JAPLA 研究会資料　2007/9/22

Ｊによるディジタル幾何学－その１

Ｊ幾何グラフィックスとラーソンの問題

西川　利男

０．ディジタル幾何学とは

Ｊの上でグラフィックスをこのところ関数のグラフ表示、フラクタル・カオスなどいろいろやってきた。夏休みのこととて孫たち、岬（高校１年）、弥（中学２年）が来たので、宿題の数学やいかにと見てみた。図形の問題があったが、すぐは解けそうもない。そういえば前から幾何は苦手だった。

ふとした偶然に近くの書店で、友田勝久著「パソコンらくらく高校数学、図形と方程式」(ブルーバックス)なる本を見つけた。ここで使用している、同氏の作成によるソフトウェア GRAPES はすばらしい[1]。すっかりとりこになってしまい、有名な「九点円」などの問題を解いて、JAPLA の老 FAX メル友、中野嘉弘、山下紀幸両氏に送った。するとお返しに中野氏からは「フェルマー点」、山下氏からはラーソンの本にあったとして類似の問題が送られ[2]、こんな問題はどうかとけしかけられた。慣れない GRAPES で悪戦苦闘したが、それではいっそのこと、Ｊのグラフィックスを使ってできないものかと始めてみたのが、今回のＪのディジタル幾何学のはじまりである。

学校で学ぶ「ふつうの幾何学」（＝ユークリッド幾何学）は、本来紙と鉛筆で定規とコンパスにより描ける三角形や円などの図形の問題であり、どちらかというとアナログ的である。その後、解析幾何学も学ぶがここでは関数や方程式のグラフ表示が主だが、やはりグラフ用紙をベースにしている。

最近、幾何学はあまり人気がなく、幾何学の復権がさけばれている[3]。コンピュータ画面の上で対話的にマウス操作を活用し、ピクセルをベースにした幾何学、つまりディジタル幾何学が学校でも教えられ、使われるべきと思う。

先の GRAPES 以外には、西欧では「シンデレラ」というソフトが使われているという。Ｊ幾何グラフィックスはこれらを目指すものである。

１．Ｊとディジタル幾何学

従来の幾何学とディジタル幾何学とをＪの考え方で対比してみると興味深い。

従来幾何学	ディジタル幾何学	Ｊ
紙面上の定規とコンパス	ディスプレイ画面上のピクセル	
点	X,Y の２値/複素数値	ランク０　スカラ
線（三角形、円）	上の値の集合	ランク１　リスト
面	上の値の２次元の集合	ランク２　配列

三角形 ABC とは３つの頂点を指すのか？ 線で結ばれた図形か？ その内部の領域か？

円Ｃ（＝中心０と半径ｒ）とは円周か？ 円の内部か？

これまであいまいであったが、これらの区別をハッキリとさせることが基本になろう。

２．Ｊ幾何グラフィックスの基本の仕様と操作

まず幾何学の基本となる点、線などをその値の属性とともに次のように決める。

点 A_A = (A_x, A_y) 名詞として、名前A_Aで２つの値(A_x, A_y)を持つ。

線 A_B = (A_x, A_y, B_x, B_y)　名前A_Bで２つの点から成る。

次に、幾何学に必要ないろいろな操作をツールとして作成した。

・点を打つ……エディットボックス Parameters で点を「A」のように指定して、
グラフ画面上で位置を決めて、
マウス右ボタンのクリックで行う。

・点を動かす…上で入力した点は、マウス左ボタンのドラッグで、
グラフ画面上、自由に別の位置へ移動できる。

・線を結ぶ……エディットボックスで、例えば「AB」のように指定すると、
ボタン Connect により点AとBとを結んで線を引く。また、
（多角形）　「ABCA」のようにすると、これらを頂点として三角形を描く。

・円を描く……エディットボックスで、例えば「P, r」のように指定すると、
ボタン Circle により中心(P), 半径(r)の円を描く。また、
「P, PQ」とすると、線分PQを半径とする円を描く。

・点と直線との距離（垂線の足）
エディットボックスで、例えば「P,AB」のように指定した上で、
ボタン Perpend により、エディットボックス Results には
点Pと直線ABとの距離が表示される。
エディットボックスで、「P,AB = Q」のように指定したときは、
垂線の足の位置で点Qを表示し、点Pから線分ABに垂線を引く。

