本稿は Linux Japan 誌 2002 年 04 月号に掲載された記事に補筆修正したものです.

Tclet

遠隔講義や e-learnig など，大学でもインターネットを利用した教育への試みは盛んです．また，それをある程度使いこなせないようでは大学淘汰の時代に生き残れないと言われています．私も物理の教員の端くれですから，いろいろ Linux を使って楽しく試行錯誤しています．今のところ何でもブラウザーで表示させるという考え方には否定的ですが，実際にはバイナリ配布というわけにもいかず（危険ですからね），ソースあるいはスクリプトを配布して安全を確認してもらって，クライアント側でコンパイル即実行が良いと思っています．まあ，しかし安全であることが保証されればスクリプト言語を用いるのが一般的でしょう．クライアント側のスクリプト言語といえば，まずは JAVA Script が思い浮かびます．しかし，JAVA と言っておきながら JAVA とは異なる言語 JAVA Script を使う気になりません．また，筆者の天の邪鬼な性格からか，市場で一番などという理由は最も忌み嫌うところです．そこで，前回の続きで筆者が好んで用いる GUI 構築ツール Tcl/Tk の Netscape へのプラグインである Tclet[1]W[3] を紹介したいと思います．Tclet はほとんど Tcl/Tk ですが，**セキュリティを考慮して，ファイル操作などの関数が使えないようになっています**．間違ってあるいは悪意を持って書こうにも，元々悪さができないように考えられています．だからといって MS-Windows で IE なんかを使っては元も子もありませんから，まだましな Netscape のプラグインというのは全うな選択です．もちろん Linux では問題なしです．

インストール

前回の Expect と同様に，使えるようにするには多少の労力を要します．もっとも既にホストにある Tcl/Tk とは別個にインストールできるので，依存性の問題が発生しないぶん楽です．配布サイトからバイナリ

```
tclplug20-x86-linux-glibc.tar.gz
```

を持ってきて展開し，install.sh を実行するだけで済みます．RedHat5.x 用となってますが，glibc2.2 でも動いてます． ただし，日本語を表示させるとなるをパッチを当ててコンパイルしなければなりません [2]W[3]．なお，NASA で公開されている LHEATcl Plugin もあります [3]W[3]．両方をインストールしても大丈夫ですから，比べてみるのも面白いかもしれません（こちらは日本語化パッチはないようです）.

HTML 内での記述

Tclet は Tcl/Tk を知っていれば新しく覚える事柄はほとんどありません．HTML では他のスクリプトを使う場合と同様に

```
<embed src=***.tcl width=620  height=400>
```

の様に埋めこみます．これで設定してあれば Tclet が呼ばれてブラウザー内部に表示されます．もちろん，拡張子 tcl がついたファイルの処理を Tcl/Tk に任せる（wish など）ように設定することもできますが，これは生の Tcl/Tk がファイルを操作するあるいはシステム関数を呼び出せるなどの機能を持ってますから，大変危険です．

