JAPLA シンポジウム資料 2006/12/9

Ｊのウィンドウズ・プログラミングとそのグラフィックス入門

－微分方程式グラフィックスに向けて－

西川 利男

０．はじめに

現在、ユーザにとってパソコンはその使用する目的によってさまざまに使われる。データ処理を目的とするときでも、エンド・ユーザ向きとプレゼンテーションのためとでは、その内容、性格がかなり異なっている。

Ｊの場合では次のようになると考えられる。

	エンド・ユーザ向き	プレゼンテーション向き
一般処理	コマンドレベルでの実行	マウス・クリックによる実行
グラフィックス	高レベルグラフィックス	低レベルグラフィックス

ここで、グラフィックスの高レベル、低レベルとは前回の例会で筆者が名づけたものでそれぞれ以下のものを指すことにする。

・高レベルグラフィックス … plot, gdgraph

・低レベルグラフィックス … isigraph, gl2, gl3

最近、プログラミングを行う目的は、単に自分の個人的使用（Personal Use）として、計算など処理をしてその結果を見れば良い、というだけではなくなって来ている。処理結果をいかに見栄えよく見せるか、というプレゼンテーション（Presentation）の道具のためとなっている。つまり、Ｊに限らず、Visual Basic, C++などでもそうだが、ウィンドウズ・レベルのプログラムをどう作成するかが求められている。

Ｊではいずれの目的にも極めて有用である。ウィンドウズ・プログラミングでは画面上の種々のボタン、入力窓などいわゆるフォームの設計、作成のプログラミングが必要であるが、従来からの古くからのプログラマはこの部分が苦手という人も少なくない。

Ｊのチュートリアルとして、エンド・ユーザ向きのものは鈴木義一郎氏、志村正人氏をはじめとして何人もの方のすぐれた解説がある。今回、筆者はＪのウィンドウズ・プログラミングとその上でのグラフィックスに焦点を当て、先々月[1],[2]から発表している微分方程式グラフィックスを例として解説したいと考えている。

[1] 西川利男「Ｊによる微分方程式のグラフィック・アプローチーその１
１階常微分方程式－数値解と方向場表示の活用」Ｊ研究会資料　2006/10/28

［2］西川利男,中野嘉弘「Ｊによる微分方程式のグラフィック・アプローチーその１・続き
Ｊのバージョンとウィンドウズ・グラフィックス」Ｊ研究会資料　2006/11/25

１．簡単な（コマンド・レベル）のプログラムと実行

Ｊが起動すると、最初のウィンドウは実行ウィンドウであり、キー入力したコマンドは直ちに実行される。しかし、プログラムの作成はここではなく、新しく開いたプログラム作成ウィンドウ（これをスクリプトウィンドウと呼ぶ）で行う。それには初めの実行ウィンドウのメニューバーから[File]-[New IJS]をクリックして始める。

Ｊのプログラムは次のように２種類の形式で作る。

まず以下のような一行のプログラムは tacit form と呼ばれる。

```
square =: *:
```

初心者は次の explicit form で作ることをお勧めする。

```
sum =: 3 : 0
+/ y.
)
```

このスクリプトファイルを仮に ntest.ijs という名前で保存し、CTRL-w により実行できる形にする。（これを普通コンパイルと呼ぶ）そして CTRL-TAB により実行ウィンドウに切り替える。

この実行ウィンドウ上で

```
names ''
```

と打ってみると、square, sum との２つのプログラムが出来ている。そして、それぞれ以下のように実行する。

```
square 2 3 4
 4 9 16
sum 2 3 4
 9
```

