本稿は Linux Japan 誌 1999 年 1 月号に掲載された記事に補筆修正したものです.

仮題：スクリーンセーバーの自作

X11R6 でアニメーションという難問に対して，2 通りの道を紹介しようと思います．一つは，図形の組み合わせで画面をその都度計算・構成していく方法です．OpenGL などを解説できるとかっこいいのですが，筆者にはその力がありません．Xlib のプリミティブなグラフィック描画関数を使った例で勘弁してください．もっとも，この教科書レベルの話は X のプログラミングの入門のつもりです．

もう一つは，予め描いておいた画像を次々表示して動画にするという，いわゆるセル画によるアニメーションです．セル画の保存形式に，前回説明した Xpm を用いると，Xpm ライブラリにより X の各窓にその画像が簡単に表示できるようになります．なぜなら，X が扱えるイメージ構造体 XImage への変換やイメージが描かれた Pixmap の生成といった，もっとも手間のかかる部分がライブラリ化されて 1 つの関数として使えるようになるからです．また，シェイプ拡張を使って長方形以外の外形を表示するには，クリッピング用のマスクが必要ですが，これも簡単に作成できます．Xpm から得られた XImage や Pixmap の表示は Xlib の関数でできます.

ルートウィンドウへの描画

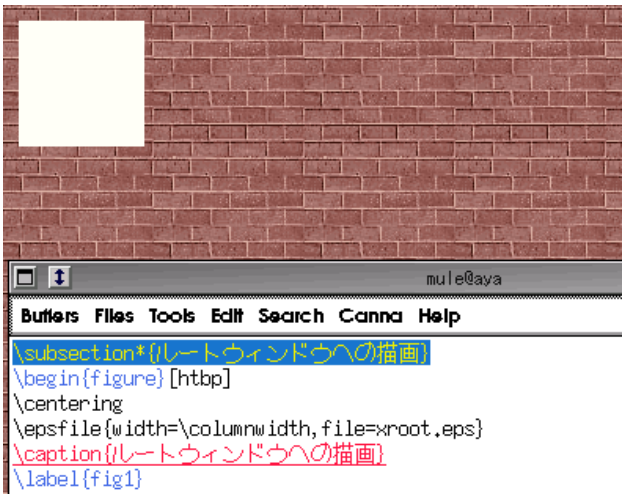

図 1　ルートウィンドウへの描画

X の教科書では，クライアント用のウィンドウを開くところから例題が始まります．が，X では全てがウィンドウであって，たまたま最初から開いているものがルートウィンドウなのです．したがって，そこに直接グラフィック操作をすることがもちろん可能です．ただし，あるウィンドウに描かれたグラフィックを綺麗さっぱり消して，元通りにするには，そのウィンドウを閉ざすだけでよいのに比べて，ルートウィンドウに描かれたグラフィックを元に戻すには，相応の処理を考えなければなりません.

では，教科書的な例題を少し変えてルートウィンドウの左上隅にアイボリー色の長方形を描くプログラムを作成して実行してみましょう.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 #include <X11/Xlib.h>
 5
 6 void main(int argc, char **argv)
 7 {
 8   Display *dpy;
 9   Window root;
10   GC gc;
11   Colormap cmap;
12   XColor fg, exact;
13   int scrn, depth;
14
15   dpy = XOpenDisplay(NULL);
16   scrn = DefaultScreen(dpy);
17   root = RootWindow(dpy, scrn);
18   cmap = DefaultColormap(dpy, scrn);
19   depth = DefaultDepth(dpy, scrn);
20
21   XAllocNamedColor(dpy, cmap, "ivory",
22                            &fg, &exact);
23   gc = XCreateGC(dpy, root, 0, 0);
24   XSetForeground(dpy, gc, fg.pixel);
25
26   while(1){
27     XFillRectangle(dpy,root,gc,10,10,80,80);
28     XFlush(dpy);
29     usleep(500000);
30   }
31 }
```

上記のソースファイルを作成して（名前は xroot.c とでもします），

