JAPLA シンポジウム資料 2007/12/8

Ｊによるディジタル幾何学－その４

Ｊ幾何グラフィックスによる楕円、双曲線、放物線など軌跡

西川　利男

Ｊ幾何グラフィックスにより軌跡の問題を解いてみる。前回のアポロニウスの円も軌跡の典型的問題であった。軌跡には、同じアポロニウスの円錐曲線である楕円、双曲線、放物線がある。

軌跡は英語では locus, loci という。新数学事典にある trace ではない。

> locus = the set of all points whose locations is determined by stated conditions.　VNR Concise Encyclopedia of Mathematics による

すなわち、「軌跡とは、ある記述された条件にしたがって決められた位置のすべての点の集合である。」

１．軌跡のＪグラフィックス・プログラム

全体のプログラムはかなり大きなものになるので、２つのモジュールからなるオブジェクト指向プログラミングにより開発した。

・ベースプログラム(test_geom.ijs)　　軌跡の問題処理

・クラスプログラム(pgeomgraph.ijs)　グラフィック幾何の入力、表示などの操作

したがって、クラスプログラムは前回のをそのまま使用し、ベースプログラムの作成のみをおこなった。くわしくはこれまでの発表資料を参照のこと[1][2][3]。

プログラムの流れは以下のようになっている。

(1) まず、必要な点や線の作成、設定をおこなう。これはクラスプログラムで行われるので、ベースプログラムで扱うには、例えば点 A の座標値は A_A__ins とする。

(2) 軌跡の問題は Parameters ボックス内に

　　楕円のときは　 ellipse

　　双曲線のときは hyperbola

　　放物線のときは parabola

と選択して入力する。これもクラスプログラムで行われるので、選択したベースプログラムを呼び出すには、例えば elipse_base_のようにする。

(3) 各軌跡の問題で必要な数値、走査の条件など(range 幅, int 区点数, eps 収束値)は、Command/J ボックスに次の書式で入力する。

　　(m ： n) = range, int, / eps

これらの文字列はベースプログラム内の parse ルーチンにより解読して処理する。

(4) ボタン Proc1 を押すと、指定した軌跡の計算処理をベースプログラム内のそれぞれのルーチンで行い、結果として軌跡の複数の点座標値がクラスプログラムに返され画面表示される。

２．軌跡の計算プログラム

これはすべてベースプログラム内のルーチンで行われる。具体的なコードをいくつか示してみよう。詳細なプログラムは最後にあげてある。

(1) 楕円の場合
　焦点となる２つの点をAとB、軌跡点をPとするとき、線分PAとPBとに対して
　　m - (PA + PB)
が収束値eps以下になる点の集合を探し出す。ここでmは一定値である。
　楕円のルーチンellipseのコードは以下のようである。

```
ellipse =: 3 : 0
A_A =: A_A__ins              NB. 点Aの座標値(クラスプログラムから)
B_B =: B_B__ins              NB. 点Bの座標値(クラスプログラムから)
'm n ran int eps' =: parse COMMAND__ins  NB. 各種の値(クラスプログラムから)
P_P =: 0, 0                  NB. 軌跡点Pの初期値
PP =: (int, ran) range P_P   NB. 走査範囲の点集合の座標値
AP =: elip L:0 PP            NB. 楕円の条件に合うかどうかのテスト値
(> eps > L:0 ,AP) # ,PP      NB. 探し出した軌跡の点集合の座標値
)

elip =: 3 : 0
| m - (dis A_A - y.) + (dis B_B - y.) NB. 楕円の条件のテスト
)
```

(2) 双曲線の場合
　楕円の場合と同様であるが、テストの式だけが異なる。
　　m - (PA - PB)
　したがって、プログラムコードの変更も上のAPの一行と、

```
AP =: hyp L:0 PP
```

　ここで呼び出すテストのサブルーチンを変えるだけである。

```
hyp =: 3 : 0
| m - ((dis A_A - y.) - (dis B_B - y.))   NB. 双曲線の条件のテスト
)
```

(3) 放物線の場合
　放物線の場合は、上の場合よりやや複雑になる。
　線分CD（準線）と１つの点A（焦点）に対して、軌跡点をPとする。
線分CDと点Pとの距離を d_CD、点Aと点Pとの距離を　d_A　として
　　d_CD - d_A
の値が収束値eps以下になる点の集合を探し出す。
　プログラムコードの変更は次のようになる。

```
AP =: parab L:0 PP
```