あるいは

```
DA0 =: 2 3 4
DA1 =: square DA0
DA2 =: sum DA1
```

とした上で、DA2 と打てば、結果は 29 (= 4 + 9 + 16)と返される。

これらの数値をグラフ化するのも簡単である。

```
load 'plot'
plot DA1
```

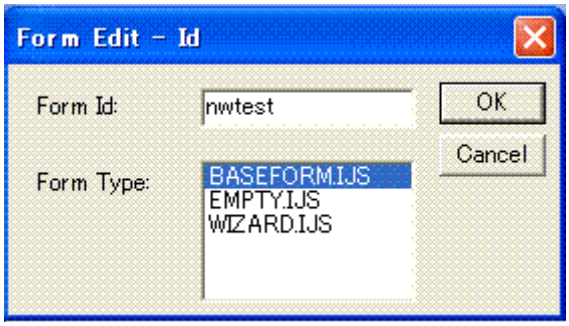
とすれば、DA1 の数値のグラフが座標目盛付きで表示される。

プログラムの終了は実行ウィンドウの[File]-[Exit]により行う。次回のスクリプト・ファイルのロードは[File]-[Open]から ntest.ijs を選んで行えばよい。

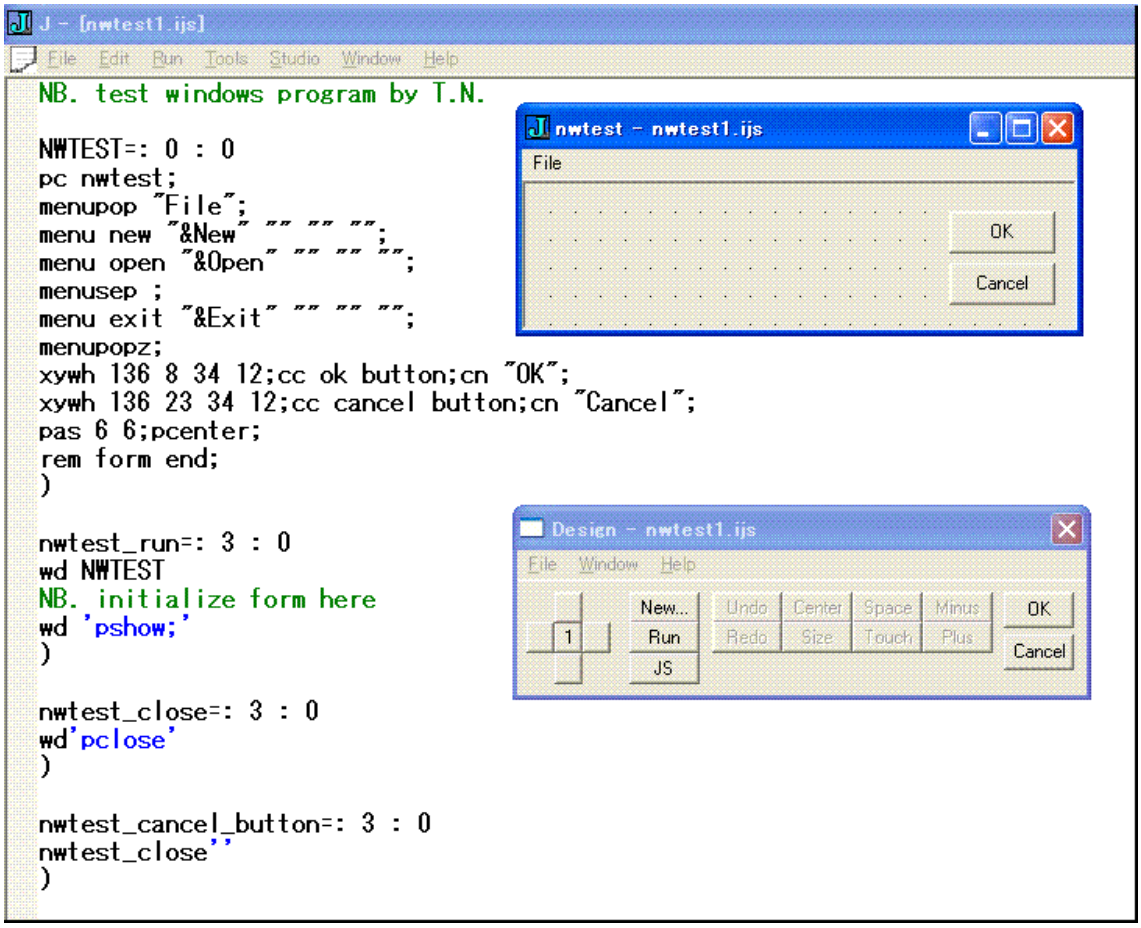
つまり、Ｊのプログラミングでは、実行ファイル(***.ijx)とスクリプト・ファイル(***.ijs)との間を行ったり来たりしてプログラム作成が行われる。