・中点を求める
エディットボックスで、例えば「A,B = C」のように指定して、
ボタン Midpoint により、ABの中点を求め、その点Cを表示する。

なお、実際の操作例は後に示す。

３．ラーソンの問題

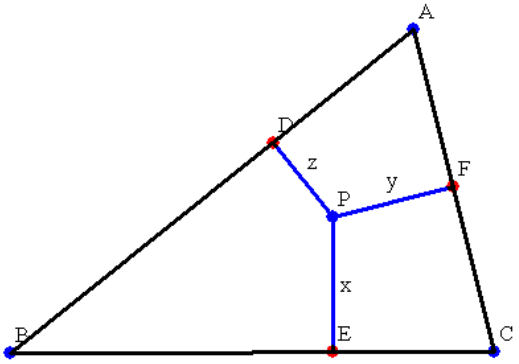
ラーソン著「数学発想ゼミナール２」p.406 [2]に次のような問題と解答・証明がのっている。

> *P*を△*ABC*の内部の点とし、点*P*から*BC*, *CA*, *AB*への距離をそれぞれ*x*, *y*, *z*とする。積*xyz*の値を最大にする点の位置はどこか。

解答

点*P*は△*ABC*の各頂点からそれぞれに対する辺の中線のとの交点、すなわち点*P*は△*ABC*の重心に一致する。

（証明は省略）



４．Ｊ幾何グラフィックスでラーソンの問題を解く

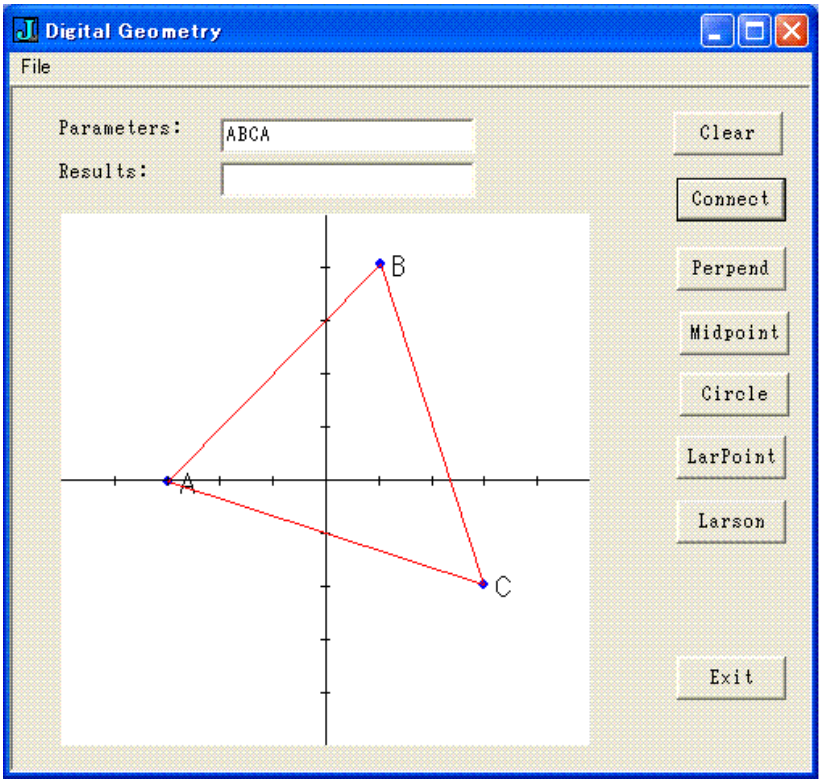
４．１　三角形を作る

まず、エディットボックスに「A」と入力し、グラフ画面の適当な位置でマウスの右ボタンのクリックすると点が入れられる。

同様にして「B」と「C」も作る。

なお、点の位置を修正したいときは、マウスの左ボタンをドラッグすると自由に位置が変更できる。

次にエディットボックスに「ABCA」と入力して、Connect ボタンを押せば、三角形 ABC が描かれる。

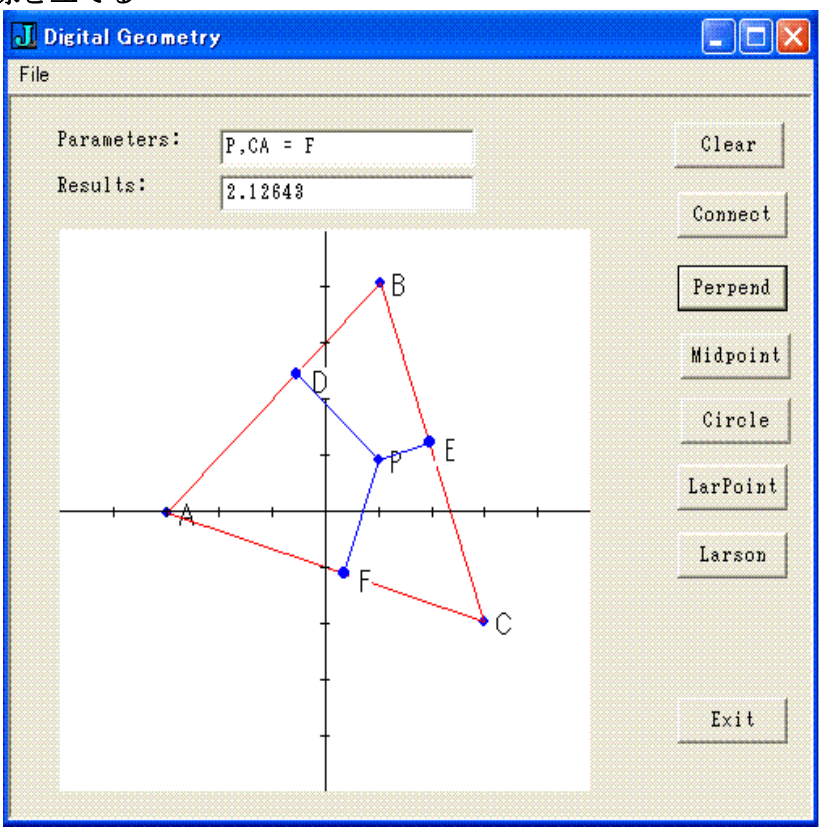


４．２　点を打って、垂線を立てる

次に三角形の内部の適当な位置に点 P を打つ。エディットボックスに「P,AB = D」と入力し、Perpend ボタンをクリックすると、点 P から辺 AB に垂線が引かれる。