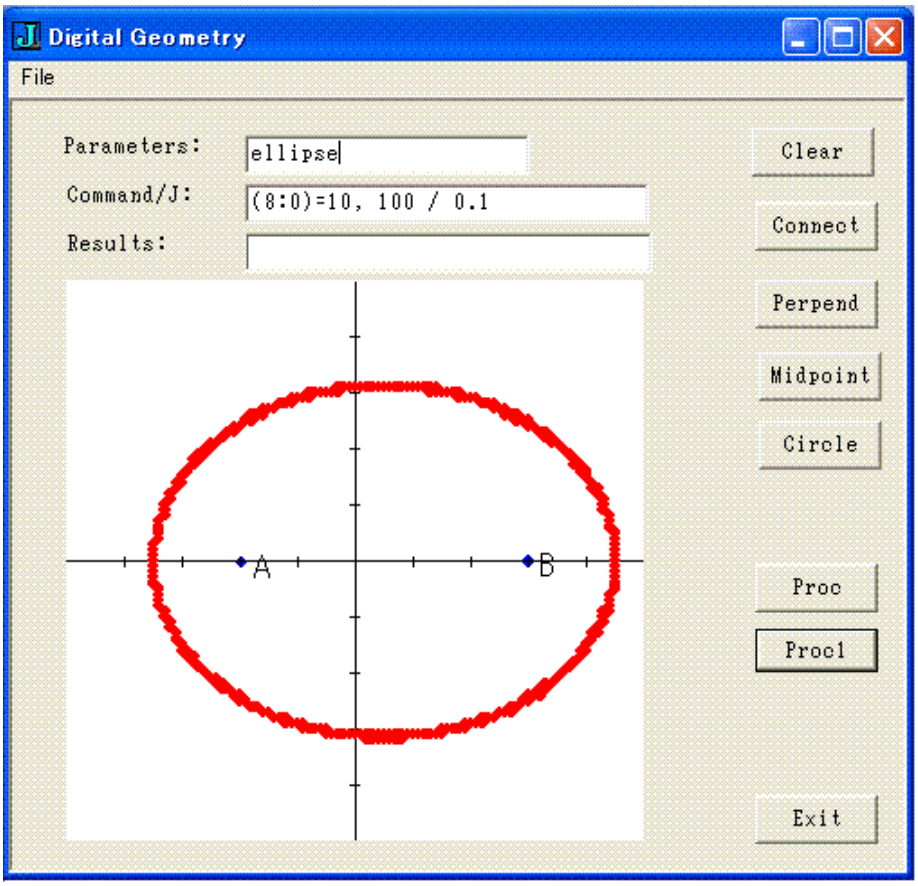
　サブルーチンparabは以下のようである。点CとDの座標値から直線の方程式を出し、点Pから直線までの距離を計算する。一方、点Aと点Pとの距離を求める。両者の距離の差が収束値eps以下になる点として、これらの座標値をテストする。

```
parab =: 3 : 0
P_P =. y.
'P_x P_y' =. P_P
'C_x C_y' =: C_C
```

```
'D_x D_y' =: D_D
M =. (D_y - C_y)%(D_x - C_x)                          NB. 直線の方程式の算出
N =. C_y - C_x*(D_y -C_y)%(D_x - C_x)
d_CD =: | ((-M*P_x)+ P_y +(-N))%(%: 1 + (-M)^2)       NB. 点Pから線分までの距離
d_A =: dis A_A - y.                                   NB. 点Pと点Aとの距離
| d_CD - d_A                                          NB. 放物線の条件のテスト
)
```