２．フォーム・エディタとウィンドウズ・プログラムの作成

Ｊのウィンドウズ・プログラムは通常フォーム・エディタにより行う。先のスクリプトファイル上で、[Edit]-[Form Edit]をクリックする。すると図のようなフォーム作成画面が現れる。フォーム名をたとえばnwtestとしOKをクリックしてフォーム作成を始める。



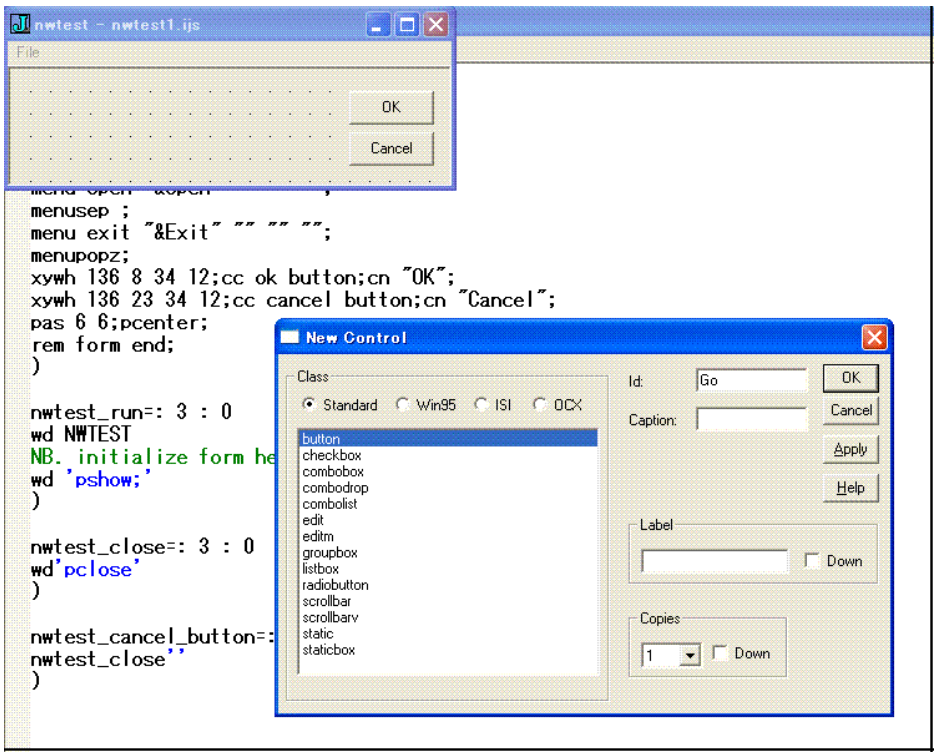
すると、次のようにデフォルトとしてOKとCancelの２つのボタンを持った基本となるフォームが自動的に作られる。