Tcl/Tk のおさらい

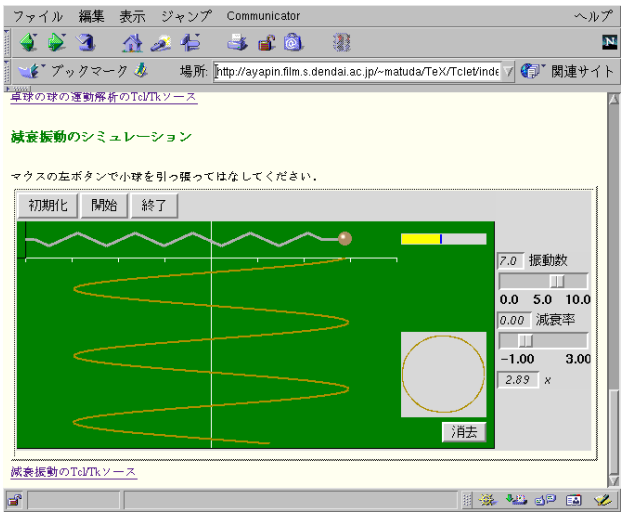

図 1　Tclet で作成した減衰振動のシミュレーション．

とうことで，あとは Tcl/Tk で書いていけば良いことになります．図 1 に減衰振動のシミュレーションの例を示します．これがたった 300 行で済みます．運動方程式（2 階の常微分方程式）を解く中心部分には，以下に示すように 4 次の Runge-Kutta 法を使っているので精度も C 言語に遜色ないはずです．

```
proc RungeKutta {x vx dx dvx} {
  global H
  upvar $dx sdx $dvx sdvx

  set kx1 [ expr $H*$vx]
  set jx1 [ expr $H*[Fdvx $x $vx] ]
  set  x1 [ expr  $x + 0.5*$kx1 ]
  set vx1 [ expr $vx + 0.5*$jx1 ]
  set kx2 [ expr $H*$vx1]
  set jx2 [ expr $H*[Fdvx $x1 $vx1] ]
  set  x2 [ expr  $x + 0.5*$kx2 ]
  set vx2 [ expr $vx + 0.5*$jx2 ]
  set kx3 [ expr $H*$vx2 ]
  set jx3 [ expr $H*[Fdvx $x2 $vx2] ]
  set  x3 [ expr  $x + $kx3 ]
  set vx3 [ expr $vx + $jx3 ]
  set kx4 [ expr $H*$vx3 ]
  set jx4 [ expr $H*[Fdvx $x3 $vx3]]

  set sdx  [ expr ($kx1 + 2*$kx2 + 2*$kx3 + $kx4)/6.0 ]
  set sdvx [ expr ($jx1 + 2*$jx2 + 2*$jx3 + $jx4)/6.0 ]
}
```

この部分について，ちょっと補足します．値を戻すために `upvar` による参照渡しをわざと使ってみました．Tcl では C 言語と同様に，プロシージャ内部で宣言された変数は局所変数です．プロシージャの引数になっている変数は値渡しなので，書き換えができません．そこで参照渡しをする変数は，プロシージャ側で `upvar` によって参照渡しとして使うことを宣言しなければなりまん．またこれに対応して参照渡しする変数には，次のように $ をつけないで呼び出します．

```
RungeKutta $x $vx dx dvx
```

このような記述は複雑で混乱するかもしれません，そこで `dx`, `dvx` を大域変数にしてしまいたい誘惑にかられます．大域変数の多用は悪いソースの見本と言われますが，まあ，300 行程度のものであれば十分に管理できるので，効率的ですし，そんなに悪くはないんじゃ

ないかなと考えています．Tcl/Tk がシミュレーションに向かないと感じられる一番の原因は速度が遅いということです．この点に関してはバージョン 8.0 以降随分とし改善されています [4]W[3]．むしろ本当のところは，この upvar による参照渡しの例のように，変数 x はそのままでは C 言語でのポインタであって，値は $x として取り出さなければならない．一々“set 変数 [expr 式]”と記述しなければならないという手間にあるのではないかと思っています．文法の問題は，確かに最初は違和感を覚えますが．なれれば大丈夫と思います．また，RungeKutta に限っていえば，このような汎用のプロシージャは一度書いてしまえば済むのでたいした手間にはなりません．むしろ，ボタン，スケール，スクロールバー等の GUI 部品（ウィジェット）が簡単に構築できることの利点が優ります．

そこで，図 1 に現れている GUI 部品がいかに簡単に実現できるかについておさらいしましょう．まず左上のボタン類ですが，


```
button .f1.f1.b0 -text "初期化" -command Init
button .f1.f1.b1 -text "中断" -command StartStop
button .f1.f1.b2 -text "終了" -command "exit"
```

と，最初に button ウィジェットの宣言，生成されるオブジェクト固有の名前の宣言，そして必要なオプション，この場合ラベルと実行させるコマンド（あるいはプロシージャ）を定義するだけです．名前は階層を守らなければならないので，ボタンをレイアウトする台紙としてフレームを次のように定義しておく必要があります．

```
frame .f1
frame .f1.f1
frame .f1.f2
```

上の二つの例をまとめると，Tcl/Tk での部品の宣言は

```
ウィジェット名　階層的な固有名　オプション
```

という単純なものであるということです．

続いて右側にあるスケールには，以下のようにオプション指定が随分としてあります．


```
scale .f2.s2 -orient horizontal -showvalue no -length 100\
      -from -1.0 -to 3.01 -tickinterval 4.0  -variable B \
      -resolution 0.01
```