３．J幾何グラフィックスによる軌跡の実際例

(1) 楕円



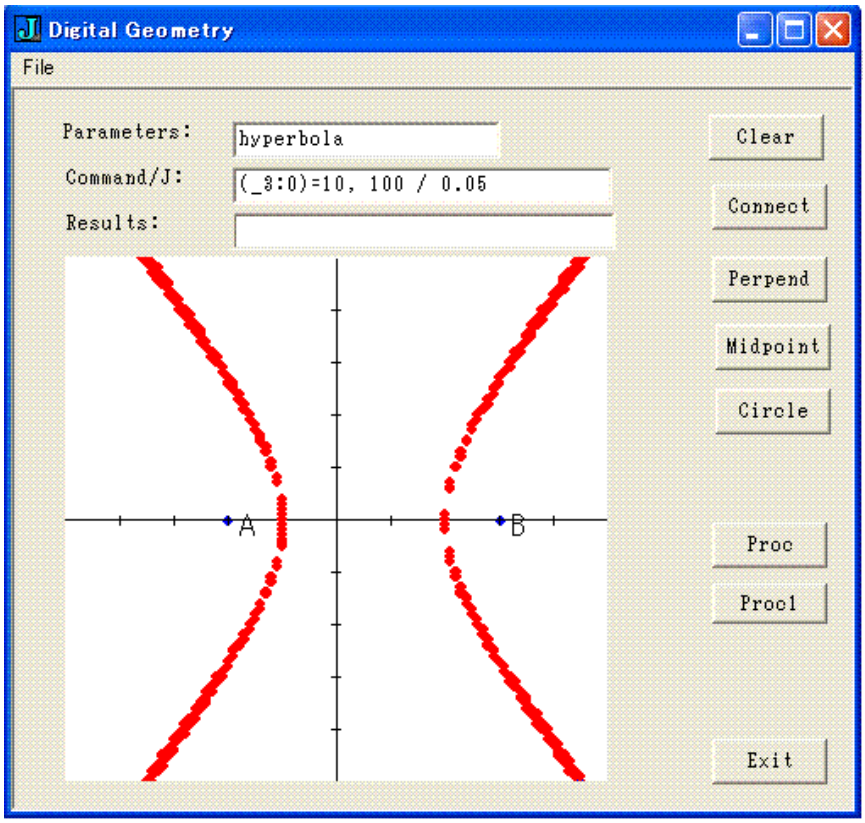
文献

[1] 西川利男「Jによるディジタル幾何学ーその１　J幾何グラフィックスとラーソンの問題」JAPLA研究会資料 2007/9/22

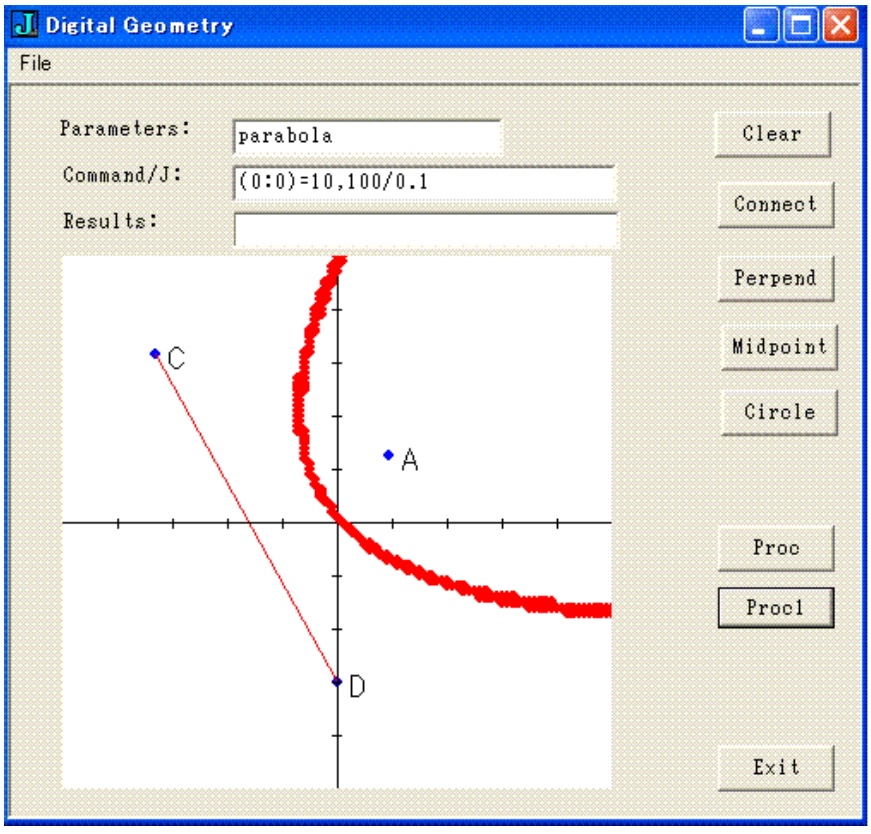
[2] 西川利男「Jによるディジタル幾何学ーその２　オブジェクト指向(OOP)による汎用・多機能化　三角形の外心、内心、シムソン線、オイラー線などへの適用」JAPLA研究会資料 2007/10/27

[3] 西川利男「Jによるディジタル幾何学ーその３　J幾何グラフィックスによりアポロニウスの円を描く」JAPLA研究会資料 2007/11/24

(2) 双曲線


Digital Geometry
File
Parameters:
hyperbola
Command/J:
(_3:0)=10, 100 / 0.05
Results:
Clear
Connect
Perpend
Midpoint
Circle
A
B
Proc
Proc1
Exit

(3) 放物線


Digital Geometry
File
Parameters:
parabola
Command/J:
(0:0)=10,100/0.1
Results:
Clear
Connect
Perpend
Midpoint
Circle
C
A
D
Proc
Proc1
Exit