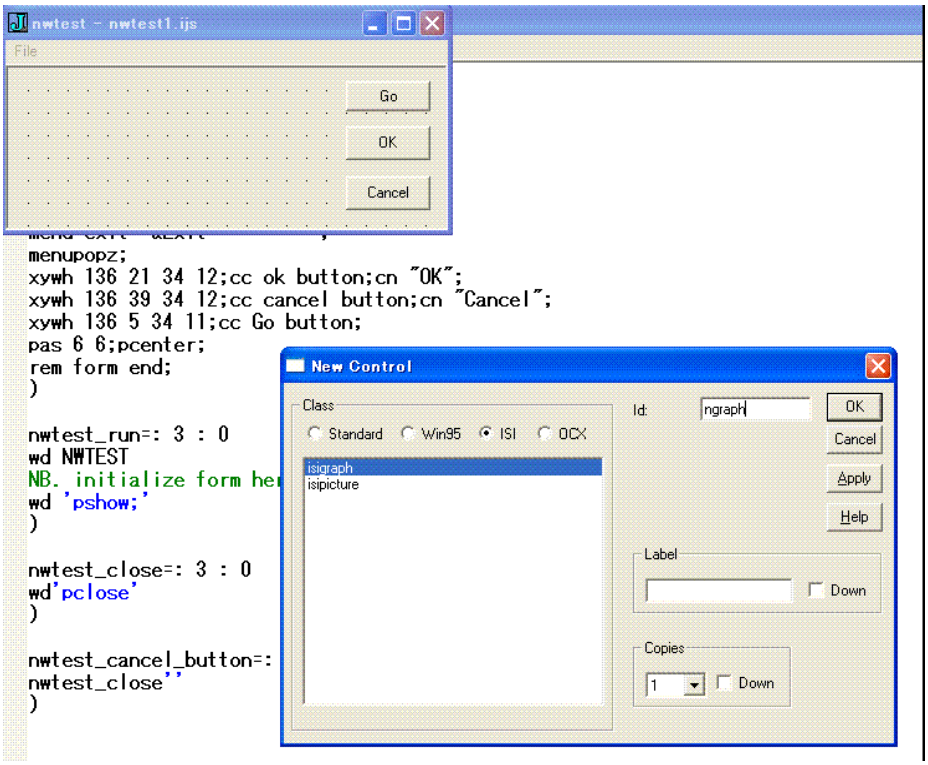
ここでは、フォーム部品の大きさ、位置などがマウス操作で対話的に変更調整ができる。同

時にその結果はフォームのスクリプト作成データ NWTEST に反映される。

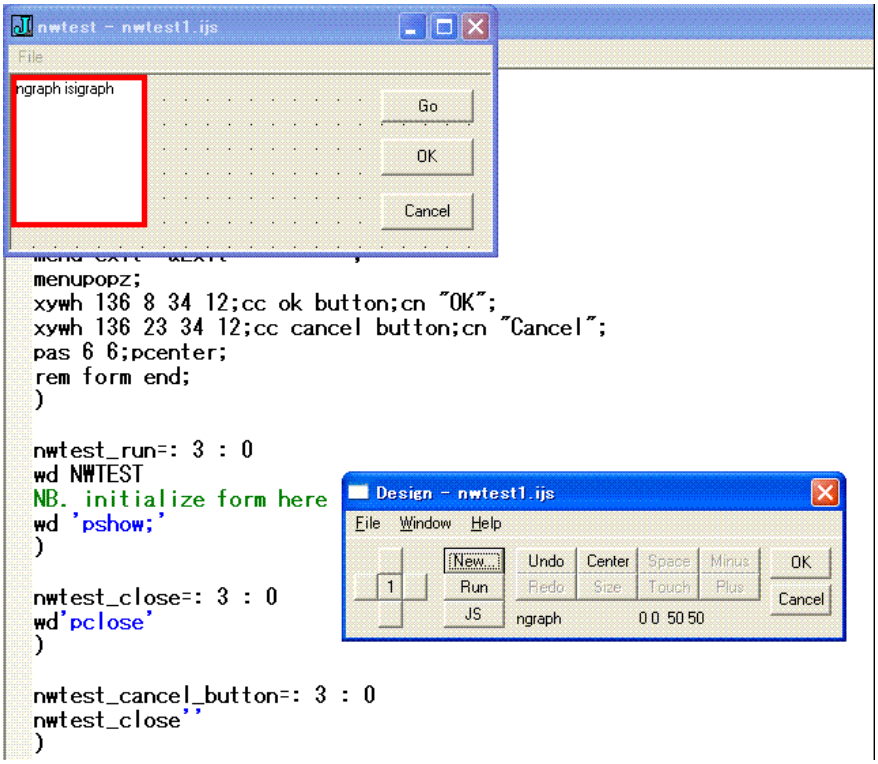
プログラマはこれを元に必要なボタン、入力ボックスなどのウィンドウ部品をさらに追加作成することが出来る。たとえば以下のようにしてもう１つボタン Go を作る。