そのとき同時にボックス Results にはその距離が表示される。

同様にして、点 P から辺 BC，辺 CA にも垂線を立てる。



４．３　点を動かし、ラーソンの条件（各辺までの距離の積の最大）を探す

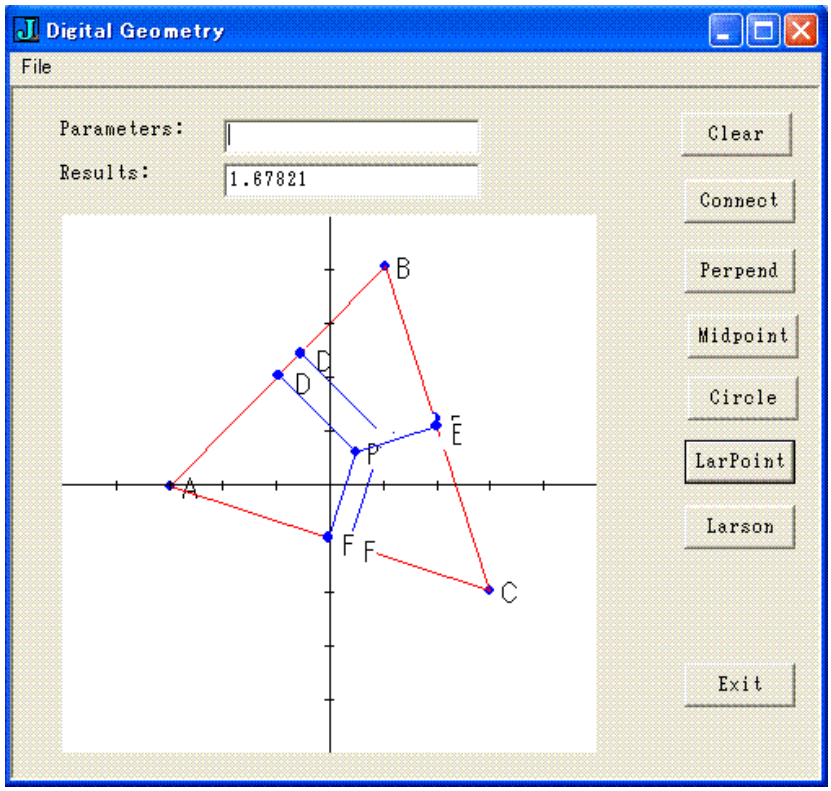
ボタンLarPointはこの手順をまとめて行うように定義してある。

前と同様にして三角形の内部に適当な位置に点Pを作る。

ボタンLarPointを押すと三角形への垂線とその距離の計算と行い、Resultsには３つの距離の積を表示する。

次には点Pをドラッグして別の位置でボタンLarPointを押すとResultsには前とは異なる数値が示されるだろう。

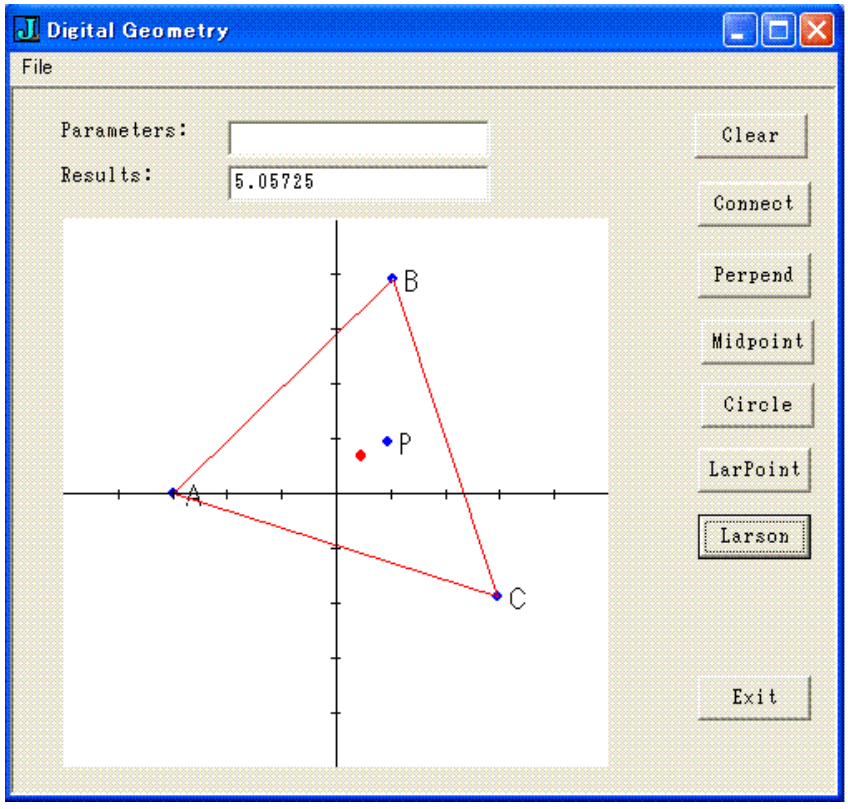
このようにして、最大値になる点Pを探してみよう。



４．４　ラーソンの問題を自動的に解く

ラーソンの条件に合う点を１点ずつマニュアルで調べるのは仲々大変である。自動的に距離の積の最大値になるような点を求めるプログラムを作った。

三角形の内部に適当に点Pを作り、ボタンLarsonを押すと点Pを中心にある範囲の複数の点を生成し、その中から最大になる点を探し出して表示する。



４．５　重心と一致することを確かめる

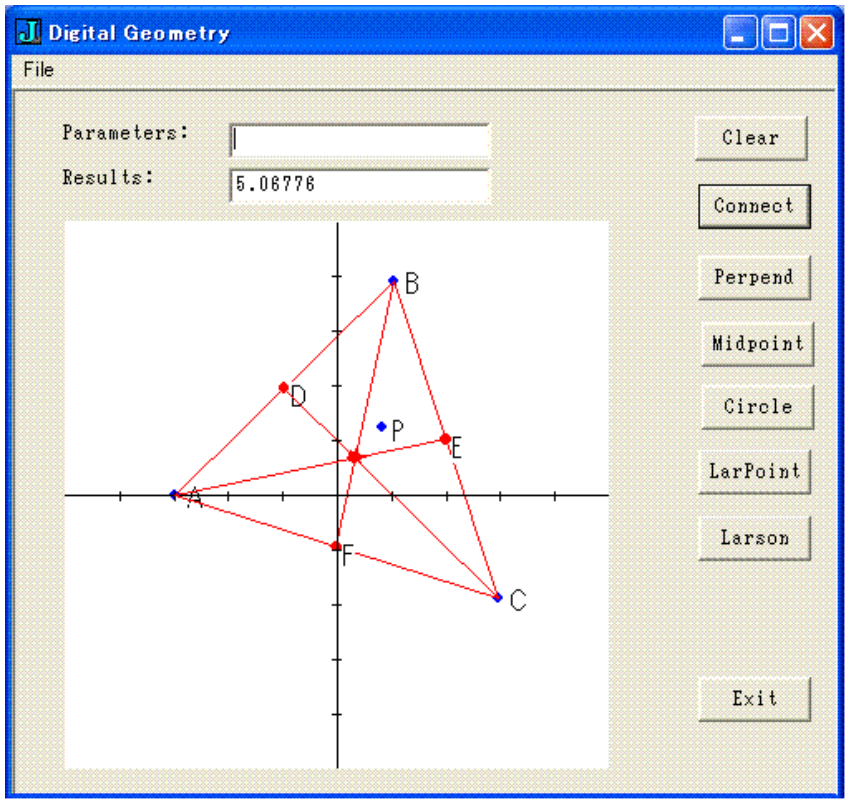
ラーソンの点が三角形の重心に一致することを確かめてみよう。

三角形の辺 AB の中点を求めるには、エディットボックスに「A,B = D」と入力して、ボタン Midpoint を押すと中点の位置に点 D と示される。

他の２辺 BC, CA も同様にして中点を求める。

その後、点 C と D、点 B と F、点 A と E とをそれぞれ結ぶと一点に交わる。

つまり、重心と一致することが示された。



５．おわりに－システムを作って、使っての一言

GRAPES と同じような「幾何グラフィックス」をと意気込んで始めたものの、例えば、「垂線を立てる」「中点を求める」といった数学の処理はそれほどのことはないが、「マウスの左ドラッグで、グラフ上で点とそれに連結した線とを移動する」などユーザ・インターフェースにかかわるプログラミングは仲々大変であった。