プログラム・リスト

```
NB. geometrical graphics in OOP
NB. revised Parsing PARA by parse_PA 2007/10/17
NB. revised Clear, newed display_point 2007/11/19
NB. parabola 2007/12/2

corequire 'user¥Classes¥pgeomgraph.ijs'

run =: 3 : 0
ins =: '' conew 'pgeomgraph'
run__ins ''
)

reset =: 3 : 0
coerase conl 1
)

NB. Parabola ================================================
parabola =: 3 : 0
A_A =: A_A__ins
C_C =: C_C__ins
D_D =: D_D__ins
'm n ran int eps' =: parse COMMAND__ins
P_P =. 0, 0
PP =: (int, ran) range P_P
AP =: parab L:0 PP
(> eps > L:0 ,AP) # ,PP
)

parab =: 3 : 0
P_P =. y.
'P_x P_y' =. P_P
'C_x C_y' =: C_C
'D_x D_y' =: D_D
M =. (D_y - C_y)%(D_x - C_x)
N =. C_y - C_x*(D_y -C_y)%(D_x - C_x)
d_CD =: | ((-M*P_x)+ P_y +(-N))%(%: 1 + (-M)^2) NB. distance of P and CD
d_A =: dis A_A - y.
| d_CD - d_A
)

NB. Ellipse and Hyperbola ===================================
ellipse =: 3 : 0
A_A =: A_A__ins   NB. global point data
B_B =: B_B__ins   NB. ,,
```

```
'm n ran int eps' =: parse COMMAND__ins
P_P =: 0, 0
PP =: (int, ran) range P_P
AP =: elip L:0 PP
(> eps > L:0 ,AP) # ,PP
)

elip =: 3 : 0
| m - (dis A_A - y.) + (dis B_B - y.)
)

hyperbola =: 3 : 0
A_A =: A_A__ins   NB. global point data
B_B =: B_B__ins   NB. ,,
'm n ran int eps' =: parse COMMAND__ins
P_P =: 0, 0
PP =: (int, ran) range P_P
AP =: hyp L:0 PP
(> eps > L:0 ,AP) # ,PP
)

hyp =: 3 : 0
| m - ((dis A_A - y.) - (dis B_B - y.))
)

dis =: %:@(+/@:*:)

NB. (4, 2) range (2, 3) => 5 x 5 values  centered at (2, 3)
NB.  4 intervals of total width 2
NB.  revised 2007/9/27
range =: 3 : 0
:
ra =: (((-@-:) + i.@(>:)) % ]) ({.x.)
px =: ({.y.) + ({:x.)*ra
py =: ({:y.) + ({:x.)*ra
|: (<"(0) px) (,L:0)/"(0 1) (<"(0) py)
)
```

```
NB. Parse Command String
NB. parse COMMAND__ins => m, n, ran, int, eps
parse =: 3 : 0
PA0 =. y.
m =. ". ('(';':') extract PA0
n =. ". (':';')') extract PA0
if. '=' e. PA0
 do.
  ran =. ". ('=';',') extract PA0            NB. range
  if. ran = 0 do. ran =. 8 end.
  int =. ". (',';'/') extract PA0            NB. point of intervals
  if. int = 0 do. int =. 64 end.
  eps =. ". ('/';'') extract PA0             NB. epsilon
  if. eps = 0 do. eps =. 0.1 end.
 else.
  ran =. 8
  int =. 128
  eps =. 0.1
end.
m, n, ran, int, eps
)

NB. (from;till) extract string => extracted string
NB.  e.g. PARA = 'ABC,P=R(circum)'
NB.   ('';',') extract PARA => ABC
NB.   (',';'=') extract PARA => P
NB.   ('=';'(') extract PARA => R
NB.   ('(';')') extract PARA => circum
extract =: 3 : 0
:
('x0';'x1')=: x.
y0 =. y.
y1 =. (y0 i. x1){.y0
y2 =. (>: y1 i. x0)}.y1
)
```

```
NB. geometrical graphics in OOP
NB. class program
coclass'pgeomgraph'

create=:3 : 0
proc =: y.
)

destroy=:codestroy

wr=: 1!:2&2

require 'trig numeric'
require 'gl2'

GEOM=: 0 : 0
pc geom closeok;pn "Digital Geometry";
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 206 171 34 12;cc cancel button;cn "Exit";
xywh 15 44 160 138;cc geomgr isigraph;
xywh 205 7 34 12;cc Clear button;
xywh 207 79 34 12;cc Circle button;
xywh 206 25 34 12;cc Connect button;
xywh 14 8 50 10;cc label0 static;cn "Parameters:";
xywh 64 8 80 11;cc Param edit ws_border es_autohscroll;
xywh 206 44 34 12;cc Perpend button;
xywh 64 32 114 11;cc Result edit ws_border es_autohscroll;
xywh 15 32 50 10;cc label1 static;cn "Results:";
xywh 206 114 34 12;cc Proc button;
xywh 207 62 34 12;cc Midpoint button;
xywh 206 130 34 11;cc Proc1 button;
xywh 64 20 113 11;cc Calc edit ws_border es_autohscroll;
xywh 15 20 43 10;cc calu static;cn "Command/J:";
pas 6 6;pcenter;
rem form end;
)