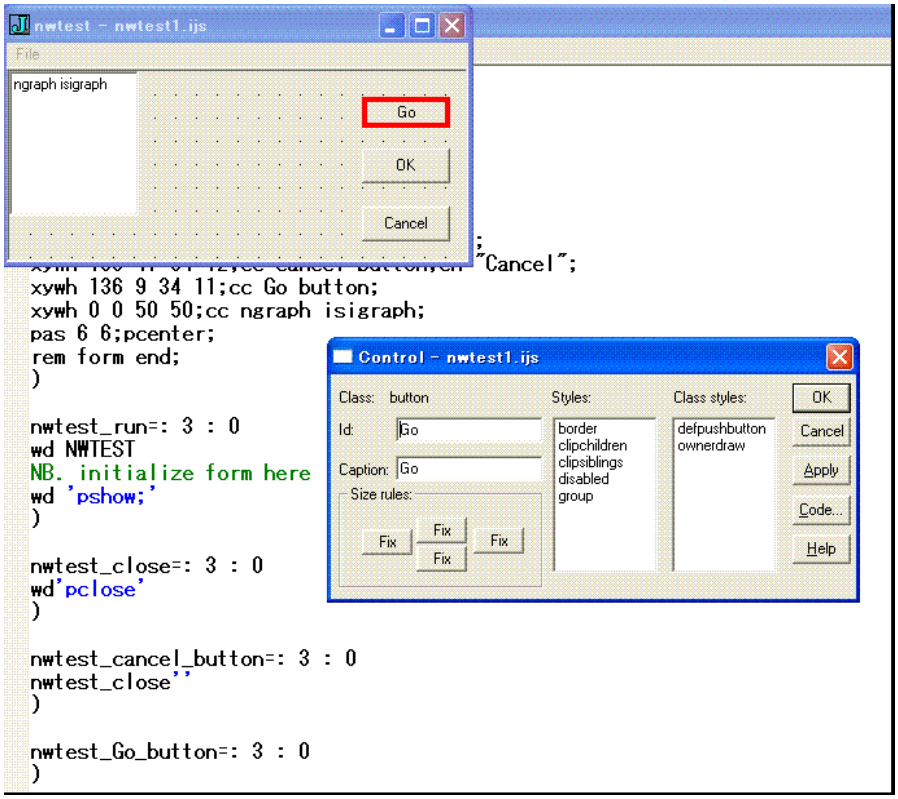
つぎに、グラフィックス用のフォーム部品を isigraph として ngraph という名前で作る。



グラフ・ウィンドウは赤ワクで作られ、これも大きさは自由に変えられる。



さて、それではこのグラフィックス画面にグラフを描いてみよう。それには先に作ったボタン Go の上でマウスをクリックする。



すると別のControlウィンドウ画面が開くのでその中でCodeボタンを選びクリックする。すると今度はスクリプト画面上にマウス・カーソルが飛んでJのプログラム・コードが入力できる状態になる。

ここにグラフ表示のプログラム・コードを書き込むことになる。

```
nwtest_Go_button=: 3 : 0
gllines 0 500 500 1000
glshow ''
)
```

