡単に説明します。

padwish の起動

padwish コマンドを起動すると Padwish ウィンドウが表示され、Tcl のプロンプトが表示されます。

```
% padwish
%
```

Pad++は Tk の部品として実装されており、padwish は wish の拡張となっています。したがって、padwish では普通の Tcl/Tk と同様のプログラミングが可能です。

```
% set x 10
10
% puts abc
abc
%
```

Pad++のマニュアルは 100 ページもあり、すべての機能はとりあげられません。基本的な機能に絞って以下に紹介します。

Pad++オブジェクトの生成と表示

Pad++画面を表示する場合は、通常の Tcl/Tk で Canvas オブジェクトを作る場合と同様に Pad オブジェ

3 連続的ではなく〝跳躍的〟という感じでしょうか。

4 http://www.cs.umd.edu/~bederson/

5 Bederson が University of Maryland に移ったため、現在は http://www.cs.umd.edu/hcil/pad++/ で公開されています。IRIX、SunOS、Solaris、Linux など各種の UNIX 版と Windows 95 版がありますが、Windows 版は現時点では動作が遅く、あまり安定していません。

図 2　矩形の表示

```
% .pad create rectangle 0 0 50 50
2
%
```

クトを作成、表示します。たとえば、下記の pad コマンドを実行すると、200 × 200 の大きさの ``.pad'' という名前の Pad オブジェクトが生成され、pack コマンドで画面に表示されます。

```
% pad .pad -width 200 -height 200
.pad
% pack .pad
%
```

生成された Pad オブジェクトについて、数多くのコマンドやサブコマンドが用意されているため、きめ細かに制御できます。

図形の生成

図形やウィジェットは **create** コマンドで生成します。たとえば、図 2 のようにすると、大きさ 50 × 50 の矩形が原点に表示されます。

create コマンドは、生成した矩形の ID 番号を返します（この例では ``2''）。これは、あとで図形を参照するときに使えます。create コマンドに **-tags** オプションを指定し、図形に名前を付けることもできます。

図 3 のように for 文を直接指定すれば、複数の図形を一度に生成できます。

移動とズーム

画面上で視点を移動するには **moveto** コマンドを使います。moveto では、画面の中心に表示される点の座標と画面の拡大率を引数に指定します。padwish の起動時には、これらの値は ``(0, 0, 1)'' になっています。

図 3 で視点を変えずに拡大率を変化させると、画面は図 4 のように変化します。

図 3　複数の矩形を表示

```
% for {set x -50} {$x <= 50} {
set x [expr $x + 50]} {
for {set y -50} {$y <= 50} {
set y [expr $y + 50]} {
.pad create rectangle $x $y \
[expr $x + 20] [expr $y + 20]
}}
%
```

図 4　画面の拡大

```
% .pad moveto 0 0 1.8
%
```

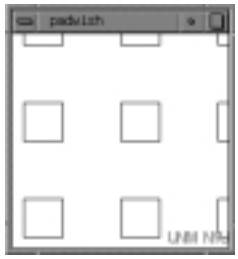


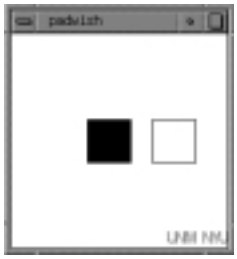
図 5　画面の縮小/移動のアニメーション
（1 秒間で、この図のように変化）

```
% .pad moveto 50 0 0.7 1000
%
```

moveto コマンドに 4 番目の引数を指定すると、その時間（単位はミリ秒）ぶんのアニメーションを表示しながら移動・拡大・縮小がおこなわれます（図 5）。

図 6　黒い矩形をクリック

図 7　白い矩形をクリック



イベント・バインディング

図形にイベント処理を割り当てるときは、**bind** コマンドを使います。

```
% set rect1 [.pad create rectangle\
-50 -20 -10 20 -fill black]
% set rect2 [.pad create rectangle\
10 -20 50 20 -fill white]
% .pad bind $rect1 <ButtonPress> {
.pad moveto 20 0 1 1000}
% .pad bind $rect2 <ButtonPress> {
.pad moveto -20 0 1 1000}
%
```

黒い矩形をクリックすると画面は図 6 のように変化し、白い矩形をクリックすると図 7 のように変化します。

Paddraw

padwish では GUI 操作は定義されていませんが、図形の描画などの処理を Tcl で記述した Paddraw プログラムが用意されています。

Paddraw の起動

paddraw コマンドを実行すると、padwish に pad.tcl というファイルが読み込まれて Paddraw が起動し[6]、図 8 のような画面が表示されます。ここでは、Pad の開発元である University of New Mexico のロゴなどが表示されています。この画面は、数秒後に画面いっぱいに拡大されたあと消えてしまいます。