run =: geom_run
geom_run=: 3 : 0
wd GEOM
```

```
NB. initialize form here
grid ''
wd 'setfocus Param'
glshow ''
wd 'pshow;'
)

NB. Input Parameters
geom_Param_button=: 3 : 0
PARA =. Param
PARA =: PARA -. ' '
NB. To run base program, enter e.g. pr=circum
NB. if. 1 e. 'pr=' E. PARA do. pr =: (>:PARA i. '=')}.PARA end.
)

NB. (from;till) extract string => extracted string
NB.   PARA = 'ABC,P=R(circum)'
NB.   (',';'=') extract PARA => P
NB.   ('';',') extract PARA => ABC
NB.   ('(';')') extract PARA => circum
extract =: 3 : 0
:
('x0';'x1')=: x.
y0 =. y.
y1 =. (y0 i. x1){.y0
y2 =. (>: y1 i. x0)}.y1
)

NB. if small character, then large character
small_char =: 3 : 0
if. (96&< *. 123&>)@(a.&i.) y.
 do.
   ((_32&+) &. (a.&i.)) y.
 else. y.
 end.
)

smallchk =: ((96&< *. 123&>)@(a.&i.))"(0)
small2large =: (]`((_32&+) &. (a.&i.))@.((96&< *. 123&>)@(a.&i.)))"(0)

NB. Proc_Button = execute myproc_base_ , point result ========
NB. Format: eg. ABC,P=Q(circum) in PARA edit box
NB. any name enable for point name 2007/10/2
NB. Formerly Larson = Maximiz Product of Distances of Perpendicular Foot
geom_Proc_button=: 3 : 0
```

```
NB. parsing by extract routine
P1 =. (',';'=') extract PARA    NB. initial point
NB. in case small character as 'p', then erase initial point 'P'
NB. eg. ABC,p=Q(circum)     2007/10/22
if. (96&< *. 123&>)@(a.&i.) P1  NB. small check
  do.
      ix =. PARA i. P1
      P11 =.  ((_32&+) &. (a.&i.)) P1  NB. small to large
      PARA =: P11 ix } PARA
      NB. erase point & name as follows
      PP0 =. P11,'_',P11
      'PX0 PY0' =. val2pixel ". PP0
      glrgb 255 255 255
      glpen 1 0
      glncircle PX0, PY0, 20
      glbrush ''
      glflood PX0, PY0, 255, 255, 255
      gltextxy (PX0+20), (PY0+40)
      gltext '    '
      gltextxy (PX0+20), (PY0+20)
      gltext '    '
  end.
P0 =. ('';',') extract PARA           NB. triangle
P2 =. ('=';'(') extract PARA    NB. result point
P3 =. ('(';')') extract PARA    NB. procedure in base locale
NB. execute proc from locale base_ , Ret = return values
NB. enter base program name in edit box Param
NB. P1 =. small_char"(0) P1
pr =. P3
Ret =: ". pr,'_base_', ' ,,,,,'
RT =. 2{. Ret
RMax =. 2}. Ret
glrgb 255 0 0
glpen 1 0
red_dot (val2pixel RT), 10
glrgb 0 0 0
glpen 1 0
gltextxy 20 + val2pixel RT      NB. write point name
gltext P2
glshow ''
wd 'setfocus Param'
wd 'set Result ', (": RMax)
NB. ". ('RT'), ' =: ', ": RT
". (P2,'_',P2), ' =: ', ": RT
)
```

```
red_dots =: 3 : 0
red_dot y. , 10
)

NB. For Apollonius's Circle, plot points RT
geom_Proc1_button=: 3 : 0
pr =: PARA
Ret =: ". pr,'_base_', ' '''''
NB. Ret =: apollo_base_ ''
RT =: Ret
NB. RT =: 2{. Ret
NB. RMax =: 2}. Ret
NB. RT =: (0, 1);(1, 2);(2, 2)
glrgb 255 0 0
glpen 1 0
RR =: val2pixel L:0 RT
red_dots L:0 RR
glshow ''
wd 'setfocus Param'
)




NB. Command and J-Calculator ====================================
patix =: (1: i.~ ([ E. ]))
geom_Calc_button=: 3 : 0
Calc =. Calc -. ' '
if. 1 e. 'J:' E. Calc
  do.
    COMMAND =: ('J:' patix Calc){. Calc
    JCALC =: (>: >: 'J:' patix Calc)}. Calc
    wd 'set Result *', (": ". JCALC)
  else.
    COMMAND =: Calc
end.
)

NB. eg. dis A_B
dis =: %:@(+/@:*:)

NB. eg. ang 'ABC'
ang =: 3 : 0
'A B C' =: y.
```

```
ab =. (". A,'_',A) - (". B,'_',B)
bc =. (". B,'_',B) - (". C,'_',C)
atab =. 180p_1 * _3&o.@({: % {.) ab
atbc =. 180p_1 * _3&o.@({: % {.) bc
if. atbc < 0 do. ANG =: 180 - (atab - atbc) end.
if. atbc > 0 do. ANG =: atbc - atab end.
if. atab > 0 do. ANG =: 180 - (atab - atbc) end.
if. atab < 0 do. ANG =: atbc - atab end.