```
gcc -o xroot xroot.c -lX11
```

とコンパイルしてください．**図 1** に実行例を示します.

イニシャアイズが長いようですが，グラフィックでは普通のことですから驚くには値しません．X はサーバ・クライアント方式なので，クライアントは接続するサーバの情報が必要です．8–19 行はそのためのお題目で，普通そう書くものと思ってください．X ではグラフィックの属性（色や線の太さや種類など）はあらかじめ定義しておき（グラフィックコンテキストと呼びます），描画

命令のときに色や線種を指定するようにはなっていません．10–12,21–23 行ではグラフィックコンテキスト gc の前景色を設定しています．26–30 行の `while(1){...}` 無限ループ中，27 行目 `XFillRectangle()` で長方形を描いています．関数の後半の 4 つの引数 `x,y,w,h` には，長方形の左上隅の位置座標 (`x,y`)（もちろんルートウィンドウに対してです）と幅 (`w`)，高さ (`h`) を指定します．28 行目 `XFlush(dpy)` でサーバーに強制的に描画を行わせます．ほっておくとサーバーへの要求がキャッシュされてしまい，ぎくしゃくします．また，サーバーへの負荷が高くなりすぎないように 29 行目 `usleep()` で，ある程度描画間隔を開けています（これをしないと，マウスも動かな程サーバーが重くなってしまいます）．なお，X を少し詳しく勉強すると，このプログラムはひどく行儀が悪いものであることがわかります．特に，このままでは終了は Ctrl-C しかありません．しかし,『とりあえず』動く最小限のプログラムを示そうとしていることにご理解をください．

雪降る夜の...

暗い画面上で，数多くの小さな雪を上から下に揺らしながら移動させて,『雪降る夜』の情景をアニメーションしてみましょう．ここで問題になるのは，雪の移動です．跡が残らないよう移動前の像は消去しなければなりません．すぐに思いつく簡単な方法は,『全画面を背景色で塗つぶし』てから，新しい位置に各々の雪を描画するというものです．しかし，これはうまくいきません．画面全体を塗りつぶすのに時間がかかり，ちらちらしてしまいます．移動前の位置に背景色で雪を塗りつぶす方法を採用しましょう．

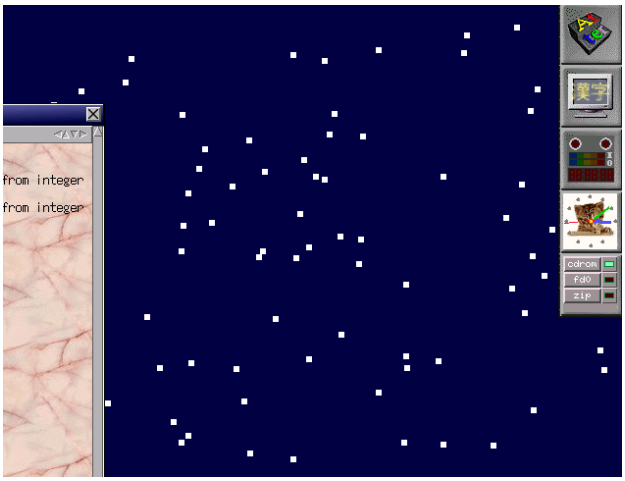

図 2　『雪降る夜の情景』のアニメーション

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 #include <X11/Xlib.h>
 5 #include <X11/vroot.h>
 6
 7 void main(int argc, char **argv)
 8 {
 9   Display *dpy;
10   Window root;
11   GC gc, gcdel;
12   Colormap cmap;
13   XColor bg, fg, exact;
14   int scrn, depth, dpy_w, dpy_h;
15   int k, flk=300;
16   struct{int x; int y;}pos[flk];
17
18   dpy = XOpenDisplay(NULL);
19   scrn = DefaultScreen(dpy);
20   root = RootWindow(dpy, scrn);
21   cmap = DefaultColormap(dpy, scrn);
22   dpy_w = DisplayWidth(dpy, scrn);
23   dpy_h = DisplayHeight(dpy, scrn);
24
25   XAllocNamedColor(dpy, cmap, "snow",
26                                &fg, &exact);
27   XAllocNamedColor(dpy, cmap, "rgb:0/0/50",
28                                &bg, &exact);
29   gc = XCreateGC(dpy, root, 0, 0);
30   gcdel = XCreateGC(dpy, root, 0, 0);
31   XSetForeground(dpy, gc, fg.pixel);
32   XSetForeground(dpy, gcdel, bg.pixel);
33   XSetWindowBackground(dpy, root, bg.pixel);
34   XClearWindow(dpy, root);
35
36   for (k = 0; k < flk; k++) {
37     pos[k].x = rand() % dpy_w;
38     pos[k].y = rand() % dpy_h;
39   }
40   while(1){
41     for (k = 0; k < flk; k++) {
42       pos[k].x += (1 + (rand()%10 - 5));
43       pos[k].y += (1 + (rand()%20));
44       pos[k].x %= dpy_w;
45       pos[k].y %= dpy_h;
46       XFillRectangle(dpy, root, gc,
47                 pos[k].x, pos[k].y, 6, 6);
48     }
49     XFlush(dpy);
50     usleep(100000);
51     for (k = 0; k < flk; k++) {
52       XFillRectangle(dpy, root, gcdel,
53                  pos[k].x, pos[k].y, 6, 6);
54     }
55   }
56 }
```

