計算機ソフトウェア＃4

久野　靖*

1992.5.20

本日の内容

本日はも例によって、まず野須さんの論文紹介からお願いします。椚田さんがポストしてらっしゃったけど、順番は岡野さん→野須さん→原田さん→三橋さん→中谷さん→椚田さん、でよろしいですか。

なお、豊島さんは今学期は忙しいので、Waiver 申請して、紹介はパス、その代わり夏休みですごいプログラムを作ってくださる:-P そうですから、期待しましょう。6/17 以降は論文紹介に代わる内容を準備しましょう。

後半のデモですが、今日はまず PostScript のお話 (そうしないと色々な絵が描けない)、その後それを利用した例題に行きます。本日もミニアンケートを (あとで) 出しますので、2・3 日中に答を投稿してください。

1　PostScript について

既にちょっと述べましたが、NeXT Step のウィンドウサーバは Display PostScript (DPS) を基本としています。さて、PostScript とは何でしょう? 皆様、PS についてはどのくらいご存じですか?

- 図形を記述するような言語である。
- 解釈実行型の処理系を前提としている。
- スタック方式の言語である。
- 手続きが定義できる。
- 出力装置の解像度などに依存しない。

PS で図を描いて試す場合には、Sun では gs というのを使ってください。

```
% gs
Initiailzing...
...
GS> 100 100 moveto
GS> newpath
GS> 100 100 rmoveto 100 -100 rmoveto
GS> closepath stroke
GS>^D
%
```

*筑波大学大学院経営システム科学専攻

さて、NeXT のウィンドウサーバにも同様にして PS コードを送ると絵が描ける、という風にして画面を全部描いているのですが、C から PS コードを送るのに毎回文字列を生成して送るのも面倒なので、次のような関数が一式用意してあります。

```
PSnewpath(); PSclosepath(); PSstroke(); PSfill();
PSmoveto(x, y); PSrlineto(); PSarc(x, y, r, d1, d2); ...
```

つまり、だいたい使いたいような描画機能は PS なんとか、で呼べばいいわけです。その前に、うまく描けるかどうかは gs などで確認しておく方がいいでしょう。なお、これらの関数を使いたい時にはかならずファイルの先頭に

```
#import <dpsclient/psops.h>
```

を入れておくこと。私はこれで 1 日潰しましたので。

2 円を描く

実はこれまで使ってきた NXFrameRect とかは「窓の枠を描く」などのためにあったのを流用していたわけですが、今度は上述の PS なんとか、で好きな形が描けるようになります。で、まずは円を描く例。main は昔作った何の機能もない奴に戻っています。

```
#import <appkit/appkit.h>
#import "TestView.h"

main() {
    id myWin, myPanel, myMenu, myView;
    NXRect aRect;
    [Application new];
    NXSetRect(&aRect, 100.0, 300.0, 300.0, 300.0);
    myWin = [Window newContent: &aRect
                style: NX_TITLEDSTYLE
                backing: NX_BUFFERED
                buttonMask: NX_MINIATURIZEBUTTONMASK
                defer: NO];
    [myWin setTitle: "CompSoftDemo 3.0"];
    NXSetRect(&aRect, 0.0, 0.0, 300.0, 300.0);
    myView = [TestView newFrame: &aRect];
    [[myWin contentView] addSubview: myView];
    NXSetRect(&aRect, 100.0, 700.0, 300.0, 40.0);
    myPanel = [Panel newContent: &aRect
                  style: NX_TITLEDSTYLE
                  backing: NX_BUFFERED
                  buttonMask: NX_CLOSEBUTTONMASK
                  defer: YES];
    [myPanel setTitle: "About CompSoftDemo"];
    [myPanel removeFromEventMask: (NX_KEYDOWNMASK|NX_KEYUPMASK)];
    myMenu = [Menu newTitle: "CS Demo"];
    [[myMenu addItem: "Info..."
          action: @selector(orderFront:)
          keyEquivalent: 'i'] setTarget: myPanel];
    [myMenu addItem: "Hide" action: @selector(hide:) keyEquivalent: 'h'];
    [myMenu addItem: "Quit" action: @selector(terminate:) keyEquivalent: 'q'];
    [myMenu sizeToFit];
    [NXApp setMainMenu: myMenu];
    [myWin display];
    [myWin orderFront: nil];
    [myWin makeKeyWindow];
```

```
    [NXApp run];
    [NXApp free];
}
```

次に TestView.h もやっぱり何の機能もない奴に戻っています。

```
#import <appkit/View.h>

@interface TestView : View {
}
+ newFrame: (const NXRect*)frameRect;
- drawSelf: (const NXRect*)rects : (int)rectCount;
@end
```

最後に TestView.m ですが、この中の drawSelf で円を描いているわけですね。

```
#import "TestView.h"
#import <appkit/Control.h>
#import <dpsclient/psops.h>

@implementation TestView
+ newFrame: (const NXRect*)frameRect {
      self = [super newFrame: frameRect];
      return self;
}
- drawSelf: (const NXRect*)rects :(int)rectCount {
      NXEraseRect(&bounds);
      PSnewpath();
      PSarc(100.0, 100.0, 50.0, 0.0, 360.0);
      PSsetgray(0.0);
      PSstroke();
}
@end
```

さて、ここで質問ですが、円を 3 つとか 4 つ描くにはどうしたらいいでしょうか? 「普通の」「素直な」やり方には色々問題があります。それはどんな問題でしょうか?