)

NB. Proc & J-Calc ====================================

NB. P's 0 to 2, 5x5 values
NB. PX =: -: i.5
NB. 8 intervals at center 1 i.e.  0, 0.25,, 1.75, 2
PX =: 1 + -: -: ((-@-:) + i.@(>:)) 8
PP =: |: (<"(0) PX) (,L:0)/"(0 1) (<"(0) PX)

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

NB. Draw Circle by T.Nishikawa
NB. glncircle x, y, r
glncircle =: 3 : 0
'x y r' =. y.
glellipse (x-r), (y-r), (2*r), (2*r)
glshow ''
)

red_dot =: 3 : 0
'x y r' =. y.
glellipse (x-r), (y-r), (2*r), (2*r)
glrgb 255 0 0
glbrush ''
glflood x, y, 255, 0, 0
glshow ''
)


grid =: 3 : 0
glrgb 0 0 0
glpen 1 0
gllines 0, 500, 1000, 500 NB. x-axis
gllines 500, 0, 500, 1000 NB. y-axis
```

```
gllines L:0 <"(1) ((100 * >: i.9),.490) ,. ((100 * >: i.9),.510) NB. x-grid
gllines L:0 <"(1) (490,.(100 * >: i.9)) ,. (510,.(100 * >: i.9)) NB. y-grid
)

NB. Revised Using display_point / 2007-11-19
geom_Clear_button=: 3 : 0
glclear ''
grid ''
display_point L:0 point_list ''
glshow ''
NB. ER =. erase >(>'_' e. L:0 nl 0) # nl 0
)

display_point =: 3 : 0
PP =: ". y.
glrgb 0 0 255
glpen 1 0
glncircle (val2pixel PP), 10          NB. draw point
glbrush ''
glflood (val2pixel PP), 0, 0, 255
gltextxy 20 + val2pixel PP
gltext {. y.
glshow ''
)



NB. polar to cartesian in complex number
po2de =: {. *  (2,1)&o.@{:
NB. e.g.  *. 3j4 => 5 0.927295
NB. e.g.  po2de  5 0.927295 => 3 4

val2pixel =: 3 : '500 + 100*y.'

pix2val =: 3 : '100 %~ y. - 500'

NB. Point Move by Mouse_Left Down/Move/Up =============
NB. Connected Points (=Line) Move 2007/9/14
geom_geomgr_mbldown=: 3 : 0
d=. ". sysdata
X0=:(0{d) * 1000 % (2{d)
Y0=:(1{d) * 1000 % (3{d)
NB. pick up point
grid ''
glrgb 255 0 0
```

```
glpen 1 0
glncircle X0, Y0, 12
glrgb 0 0 255
glpen 1 0
XA =: pix2val X0
YA =: pix2val Y0
testPOIX =: */"(1) (XA, YA) in_test"(1) ". >point_list ''
testPOI =: , > testPOIX # point_list ''  NB. pick point_name
testP =: {. testPOI
glshow ''
PROD =: 1
)

geom_geomgr_mmove=: 3 : 0
d=."". sysdata
if. -.4{d do. return. end.
X1=.(0{d) * 1000 % (2{d)
Y1=.(1{d) * 1000 % (3{d)
glrgb 0 0 255
glpen 1 0
NB. gllines X0, Y0, Y1, Y2
glshow''
)

geom_geomgr_mblup=: 3 : 0
d=."". sysdata
NB. if. -.4{d do. return. end.
X2=.(0{d) * 1000 % (2{d)
Y2=.(1{d) * 1000 % (3{d)
NB. glclear ''
grid ''
NB. erase previous point / paint in white
glrgb 255 255 255
glpen 1 0
glncircle X0, Y0, 20
glbrush ''
glflood X0, Y0, 255, 255, 255
gltextxy (X0+20), (Y0+40)
gltext '    '
gltextxy (X0+20), (Y0+20)
gltext '    '
NB. erase connected line
testlix =: testP e. L:0 line_list ''
testLIN =: (>testlix) # line_list ''
testOTH =: '_' -.~ testP rem ,>testLIN
```

```
lin_or_circ =: 97 > a.i. testOTH NB. test character large or small
if. lin_or_circ
  do.   NB. Line for large
      gllines L:0 val2pixel L:0 ". L:0 testLIN
  else. NB. Circle for small
      'x y' =: ". ({.,>testLIN),'_',({.,>testLIN)
      r =: ". testP,'_',testOTH
glarc (val2pixel (x-r), (y-r)), (100* 2* r, r), (val2pixel (x+r), y, (x+r),
y)
  end.
NB. gllines val2pixel P_P, A_A
NB. reset the point at new position / paint in blue
glrgb 0 0 255
glpen 1 0
glncircle X2, Y2, 10            NB. draw point
glbrush ''
glflood X2, Y2, 0, 0, 255
gltextxy (X2+20), (Y2+20) NB. write point name
gltext ({. testPOI)
NB. draw Line in red
if. lin_or_circ
  do.  NB. draw line in red
    glrgb 255 0 0
    glpen 1 0
    gllines L:0 (X2, Y2), L:0 val2pixel L:0 ". L:0 <"(1)
(testOTH,"(0)'_'),"(1 0)testOTH
  else. NB. draw circle in purple
    glrgb 255 0 125
    glpen 1 0
    'x y' =: pix2val X2, Y2