34 行目 `XClearWindow()` 関数でルートウィンドウをクリアしますが，この時に使われる色はそのウィンドウの背景色でして，33 行目の `XSetWindowBackground()` で設定しています．36–39 行は雪（のつもりの正方形）

の初期位置を画面一杯にランダムに発生させています．40 行目以降の無限ループ（行儀悪）が雪のアニメーションです．42–45 行で移動後の位置決めて，46–47 行で描きます．49 行目は定番の強制描画命令．50 行目で少し待ちます．51–54 行では，背景色と同じ暗い青色で雪一つ一つを塗りつぶして消します．たったこれだけで，ゆらゆら雪が落ちて（y の正方向）いく様子に見えると思いますが，いかがでしょうか．ところで，途中で擬似乱数発生関数 rand() を `srand(`*seed*`)` で初期化せずに用いています（一般には許されない使い方）．こうすると同じ乱数が発生しますから，このアニメーションはいつも同じ情景を再現することになります．

仮想ルートと vroot.h の効用

このプログラムを例えば，

```
gcc -o snowfall snowfall.c -lX11
```

と作成して /usr/loca/bin にでもインストールします．そして，xscreensaver から呼び出すため `~/.Xdefaults` に

```
xscreensaver.programs:  snowfall \n
```

と登録して，xscreensaver を起動し，しばらく待ちましょう．一分経って，始まりましたと，『あれ，真っ黒だ，フリーズか！』．あわててマウスを動かした貴方は，次の瞬間に雪降る夜に描き変えられたルートウィンドウを見るでしょう．冷静に考えればルートウィンドウに直接描画しているのですから，この結果は正しいのです．では，真っ黒な画面は？ 答えは xscreensaver が立ち上げた仮想ルートスクリーンなのです．そこに描画しなければなりません．『えー，そんな』と怒らないでください．xfishtank や xscreensaver に付いていたヘッダファイル **vroot.h をインクルードしてコンパイル**し直しましょう．これだけで，通常は本当のルートに，xscreensaver から呼び出された場合には xscreensaver の仮想ルートに描画するようになります．メデタシメデタシ．したがって，これからは必ず vroot.h をインクルードしましょう．

もっと複雑な動きを描きたい方は，もう少し X の関数（長いけどわかりやすい名前といえます）を覚えて，いろいろ書いてみるしかないでしょう．xscreensaver をコンパイルしたならサブディレクトリ hack にアニメーションモジュールのソースファイルが山程あります．それらはとても良い参考書ですから，読んでみましょう．きっと楽しいアイデアが浮かんできますよ．

透明 XPM 画像のウィンドウへの表示

もう一つの道，Xpm を使ったセル画アニメーションです．初めに，透明部分（Xpm ファイルでは None という色）を持った画像を 1 枚表示して動かしてみます．すなわち，静止画が画面を流れていくというものです．実は，この流れていくところが大事で，ちょっと考えると大変な処理をしなければならないことに気づきます．X のようなオーバーラッピング型のウィンドウシステムでは，ウィンドウによってスクリーンのある領域が覆われることがあります．画像が覆われた場合，その復元は誰が行うのでしょうか？ 実は，イメージを直接ルートに描くクライアントは，クライアント側で復元をする処理を持っている必要があります．これでは，初心者にはお手上げです．