つまり、上のように書き込む。

それとは別にスクリプト上で次のplotパッケージを組み込む命令が必要である。

```
require 'plot'
```

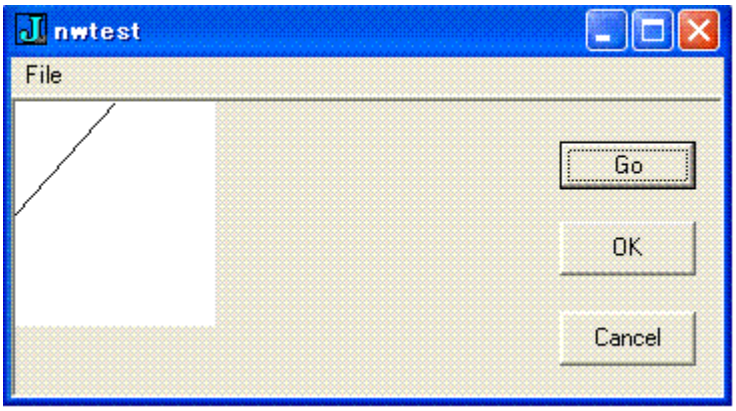
フォーム・エディタから抜け出るには、Designウィンドウ画面上でOKボタンをクリックする。

実際に、いま出来上がったスクリプト・プログラムを実行してみよう。それにはスクリプト・ウィンドウ上で、CTRL-wによりプログラムをロードし、さらに CTRL-TABで実行ウィンドウに切り替えたうえで、

```
nwtest_run ''
```

として実行する。

Jのウィンドウズ・プログラムが実行され画面が表示される。ここでボタンGoをクリックすると、以下のような簡単な図形が描かれれば成功である。



なお、先のグラフ表示のプログラム・コードの意味は次のとおりである。isigraphの仕様ではウィンドウ画面の左下が座標の(0,0)、右上が座標の(1000, 1000)になっていて、命令gllinesの引数の値により２点を結ぶ直線を描く。したがって図のような図形となる。