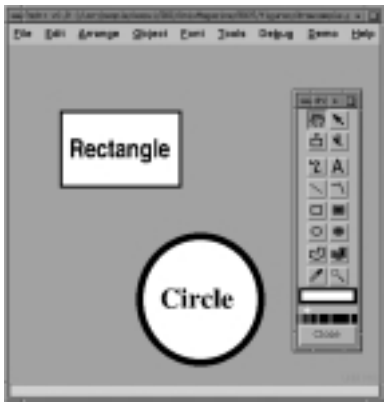
図形編集

Tools メニューから描画ツールを選択し、一般のドローツールと同様に絵を描くことができます（図 9）。

6 Paddraw プログラムは、padwish から pad.tcl を呼び出すシェル・スクリプトです。

図 8　Paddraw 起動時の画面

図 9　図形の描画



ズーミング操作

Paddraw では、マウスの中央ボタンと右ボタンを使って画面や図形を連続的に拡大・縮小させることができます。描画ツール右上の矢印ボタンを選択すると選択モードになり、マウスの左/中央/右ボタンを使って図形を個別に移動・拡大・縮小させることができます。このモードで、図 9 の矩形を左ボタンで選択してから中央ボタンを押すと、図形を拡大できます（図 10）。マウスカーソルの場所を中心として、図形は中央/右ボタンを押しているあいだはどんどん拡大・縮小していきます。

描画ツール左上の〝掌〟ボタンを押すとブラウズモードになり、マウスの左/中央/右ボタンを使って画面全体を移動・拡大・縮小させることができます。このモードでは、たとえば矩形と円のあいだで中央ボタンを押し、画面全体を拡大することができます（図 11）。

図 10　図形の拡大

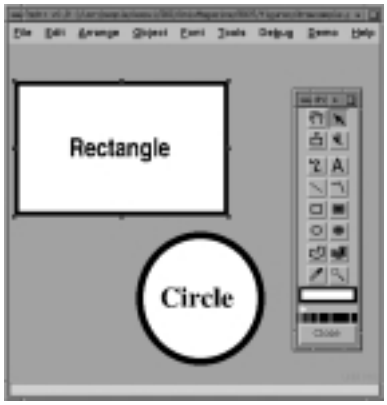


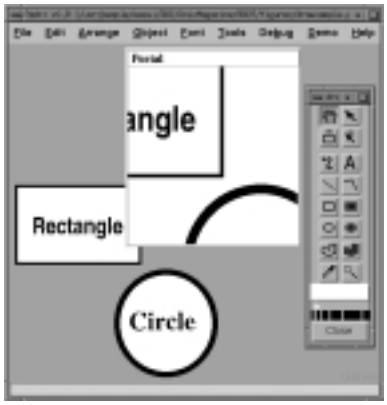
図 11　画面全体の拡大

図 12　アニメーション

アニメーション

アニメーション・ツールでは、マウスクリックにより起動されるアニメーションを定義することができます。図 12 は、選択モードで黒丸が矩形の周囲を回るというアニメーションを定義しているところです。このように定義しておき、ブラウズモードで黒丸をクリックすると定義した経路の上を黒丸が動きます。

図 13　Portal



Sticky

ズーミング操作をおこなうと、通常はすべての図形が同時に拡大・縮小されますが、つねに同じ大きさの図形を画面上に置いておきたい場合があります。Object メニューで図形に Sticky 属性を指定すれば、ズーミング操作にかかわらずその図形の大きさは変わりません。

Portal

同時に複数の場所を参照したい場合には、画面上に ¨Portal¨という特別な枠を設けることができます。図 13 は、Portal 枠を使って図形の一部を拡大表示しているところです。Portal の内側と外側で独立にズーミングをおこなうこともできます。

ブラウズモードで画面全体を拡大・縮小すると Portal の大きさも変わってしまいますが、Portal を Sticky にしておくと、全体を同じ大きさの Portal で参照しながら一部を拡大して表示するといったことが可能になります。この場合、画面を拡大・縮小しても Portal 枠内の全体表示画面の大きさは変わりません（図 14）。

ハイパーリンク

描画ツール右側の上から 2 つ目のボタンを使って、図形間にハイパーリンクを定義することができます（図 15）。この例では、矩形から円へのリンクと、円から矩形へのリンクを定義しています。ブラウズモードでリンクが定義さ