しかし旨い逃げ道があります．**ウィンドウならば，サーバーがある程度面倒をみてくれることになっています**（そうでなければ，クライアントなんて誰も描けないでしょう）．そこで，ウィンドウを開いてそこに一定の間隔で画像を描き続けます．そして，そのウィンドウ全体を動かせばよいのです．ウィンドウがルート上を動いても，その跡が残らないようにサーバーが働いてくれます．というわけで，ソースを御覧ください．

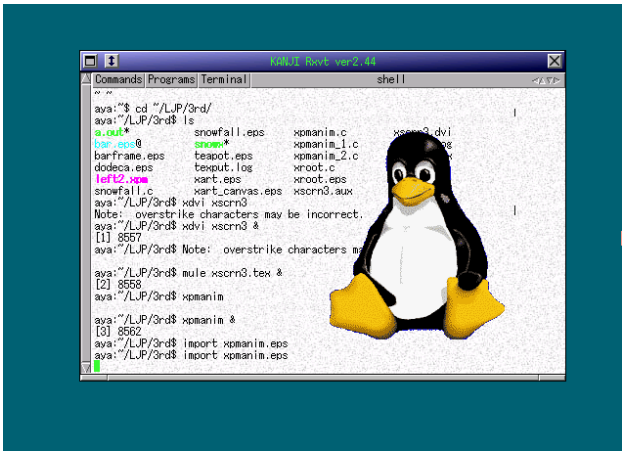

図 3　背景を透明にした Linux ペンギンロゴ

ピックスマップからウィンドウへのコピー

X では **drawable** と呼ばれる描画対象の長方形領域が 2 種類あります．おなじみウィンドウは画面上に現れますが，ピックスマップは画面上に現れません．ウィンドウに直接描画するとタイミングの問題があってちらちらしてしまう場合があります．