しかし，その指定方法は単純で -オプション名 値 の組み合わせとなっており，直感的に理解できます．また，これらのオプションのほとんどが後で動的に変更できる点も見逃せない長所です．ただし，オプション名を覚える必要はあります．

バネ振動子やその変位を曲線表示する大変重要な部品が canvas です．幅と高さのオプション指定は最低限必要ですね．

```
canvas .f1.c0 -width $CWIDTH -height $CHEIGHT \
       -background $bgcolor
```

canvas には通常の部品に加えて，図形要素（line，circle，bitmap 等）を置くことができます．しかしフレームのような pack コマンドによる部品の自動レイアウトが効きません．全てプログラマが位置と大きさを指定しなければなりません．例えば右下の消去ボタンは

```
button .f1.c0.b1 -text "Clear" -width 3 -height 1 -pady 0\
       -highlightthickness 0 -command ClearPhase
...
.f1.c0 create window [expr $CWIDTH-8] \
       [expr $HARMY+$EWIDTH+87] -window .f1.c0.b1 \
       -anchor ne
```

と canvas の中に大きさと位置を指定した窓を作成して貼りつけなければなりません．最近のバージョンでは gif 画像を扱うこともできるようになりました．この例では，バネの先のおもりが gif 画像です．base64 でエンコードしたデータを一端変数にセットして，それをもとに image オブジェクトを生成します．

```
set ball_data "
R0lGODlhEAAQAPMAAAAAAKyNZMGecc6pet...
...
"
image create photo ball_img -data $ball_data
```

アニメーションを行うには，canvas に描いた線やボールといったオブジェクトをそれぞれ独立に消去しなければなりません．canvas に描かれたオブジェクトは通常のウィジェットのような階層的な固有名がつきません．そのかわりオプションでタグ名をつけられます．このタグ名に対して消去コマンドを実行することになります．例えば，以下は振動子を描画するルーチンの一部ですが，

```
...
.f1.c0 delete ball line
DrawSpring .f1.c0 $OFFSET [expr $ix] $YC 25 7
.f1.c0 create image [expr $ix] $YC -image ball_img \
       -tags ball
...
```

タグ名 ball，line のオブジェクトを消してから，新しい位置に描いています．タグ名 line は複数のオブジェクトの共通のタグ名となっていて，DrawSpring 中のバネの折れ線をまとめて扱っています．

さて，このシミュレーションでは，運動を開始させるため，バネの先についているボール（オブジェクト .f1.c0）をつかんではなすようになっています．

このような操作を実現するには，マウスのイベント処理が必要になります．Tcl/Tk では **bind** コマンドによりマウスイベントに対する応答を細かく制御できます．今回の例では

```
bind .f1.c0  <B1-Motion> { moveball %x }
bind .f1.c0  <ButtonRelease> { DrawIt $holdx $holdvx }
```

となっており，実質のプロシージャ moveall，DrawIt はむろん他で定義しなければなりませんが，それをマウスイベントで呼び出す手続きが明確かつ簡素です．

まとめ

Tcl/Tk を使えるならばその知識で Tclet を書くことができます．したがって新たに JAVA Script を覚えずとも，ブラウザーで動く簡単なシミュレーションを作成できることがご理解いただけましたでしょうか．新しい言語やライブラリが機能満載で毎日のように登場してきますが，Tcl/Tk のようなほどほどの機能でどんな場合にも使えるというものは少ないです．

参考文献

[1] Tclet のダウンロードページ. W3
http://www.scriptics.com/software/plugin/
次のサイトの方が文字が大きくて読み易いです. W3
http://www.demailly.com/tcl/plugin/

[2] SRA 社の FTP サイト W3.
ftp://ftp.sra.co.jp/pub/lang/tcl

[3] NASA の HEASARC（High Energy Astrophysics Science Archive Research Center）で公開されている Tcl plugin. W3
http://heasarc.gsfc.nasa.gov/Tools/maki/plugin/

[4] Tcl/Tk のグラフィック描画速度を測る TkEngine の配布ページ. W3
http://www.interq.or.jp/japan/s-imai/tcltk
/tkengine.html