図 14　固定サイズの Portal による全体表示

図 15　ハイパーリンクの定義

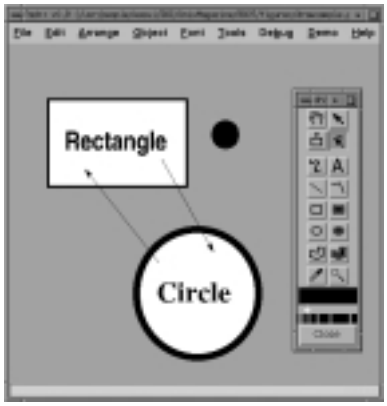


図 16　矩形をクリックして円に移動

図 17　円をクリックして矩形に移動

図 18　Paddraw によるスライド作成

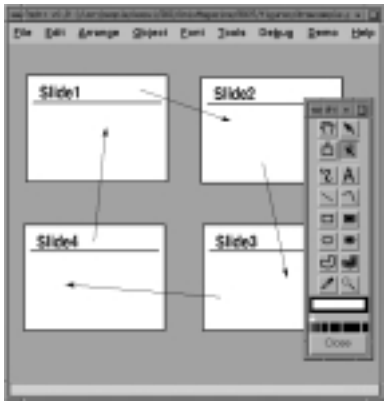
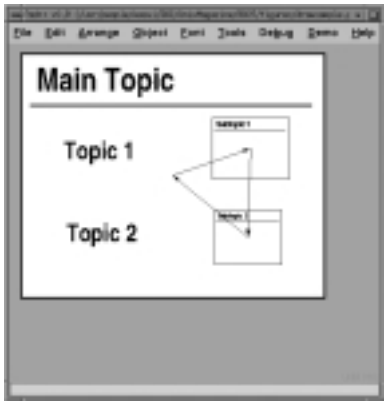


図 19　ズーミングするスライド



れている図形をクリックすると、リンク先の図形が画面いっぱいに表示されます。たとえば矩形をクリックすると、全体図が滑らかに移動しながら拡大して円が画面いっぱいに表示されます（図 16）。この状態で円をクリックすると、画面が移動して今度は矩形が画面いっぱいに表示されます（図 17）。

このハイパーリンク機構を利用して、プレゼンテーション用の資料を簡単に作ることができます。スライドを用意

図 20　Paddraw 出力ファイル例

```
# 0.9
global Pad_ObjectList PADLOAD
set Pad_ObjectList ""

$PADLOAD config -background #8ea8ab

set Pad_ID [$PADLOAD create rectangle -226.5 209 -0.500001 67 \
-events 1 -layer main -rposition "-125.52 140.994 0.975416" \
-tags "item link2 linkdest" -fill #ffffff -joinstyle miter \
-pen #000000 -penwidth 5]
set Pad_NewID(974) $Pad_ID
lappend Pad_ObjectList $Pad_NewID(974)
set Pad_ID [$PADLOAD create oval -95.7899 4.59107 96.2922 -197.605  \
-anchor center -events 1 -layer main -rposition "0.251183 -96.507 1" \
-tags "item link1 linkdest link linkto2" -linkto link2 -fill #ffffff \
-joinstyle miter -pen #000000 -penwidth 11.5]
set Pad_NewID(975) $Pad_ID
lappend Pad_ObjectList $Pad_NewID(975)
set Pad_ID [$PADLOAD create text  -anchor nw -events 1 -layer main \
-rposition "-208.728 172.221 29.6278" -tags "item text link linkto1" \
-linkto link1 -font Helvetica-bold-1 -pen #000000 -text Rectangle]
set Pad_NewID(976) $Pad_ID
lappend Pad_ObjectList $Pad_NewID(976)
set Pad_ID [$PADLOAD create text  -anchor nw -events 1 -layer main \
-rposition "-63.5719 -67.5237 31.0265" -tags "item text" \
-font Times-bold-1 \-pen #000000 -text Circle]
set Pad_NewID(977) $Pad_ID
lappend Pad_ObjectList $Pad_NewID(977)

$PADLOAD itemconfigure 1 -visiblelayers "all"

set view [$PADLOAD getview]
$PADLOAD moveto -39.3142 47.9664 0.918069
if [info exists Pad_NewID(18)] {$PADLOAD ic $Pad_NewID(18) -sticky 1}
if [info exists Pad_NewID(3)] {$PADLOAD ic $Pad_NewID(3) -sticky 1}
if [info exists Pad_NewID(2)] {$PADLOAD ic $Pad_NewID(2) -sticky 1}
eval $PADLOAD moveto $view
```