プログラムを終了するには右上の×印でも良いが、ウィンドウズのプログラム操作によればボタンCancelをクリックして行う。

以上でJの極く基本的ウィンドウズ・プログラミングの操作を説明した。
当日は微分方程式グラフィックスを実際にわたって解説デモンストレーションを行う。

微分方程式グラフィックスープログラムリスト

```
NB. dfield J4,J5 version 2006/11/2 by T. Nishikawa
NB.       revised from suggestion by Y. Nakano
NB. ODE Direction Field and Numerical Calculation

require 'gl2'
require 'isigraph'
coinsert 'jgl2' NB. required only for J5

DFIELD=: 0 : 0
pc dfield;
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep;
menu exit "&Exit" "" "" "";
menupopz;
xywh 183 113 34 12;cc clear button;cn "Clear";
xywh 182 134 34 12;cc cancel button;cn "Exit";
xywh 9 21 167 141;cc graph isigraph;
xywh 182 5 50 12;cc go button;cn "Dir. Field";
xywh 11 7 50 10;cc label static;cn "Dif_Equation:";
xywh 68 6 108 11;cc DE edit ws_border es_autohscroll;
xywh 183 68 50 12;cc sol button;cn "Part. Sol.";
xywh 183 85 78 12;cc label static;cn "Initial Position:";
xywh 183 98 50 11;cc POS edit ws_border es_autohscroll;
xywh 183 20 42 10;cc label static;cn "x-range:";
xywh 183 31 49 11;cc xrange edit ws_border es_autohscroll;
xywh 182 42 50 10;cc label static;cn "y-range:";
xywh 184 52 50 11;cc yrange edit ws_border es_autohscroll;
pas 6 6;pcenter;
rem form end;
)

run =: dfield_run
```

```
dfield_run=: 3 : 0
wd DFIELD
NB. initialize form here
XR =: _4, 4
'XRA XRB' =: XR
YR =: _4, 4
'YRA YRB' =: YR
range_set ''
icol =: 0
wd 'pshow;'
)

dfield_clear_button=: 3 : 0
glclear ''
icol =: 0
range_set ''
glshow ''
)

dfield_close=: 3 : 0
wd'pclose'
)



NB. Imported from dfield.js J3 version ================================
NB. Ordinary Differential Equation(ODE)
NB. Direction Field Graphic Display 2006/9/27
NB. and DE Numerical Solution       2006/9/28
NB. revised for any x, y ranges     2006/10/6

NB. Conversion 'y / x' => '(1&{) % (0&{)'
df =: 3 : 0
require 'regex'
y =. y.
if. '-' = {. y do. y =. '0 ', y end.
y =. '\([[:space:]]*-' ('(0'&,@(}.)) rxapply y  NB. '(-' => '(0-'
```

```
y =. '[[:digit:]]+\.*[[:digit:]]*' (,&'"_') rxapply y
y =. ('\/';'%') rxrplc y
y =. ('sqrt';'%:@') rxrplc y
y =. ('sin';'1: o. ') rxrplc y
y =. ('cos';'2: o. ') rxrplc y
y =. ('tan';'3: o. ') rxrplc y
y =. ('cot';'4: o. ') rxrplc y
y =. ('exp';'^@') rxrplc y
y =. ('log';'^.@') rxrplc y
y =. ('x';'(0&{)') rxrplc y
y =. ('y';'(1&{)') rxrplc y
'(', y, ')'
)


NB. (d '*:@y - *:@x') dfunc 2 3 => 5
dfunc =: 1 : 0
'x y' =. y.
z =. ". u. ,' ','"(1) ', (":x),' ',(":y)
x, y, z
)




range_set =: 3 : 0
glclear ''
glrgb 0 0 255
glpen 1 0
draw"(1) > (XRA, 0, XRB, 0);(0, YRA, 0, YRB)          NB. draw x-axis and y-axis
draw"(1)    (((>:XRA)+i.(>:    XRB-XRA)),._0.1),"(1)    (((>:XRA)+i.(>:    XRB-XRA)),.0.1)
          NB. draw x-graduations
draw"(1)    (_0.1,.((>:YRA)+i.(>:    YRB-YRA))),"(1)    (0.1,.((>:YRA)+i.(>:    YRB-YRA)))
          NB. draw y-graduations
glshow ''
)
```

```
NB. draw collection of lines, given as x, y in array