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 #include <X11/Xlib.h>
 5 #include <X11/xpm.h>
 6 #include <X11/extensions/shape.h>
 7 #include <X11/vroot.h>
 8
 9 char pixmapfile[] =
10   "/usr/include/X11/pixmaps/xpenguin.xpm";
11 Pixmap icon_pic, icon_msk;
12 XpmAttributes icon_attr;
13
14 void main(int argc, char **argv)
15 {
16   Display *dpy;  Window root, win;
17   GC gc;  Colormap cmap;  XColor bg, exact;
18   int scrn, x, y, dx, dy, sh, sw, ph, pw;
19
20   XSetWindowAttributes attr;
21   unsigned long attrmask;
22
23   dpy = XOpenDisplay(NULL);
24   scrn = DefaultScreen(dpy);
25   root = RootWindow(dpy, scrn);
26   sw = DisplayWidth(dpy, scrn);
27   sh = DisplayHeight(dpy, scrn);
28
29   x = sw/2; y = sh/2; dx = 4; dy = 3;
30
31   icon_attr.valuemask = XpmSize;
32   XpmReadFileToPixmap(dpy, root, pixmapfile,
33          &icon_pic, &icon_msk, &icon_attr);
34   pw = icon_attr.width;
35   ph = icon_attr.height;
36   win = XCreateSimpleWindow(dpy,root, x, y,
37                             pw, ph, 0, 0, 0);
38   attr.override_redirect = True;
39   attr.save_under = True;
40   attrmask = CWOverrideRedirect|CWSaveUnder;
41   XChangeWindowAttributes(dpy, win,
42                           attrmask, &attr);
43   XShapeCombineMask(dpy, win, ShapeBounding,
44                     0, 0, icon_msk, ShapeSet);
45   gc = XCreateGC(dpy, win, 0, 0);
46   XMapWindow(dpy, win);
47
48   while(1){
49     XCopyArea(dpy, icon_pic, win, gc, 0, 0,
50                                pw, ph, 0, 0);
51     XFlush(dpy);
52     usleep(200000);
53     x += dx;
54     if( x < 0 || x > sw-pw) dx *= -1;
55     y += dy;
56     if( y < 0 || y > sh-ph) dy *= -1;
57     XMoveWindow(dpy, win, x, y);
58   }
59 }
```

画像をさっと切替えるには，ピックスマップに予め画像を準備して，いっきにウィンドウにコピーするという方法が一般的です．

49 行目 XCopyArea(dpy，src，dest，gc，src_x，src_y，width，height，dest_x，dest_y) でコピーを行っています．src に含まれる幅 width 高さ height の長方形領域を dest にコピーします．*_x，*_y は長方形の左上の頂点座標です．この場合にはピックスマップ icon_pic からウィンドウ win へコピーしてますが，大きさが等しい（幅 pw, 高さ ph）ので，コピー前後の左上の頂点は (0,0) となっています．

32,33 行の XpmReadFileToPixmap（）は Xpm 画像ファイルを読み込んで，絵自身とマスク（None があれば）を ピックスマップに張り込む関数です．4,5 番目に引数にピックスマップのポインタを与えます（書き込みをしますから当然ですね）．34,35 行では，属性 icon_attr から，画像の大きさを読みだしています．

サーバーにおまかせの設定

36,37 行の XCreateSimpleWindow() でウィンドウを生成します．最後の 3 つの引数は，ウィンドウ境界の幅・色およびウィンドウの背景色を与えますが，ここではそれぞれ 0 にしてしまいました．38-42 行で生成されたウィンドウの属性を変えています．属性の設定値と，どの属性をかえるかのビットマスクを立てて XChangeWindowAttributes()（そのままの名前でわかりやすいですが，長いですね）を呼びだします．override_redirect はウィンドウマネージャーの介在を無効にします．したがって，クライアントが指定した通りの位置や大きさで表示されます（タイトルバーなどの装飾もなくなります）．save_under は指定したウィンドウの下に隠れた他のウィンドウの内容をサーバーにセーブさせます．したがって，指定したウィンドウが動いて隠れていた領域が露出されたとき自動的に元の画面が復帰します．以上の説明のように，**画面の復帰という面倒な処理をサーバーに負わせる**ことができるわけです．

型取りはシェイプ拡張で

絵とマスクを組み合わせて，型取られた画像を合成するには Xpm ライブラリではなく，X11R4 からサポートされた**シェイプ拡張**を利用します．43,44 行にある XShapeCombineMask() でマスクを目的のウィンドウ（ピックスマップは不可）にセットしておくと，その後は，マスクされた部分のみ表示され，残りは透明となります．引数の 6 番目がマスク用のビットマップ

(白黒のピックスマップ)，4,5 番目がマスクをかぶせるときのウィンドウでの左上の頂点座標です．

2 つのピックスマップ（絵とマスク）を用意すればシェイプ拡張を利用できます（Oclock や Xteddy など数多くのアプリケーションが思い浮かびます）．しかし，Xpm ライブラリを使えば，ぴったりしたマスクを”None” を利用して **1 つの Xpm 画像ファイルから簡単に作成できますから，とても便利です**．Xpm ライブラリでは，ピックスマップ・ファイル・インクルードされた配列・バッファの間で相互にデータを変換する関数が用意されています．また，XImage に変換する関数群（実は，これを利用してピックスマップへ読み書きしているので，こちらの方が下位関数）もあります．

複数のセル画の切替え

さて，いよいよ今度こそセル画によるアニメーションです．しかし，もうお気づきと思いますが，なんのことはない，複数の Pixmap を用意してセル画を書き込んでおき，表示用のウィンドウに,『`XShapeCombineMask()` でマスクをセットし `XCopyArea()` で絵をコピーする』を次々と実行するだけのことです．他のウィンドウと重なったときなどの面倒な処理は一切サーバー任せ という，お気楽な構想です (^^;．凝った画像を準備したり，ウィンドウを動かしてそれらしく見せたりする部分に集中しましょう．とは言っても，サーバーへの負荷が高いのであまり多くのセル画や大きいセル画を使うことはできませんので，ご注意ください．

図 4　oneko の tora を拡大して色付けした『虎猫』

例として，かの有名な oneko のビットマップ画像を convert で 2 倍に拡大して，以下に紹介する Xart で色づけしたものを使って，画面を駆け回る『虎猫』を作ってみました（**図 4**）．ただし，2 つしかフレームがないので滑らかではありません．

XPaint からの発展 XArt

今回は，プログラムのソースとその説明ばかりで，書いている筆者も息苦しくなってきました．そこで，この連載のモットーである『楽しく Linux を使ましょう』の精神に従って,『虎猫』の画像を作ったペイントツールの紹介をして筆をおきたいと思います．

ペイント系ツールでは，『PhotoShop キラー』とまで称される GIMP のバージョン 1.0 がついに正式リリースさました．おかげで，ちょっとしたお絵書きに使っていた XPaint はもう廃れゆく運命かと思っていましたところ，なんと Rick Hohensee (`http://cqi.com/~humbubba/linux.html`) 氏によって XArt なるペイントツールに生まれ代わって登場しました.『にじみ』などのフィルターも備わり，確かに芸術的な仕事ができそうです．GIMP のマルチレイヤーなどがない分，機能は制限されますが，操作は直観的でシンプルです．

図 5　ペイントツール Xart のツールボックス

では，Xart の紹介を兼ねて oneko のモノクロ虎猫から天然色虎猫への変身の過程を少し説明します．まず，oneko.tar.gz を展開し，サブディレクトリ bitmaps/tora/ にある，左に移動するときの絵 left1_tora.xbm と left2_tora.xbm を 64x64 の Xpm 画像に convert で変換します．