    r =: ". testP,'_',testOTH
    glarc (val2pixel (x-r), (y-r)), (100* 2* r, r), (val2pixel (x+r), y,
(x+r), y)
  end.
grid ''
NB. gltextxy (X2+20), (Y2+40)
NB. gltext DA
glshow''
NB. renew point position values
XB =: pix2val X2
YB =: pix2val Y2
". (testPOI), ' =: ', ": XB, YB
if. -. lin_or_circ do. return. end. NB. if circle, then end
NB. renew connected lines OK 2007/9/14
LL =: ": L:0 (XB, YB), L:0 ". L:0 <"(1) (testOTH,"(0)'_'),"(1 0)testOTH
```

```
LLL =: ' =: ',L:0  (2 1$LL)
″. L:0 (2 1$testLIN) ,″(1) L:0  LLL
NB. ″. (,>testLIN), ' =: ', ″: XB, YB, ″. testOTH,'_',testOTH
)

NB. get points as A_A, B_B,,, from nl 0
point_list =: 3 : 0
p =. nl 0
p1 =. (3 = >#  L:0 p)#p
p2 =. (>'_' e. L:0 p1)#p1
p3 =. (2 = >#  L:0 ″. L:0 p2)#p2
)

NB. get lines as A_B, B_C,,,
line_list =: 3 : 0
p =. nl 0
p1 =. (3 = >#  L:0 p)#p
p2 =. (>'_' e. L:0 p1)#p1
p2 -. point_list ''
)

in_test =: ((> -&0.1)*.(< +&0.1))~
NB. 2.5 in_test 2.4  => 0
NB. 2.5 in_test 2.49 => 1
NB. 2.5 in_test 2.59 => 1
NB. 2.5 in_test 2.6  => 0

NB. remainder 'AC' rem 'ABCDE' => 'BDE'
rem =: (-.@(e.~))#]



NB. Point Initial Set ==============================
NB. First Enter Point Name in Parameters Edit
NB. Then Mouse_Right Down/Up in Graph Area
geom_geomgr_mbrdown=: 3 : 0
d=. ″. sysdata
X10=:(0{d) * 1000 % (2{d)
Y10=:(1{d) * 1000 % (3{d)
NB. glclear ''
grid ''
glrgb 0 0 255
glpen 1 0
glncircle X10, Y10, 10
NB. gltextalign TA_BOTTOM
```

```
XC =: pix2val X10
YC =: pix2val Y10
NB. gltext ": XC, YC
glshow ''
)

geom_geomgr_mbrup=: 3 : 0
d=.". sysdata
NB. if. -.4{d do. return. end.
X12=:(0{d) * 1000 % (2{d)
Y12=:(1{d) * 1000 % (3{d)
NB. glclear ''
grid ''
glrgb 0 0 255
glpen 1 0
NB. glncircle X10, Y10, 10
gllines X10, Y10, X12, Y12
glncircle X12, Y12, 10
glbrush ''
glflood (X12), (Y12), 0, 0, 255
gltextxy (X12+20), (Y12+20)
gltext PARA
grid ''
XD =: pix2val X12
YD =: pix2val Y12
". (PARA,'_',PARA), ' =: ', ": XD, YD
wd 'set Param ','           '
glshow''
)


NB. Draw Line connecting A, B
NB. Make Polygon e.g. ABCA 2007/9/12
NB. PARA is 'AB' ,'BC', 'ABCA' for polygon
NB. register line names for more than 3 points 2007/9/15
geom_Connect_button=: 3 : 0
glrgb 255 0 0
glpen 1 0
if. (#PARA) > 3
  do.   NB. more than 3 points
    points =: }: ,(PARA,.'_',"(0)PARA),"(1)','
    gllines val2pixel ". points
    NB. register line names 2007/9/15
    linep =: ex_asc L:0 (2<¥PARA)
    lines =: ({., '_'&,@{:) L:0 linep
```

```
    lif =: (".@(({., '_'&,@{:)@(2&#@{.))) , (".@(({., '_'&,@{:)@(2&#@{:)))
    linet =: lif L:0 linep
    ". (>lines),"(1) (' =: ',"(1) ("": >linet))
  else. NB. 2 points only
    'AA BB' =: PARA
     gllines val2pixel (". AA,'_',AA), (". BB,'_',BB)
    NB. register line names in ASCII order
    'AA BB' =: ex_asc PARA
     ". (AA,'_',BB), ' =: ', "": (". AA,'_',AA), (". BB,'_',BB)
  end.
wd 'set Param ','           '
wd 'setfocus Param'
glshow ''
NB. define function of connected line
NB. 'A_x A_y B_x B_y' =: PAXY
NB. fn0 =. A,'_',B, ' =: 3 : '
NB. fn1 =. '''(".A_y)+(((".B_y) - (".A_y))%((".B_x) - (".A_x)))*(y. -
(".A_x))'''
NB. ". fn0, fn1
)

NB. exchange character by ascii value
NB. ex_asc 'PA' => 'AP'
ex_asc =: 3 : 0
'a b' =: y.
if. 0 < (a.i.a) - (a.i.b)
  do. c =. b
      b =. a
      a =. c
      a,b
  else.
      a,b
  end.
)

NB. Draw Circle at C_C with radius C_c
NB. eg. PARA = 'P, 2.5' or 'P, PQ' 2007/9/19
NB. revised C=>CE, R=>RA 2007/9/26
geom_Circle_button=: 3 : 0
CE =. (',' i.~ PARA){. PARA
x =: {. (". CE,'_',CE)
y =: }. (". CE,'_',CE)
RA =. ' ' -.~ (>:',' i.~ PARA)}. PARA