3 「図形」の部品オブジェクト化

というわけで、今度は TestView の中で直接円を描くのではなく、「円」というオブジェクトを作って、それを TestView に張り付ける、というやり方にしてみましょう。次のような感じですね。

```
#import <appkit/appkit.h>
#import "TestView.h"
#import "Circle.h"

main() {
    id myWin, myPanel, myMenu, myView;
    id cir1, cir2, cir3;
    NXRect aRect;
    [Application new];
    NXSetRect(&aRect, 100.0, 300.0, 300.0, 300.0);
    myWin = [Window newContent: &aRect
                style: NX_TITLEDSTYLE
                backing: NX_BUFFERED
                buttonMask: NX_MINIATURIZEBUTTONMASK
                defer: NO];
    [myWin setTitle: "CompSoftDemo 3.1"];
```

```
    NXSetRect(&aRect, 0.0, 0.0, 300.0, 300.0);
    myView = [TestView newFrame: &aRect];
    [[myWin contentView] addSubview: myView];
    cir1 = [[Circle newRadius: 80.0] setPosX: 100.0 Y: 100.0];
    [myView addOne: cir1];
    cir2 = [[Circle newRadius: 40.0] setPosX: 200.0 Y: 50.0];
    [myView addOne: cir2];
    cir3 = [[Circle newRadius: 100.0] setPosX: 150.0 Y: 200.0];
    [myView addOne: cir3];
    NXSetRect(&aRect, 100.0, 700.0, 300.0, 40.0);
    myPanel = [Panel newContent: &aRect
                  style: NX_TITLEDSTYLE
                  backing: NX_BUFFERED
                  buttonMask: NX_CLOSEBUTTONMASK
                  defer: YES];
    [myPanel setTitle: "About CompSoftDemo"];
    [myPanel removeFromEventMask: (NX_KEYDOWNMASK|NX_KEYUPMASK)];
    myMenu = [Menu newTitle: "CS Demo"];
    [[myMenu addItem: "Info..."
          action: @selector(orderFront:)
          keyEquivalent: 'i'] setTarget: myPanel];
    [myMenu addItem: "Hide" action: @selector(hide:) keyEquivalent: 'h'];
    [myMenu addItem: "Quit" action: @selector(terminate:) keyEquivalent: 'q'];
    [myMenu sizeToFit];
    [NXApp setMainMenu: myMenu];
    [myWin display];
    [myWin orderFront: nil];
    [myWin makeKeyWindow];
    [NXApp run];
    [NXApp free];
}
```

で、この Circle というのはどうなっているかというと。

```
#import <appkit/appkit.h>

@interface Circle : Object {
    float x, y, r;
    id view;
}
+ newRadius: (const float)radius;
- setPosX: (const float)xpos Y: (const float)ypos;
- drawIt;
@end


#import "Circle.h"
#import <dpsclient/psops.h>

@implementation Circle
+ newRadius: (const float)radius {
      self = [super new];
      r = radius;
      x = y = 0.0;
      return self;
}
- setPosX: (const float)xpos Y: (const float) ypos {
      x = xpos;
      y = ypos;
      return self;
```

```
}
- drawIt {
      PSnewpath();
      PSarc(x, y, r, 0.0, 360.0);
      PSsetgray(0.0);
      PSstroke();
}
@end
```

簡単ですね? では、これを「張り付ける」ように直した TestView の方も。

```
#import <appkit/View.h>
#import "Circle.h"

@interface TestView : View {
      id objs[100];
      int count;
}
+ newFrame: (const NXRect*)frameRect;
- drawSelf: (const NXRect*)rects : (int)rectCount;
- addOne: (id)anObj;
@end


#import "TestView.h"
#import "Circle.h"
#import <dpsclient/psops.h>

@implementation TestView
+ newFrame: (const NXRect*)frameRect {
      self = [super newFrame: frameRect];
      count = 0;
      return self;
}
- addOne: (id)anObj {
      objs[count++] = anObj;
}
- drawSelf: (const NXRect*)rects :(int)rectCount {
      int i;
      NXEraseRect(&bounds);
      for(i = 0; i < count; ++i)
          [objs[i] drawIt];
}
@end
```

以上なわけです。さて、練習問題です。

練習 1 円以外にもいくつか図形のクラスを作って、張り付けてみよ。

練習 2 またまた、ボタンなどで動かせるようにしてみよ。

練習 1 は簡単ですから、ぜひともやってみてください。これが簡単でものたりない人は練習 2 ですが、これはどうデザインするかが腕の見せどころですね。ではがんばって。