draw =: 3 : 0
'x0 y0 x1 y1' =. y.
gllines"(1) (XR Adj x0), (YR Adj y0), (XR Adj x1), (YR Adj y1)
NB. wd 'glines ', ": (XR Adj x0), (YR Adj y0), (XR Adj x1), (YR Adj y1)
)

dfield_cancel_button=: 3 : 0
wd 'pclose;'
)

NB. Input Equation in Edit Box
dfield_DE_button=: 3 : 0
Eq =: DE
icol =: 0
)

NB. Color Data
COL =: 255 0 0;255 0 255;0 128 0;0 255 0;0 255 255;128 0 128;128 128 0;0 128 128

NB. DE numerical solution
dfield_sol_button=: 3 : 0
glrgb"(1) > (8|icol){COL
icol =: icol + 1
glpen 1 0
DA =. 0.2 (df Eq) rk^:(i. >: >. 5*XRB) PX, PY
'DXA DYA' =. |: DA
DXYA =. , (XR Adj DXA),.(YR Adj DYA)
DB =. _0.2 (df Eq) rk^:(i. >: >. 5*(-XRA)) PX, PY
'DXB DYB' =. |: DB
DXYB =. , (XR Adj DXB),.(YR Adj DYB)
gllines"(1) DXYB
gllines"(1) DXYA
glshow ''
)
```

```
NB. Input Initial Position(X, Y) in Edit Box
dfield_POS_button=: 3 : 0
'PX PY' =: ". POS
)


dfield_xrange_button=: 3 : 0
XR =: ". xrange
'XRA XRB' =: XR
dfield_clear_button "
)

dfield_yrange_button=: 3 : 0
YR =: ". yrange
'YRA YRB' =: YR
dfield_clear_button "
)

NB. adj =: %&4.0@(500&*@(4.0&+))
Adj =: 3 : 0
:
'a b' =. x.
(y. - a) * 1000 % (b-a)
)

NB. Direction Field Display
dfield_go_button=: 3 : 0
NB. I =: 0.4 * (-@(i.@-) , }.@i.) 10
IX =: XRA + 0.4*(>:i.19)
IY =: YRA + 0.4*(>:i.19)
IXY =. >,{ IX;IY
glrgb 0 0 0
glpen 1 0
0.2 draw_arrow"(1) (df Eq) dfunc"(1) IXY
)
```

```
draw_arrow =: 3 : 0
:
'ax ay' =. x. arrow y.
NB. AR =: , (_4, 4) Adj ax,.ay
AR =: , (XR Adj ax),.(YR Adj ay)
gllines"(1) AR
glshow ''
)

NB. Arrow Symbol
NB.  length arrow center slope
NB.   eg. 1 arrow _2 1 0.5
arrow =: 3 : 0
:
require 'plot'
d =. x.
'x0 y0 a' =. y.
NB. arrow figure
'p0x p0y' =: (-d), 0
'p1x p1y' =. d, 0
'p2x p2y' =. (0.9*d), (0.1*d)
'p3x p3y' =. (0.9*d), (_0.1*d)
'p4x p4y' =. d, 0
NB. plot (_4, 4, 4, _4, _4, p0x, p1x, p2x);(_4, _4, 4, 4, _4, p0y, p1y, p2y)
px =. p0x, p1x, p2x, p3x, p4x
py =. p0y, p1y, p2y, p3y, p4y
NB. rotate arrow
Cos =. 1 % %: 1 + *: a
Sin =. a % %: 1 + *: a
'xx yy' =. (2 2$Cos, (-Sin), Sin, Cos) (+/ . *) (px ,: py)
xx =. (x0) + xx
yy =. (y0) + yy
NB. plot (_4, 4, 4, _4, _4, xx);(_4, _4, 4, 4, _4, yy)
xx;yy
)
```

```
NB. Differential Equation ==================================
NB. Runge-Kutta argumented Dif_Equation /2006/9/5
NB.   inc (dif_eq)  rk^:(cycles) initial x, y
NB. Using d-verb, Enable Ordinary Notation as d 'y-x'
NB.   0.1 (df 'y-x') rk^:(i.11) 0 0
rk =: 1 : 0
:
h =. x.
x =: 0{ y.
Y =: 1{ y.
F =: u.
k1 =: ". F, (":x),',',(":Y)
k2 =. ". F, (":x + h%2),',',(":Y + h*k1%2)
k3 =. ". F, (":x + h%2),',',(":Y + h*k2%2)
k4 =. ". F, (":x + h),',',(":Y + h*k3)
(x+h), (Y + h*(k1 + (2*k2) + (2*k3) + k4)%6)
)

NB. Euler Adverb Defiition
NB. 0.1 (df'y-x') euler^:(i.11) 0 0
euler =: 1 : 0
:
h =. x.
'X Y' =. y.
k =. ". u. , (":X), ',', (":Y)
(X+h), (Y + h*k)
)

NB. 0.1 ((1&{)-(0&{)) euler0^:(i.11) 0 0
euler0 =: 1 : 0
:
h =. x.
'X Y' =. y.
k =. u. X, Y
(X+h), (Y + h*k)
)
```