if. ((47&<) *. (58&>))@(a.&i.@{.) RA  NB. test RA number or character?
  do.            NB. radius in number
```

```
    r =: ". RA
  else.          NB. radius between PointNames
    r0 =. | (".({.RA),'_',({.RA)) - (". ({:RA),'_',({:RA))
    r =: %: +/ *: r0
  end.
glrgb 255 0 125
glpen 1 0
glarc (val2pixel (x-r), (y-r)), (100* 2* r, r), (val2pixel (x+r), y, (x+r),
y)
grid ''
wd 'set Param ','           '
wd 'setfocus Param'
glshow ''
ce =. ((32&+)&.(a.&i.)) CE
". (CE,'_',ce), ' =: ', (":r)
)

subs=: [. & ((((e.&) ((# i.@#)@)) (@])) })
NB.   (',' subs ' ') 'A BC' =>  A,BC
m149 =: #~(+. 1&|.@(></¥))@(' '&~:)
dexbl =: m149
NB. delete extra blank from J Phrases p.38
first =: 4 : '(x. i.~ y.){. y.'
second =: 4 : '(>: x. i.~ y.)}. y.'

NB. Perpendicular Foot / Point_Draw, Foot_Draw & Set Point
geom_Perpend_button=: 3 : 0
NB. PARA should be 'P AB' or 'P,AB', not 'P, AB'
NB. in case 'P,AB = Q', set point as Q_Q
if. '=' e. PARA
 do.
  PARA0 =. (PARA i. '=') {. PARA
  PARA9 =. (>:PARA i. '=') }. PARA
  if. ' ' = {. PARA9 do. PARA9 =: }. PARA9 end.
 else.
  PARA0 =. PARA
end.
PARA1 =. dexbl PARA0
PARA1 =. (',' subs ' ') PARA1
PPP =. ',' first PARA1
PAB =. ',' second PARA1
if. ',' = {.PAB do. PAB =: }.PAB end.
'AAA BBB' =. PAB
'P_x P_y' =. ". PPP,'_',PPP
'A_x A_y' =. ". AAA,'_',AAA
```

```
'B_x B_y' =. ". BBB,'_',BBB
M =. (B_y - A_y)%(B_x - A_x)
N =. A_y - A_x*(B_y -A_y)%(B_x - A_x)
D =: ((-M*P_x)+ P_y +(-N))%(%: 1 + (-M)^2)
". ('d_', PPP,'_', PAB), '=:', (":D)
wd 'set Result ', (":D)
if. -. '=' e. PARA do. return. end.
Z_x =: P_x + (* M)* D * 1%(%: 1 + (-%M)^2)
Z_y =: P_y + (* M)* D * (-%M)%(%: 1 + (-%M)^2)
glrgb 0 0 255
glpen 1 0
glncircle (val2pixel Z_x, Z_y), 10
gltextxy (10 + val2pixel Z_x), (10 + val2pixel Z_y)
gltext PARA9
gllines val2pixel P_x, P_y, Z_x, Z_y
wd 'set Param ','           '
wd 'setfocus Param'
glshow ''
ZZ =: PARA9 -. ' '
". (,ZZ),'_',(,ZZ),' =: ', (": Z_x),',', (":Z_y)
NB. above revised 2007/9/29
)

NB. Larson's Problem
NB. Triangle A,B,C and Search Ponit P
NB. Larson One Point Test
geom_LarPoint_button=: 3 : 0
PARA =: 'P,AB = D'
geom_Perpend_button ''
DA =. D
PARA =: 'P,BC = E'
geom_Perpend_button ''
DB =. D
PARA =: 'P,CA = F'
geom_Perpend_button ''
DC =. D
DD =. | DA*DB*DC
wd 'set Result ', (":DD)
)



NB. 石谷茂「矢線ベクトル」p.92
NB. M =: 4r3, N =: 20r3, P_x =: _10, P_y =: 15, testP '' => 13
testP =: 3 : 0
```

```
D =: ((-M*P_x)+ P_y +(-N))%(%: 1 + (-M)^2)
)



PROD =: 1
geom_Result_button=: 3 : 0
PROD =: PROD * | D
wd 'set Result ', (": PROD)
)

NB. Inner_Point  2007/9/30
NB. PARA : 'A,B(1:2)=P' => Inner divided 1:2
NB. PARA : 'A,B=P'      => Inner equally divided
geom_Midpoint_button=: 3 : 0
if. '=' e. PARA
 do.
  PARA0 =. (PARA i. '=') {. PARA
  PARA9 =. (>:PARA i. '=') }. PARA
  if. ' ' = {. PARA9 do. PARA9 =. }. PARA9 end.
 else.
  PARA0 =. PARA
end.
PARA1 =. dexbl PARA0
PARA1 =: (',' subs ' ') PARA1
aa =: ',' first PARA1
PARA2 =: ',' second PARA1
if. ',' = {.PARA2 do. PARA2 =: }.PARA2 end.
dd =: PARA9
if. '(' e. PARA2
  do.
     bb =: (PARA2 i. '(') {. PARA2
     cc =: (PARA2 i. '(') }. PARA2
     m =. ". (>:cc i.'(') }. (cc i. ':') {. cc
     n =. ". (>:cc i.':') }. (cc i. ')') {. cc
   ". (dd,'_',dd),' =: ',": ((n*". aa,'_',aa)+(m*". bb,'_',bb))%(n+m)
  else.
     bb =: PARA2
   ". (dd,'_',dd),' =: ',": ((". aa,'_',aa) + (". bb,'_',bb))%2
  end.
glrgb 255 0 0
glpen 1 0
d_x =. {. ". dd,'_',dd
d_y =. {: ". dd,'_',dd
red_dot (val2pixel ". dd,'_',dd), 10
```

```
gltextxy (10 + val2pixel d_x), (10 + val2pixel d_y)
gltext dd
glshow ''
)
```