し、図 18 のようにハイパーリンクで結んでおけば、スライドをマウスでクリックすると、アニメーションで次のスライドに移動させることができます。

図 18 のように同じ大きさのスライドを使う場合は次のスライドに移るときに画面が移動するだけですが、図 19 のようにいろいろな大きさのスライドを用意しておけば、画面を拡大してサブトピックに移動したり、縮小して元のスライドに戻ったりといったダイナミックなプレゼンテーションができます。

ファイル形式

File メニューを使用して図 15 をセーブすると、図 20 のようなテキストファイルが出力されます。図形の属性やハイパーリンクなどが Tcl のコマンド列として定義されていることが分かります。

外部からの Pad++の制御

Pad++システムでは、基本的には Tcl でプログラムを記述します。しかし、別のプログラムで Pad を制御する Tcl スクリプトを生成して padwish に送り、Pad を表示するためのバックエンドとして使うこともできます（NeXT において、C で PostScript を生成するのに似ています）。padwish の起動直後に ~pad.tcl~ を読み込んでおくと、Paddraw も制御できるようになります。たとえば、次のプログラムを実行してパイプで padwish に接続すれば、図 3 と同じ図が Paddraw 上に表示され、Paddraw 上で編集できます。

```
#include <stdio.h>

/* Padホームディレクトリ（適当に修正） */
#define PADHOME "/usr/local/pad"

void rect(int x, int y, int width, int height)
{
  printf(".pad create rectangle %d %d %d %d\n",
    x, y, x+width, y+height);
  fflush(stdout);
}

main()
{
  int x, y;
  /* Paddraw起動 */
  printf("source \"%s/draw/pad.tcl\"\n",
    PADHOME);
  for(x=-50;x<=50;x+=50){
    for(y=-50;y<=50;y+=50){
      rect(x,y,20,20);
    }
  }
  ……
}
```

おわりに

今回は、ズーミング・インターフェイスを支援するPad++システムを紹介しました。ホームページこそ地味ですが、Pad++は機能が充実しており、プレゼンテーションに使うとかなり派手な効果が得られます。UNIX上での情報視覚化やプレゼンテーションなどに、ぜひ一度試してズーミング・インターフェイスを体験していただきたいと思います。Pad++のページでは、Shneidermanが CHI '98でのプレゼンテーションに使った資料も公開されているので、参考にするといいでしょう。

（ますい・としゆき　ソニーCSL）

[参考文献]

[1] Benjamin B. Bederson and James D. Hollan, "Pad++: A zooming graphical interface for exploring alternate interface physics", *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pp.17–26, ACM Press, November 1994

[2] Ken Perlin and David Fox, "Pad: An alternative approach to the computer interface", *ACM SIGGRAPH'93 Conference Proceedings*, pp.57–64, August 1993

おまけ：スライド生成プログラム

Paddraw のハイパーリンク機能を使ってプレゼンテーションの資料を作るとき、すべてのスライドを Paddraw で描いてリンクを張るのは大変です。下記のプログラム **text2padslide** を使って、次のようなテキストファイルから Tcl スクリプトを生成すれば、右図のようなスライド画面を自動生成することができます。

生成されたスライド

●スライド用のテキストファイル

```
SLIDE1
  Item1
    Item1.1
    Item1.2
  Item3
  Item4
SLIDE2
  Item1
  Item2
SLIDE3
  Item1
  Item2
  Item3
  Item4
```

● text2padslide プログラム

```
#!/usr/local/bin/perl
print ".pad config -background #808080\n";
while(<>){
  chop;
  if(/^\S/){
    if($n != 0){
      print ".pad coords slide$n 0 $slidey 400 $y\n";
    }
    $n++;
    $y -= 70;
    $slidey = $y;
    $n1 = $n+1;
    print ".pad create rectangle  0 0 10 10 -events 1";
    print " -layer main -fill white -tags \"link linkdest ".
          "link$n linkto$n1 slide$n\" -linkto link$n1\n";
    $y -= 10;
    print ".pad create text -anchor nw -rposition \"20 $y 2 \" ".
          "-font Helvetica-Bold -text \"$_\"\n";
    $y -= 50;
    print ".pad create line 10 $y 390 $y\n";
    $y -= 10;
  }
  else {
    print ".pad create text -anchor nw -rposition \"-20 $y 2\" ".
          "-font Helvetica -text \"$_\"\n";
    $y -= 50;
  }
}
print ".pad coords slide$n 0 $slidey 400 $y\n";
```