```
convert -colors 2 -geometry 64x64 \
                        left1_tora.xbm left1.xpm
```

色は 2 色にしてください．そうしないと，外形がぼんやりしてしまいます．

図 6　oneko の tora を拡大して得た Xpm 画像

続いて Xart で left1_tora.xpm を開いて，加工しましょう．最初に胴体に黄色を流し込みます．図 5 のツールメニューでは，下から 2 段目左から 3 列目のバケツ?を選択しましょう（その右も流し込みですが，グラデーションがかかります）．キャンバスの下部には，模様や色の選択ボタンがありますから，黄色のボタンを押せばいいです．もちろん，Add Solid ボタンを押して，任意の色を追加することもできます．あとで透明にする背景地には，青を流し込みましょう（直接 None を設定できないので）．**図 7** の左図のようになります．これだけでは，単色のビットマップと変りませんから，縞模様を滲ませます．ツールメニューでは，最上段左から 3 列目のボタンをマウス左ボタンでクリックします．デフォルトのブラシは大きすぎるので，マウス右ボタンでクリックして，一番小さいブラシを選択しておきましょう．このブラシで縞模様をなぞると，黄色と黒が滲んで，**図 7** の右図のようにまあまあそれらしくなるという寸法です．

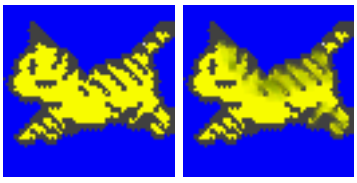

図 7　色を塗って，にじみをかける

さて，キャンバス上部のメニューバーの File メニューから Save を実行することをお忘れなく．最後の仕上げは，Xpm ファイルを mule で編集して，色を blue（あるいは #0000ff）から None に書き換えます．晴れて，透明指定”None”を含む Xpm 画像のできあがりです．

次回は

はー苦しかった．やはり X11 のプログラムの解説は筆者にはちょっと荷が重くて疲れます．ただそれでも，『そんなに難しくはないんだ』という感触だけは持っていただけたでしょうか? このような初歩的解説を読んで『これならできそうだ』と意欲を持たれた方は，冒頭の OpenGL などの 3d や 動画を扱うライブラリに挑戦してみてはいかがでしょう．楽しめると思いますよ，なにせ趣味のプログラミングですから．

それで次回は，様々な**時計**を紹介したいと思います．デスクトップを飾るアクセサリーとしては，実用的な意味から考えても一番重要な小物でしょうが，まとまった記事をみたことがなかったので，敢えて取り上げます．種類と使い方の紹介で難しい話は一切なし，図面盛り沢山の楽しいお話の予定です．