出来上がったシステムを使ってみて、「紙と鉛筆の幾何」とどこが違うのか、同じではないか。あたりまえである。古代ギリシャの昔より、人ははじめは砂の上に「図を書いては消し」を何遍となく繰り返した末、ピタゴラスをはじめとして、アポロニウス、フェルマーと後世に名を残す定理が出来たのだろう。

幾何学とは、まずは図形をよく観察してその不思議をゆっくりと味わうべきである。問題を見たとたん、どこに補助線を引けばよいか、証明が出来たらハイそのつぎ！－という学校幾何の安易さには憤りを感ずる。ディジタル幾何学は図形の不思議さを味わうゆとりをわれわれに与えてくれた。これこそ王道の幾何というべきものと思う。

文献/参考書

[1] 友田勝久「関数グラフソフト GRAPES パーフェクトガイド」文英堂(2007).
[2] ローレン．C．ラーソン著、秋山仁、飯田博和訳「数学発想ゼミナールー１，２」シュプリンガー・フェアラーク東京(1992).

［3］ H. コセクター著、銀林浩訳「幾何学入門、第２版」明治図書(1965).

プログラム・リスト

```
NB. Geometrical Graphics - geom_Larson.js
NB. Larson's Problem
NB. ローレン・C・ラーソン「数学発想ゼミナール」シュプリンガー東京(1992).
NB.  vol 2, p.406, 8.1.4

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 206 157 34 12;cc cancel button;cn "Exit";
xywh 15 35 164 147;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 20 80 11;cc Result edit ws_border es_autohscroll;
xywh 14 20 50 10;cc label1 static;cn "Results:";
xywh 206 114 34 12;cc Larson button;
xywh 207 62 34 12;cc Midpoint button;
xywh 206 96 34 12;cc LarPoint button;
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;'
```

```
)
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
)

geom_Clear_button=: 3 : 0
glclear ''
grid ''
glshow ''
ER =. erase >(>'_' e. L:0 nl 0) # nl 0
)


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. Move and Reposition Point by Mouse_Left Down/Move/Up ============
NB. Move Connected Points (=Line) 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. Input Parameters in Edit Box ===================
geom_Param_button=: 3 : 0
PARA =: Param
)

NB. Initially Set Point ============================
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)
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
geom_Circle_button=: 3 : 0
C =. (',' i.~ PARA){. PARA
x =: {. (". C,'_',C)
y =: }. (". C,'_',C)
R =. ' ' -.~ (>:',' i.~ PARA)}. PARA
if. ((47&<) *. (58&>))@(a.&i.@{.) R  NB. test R number or character?
  do.             NB. radius in number
    r =: ". R
  else.           NB. radius between PointNames
    r0 =. | (".({.R),'_',({.R)) - (". ({:R),'_',({:R))
    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 ''
c =. ((32&+)&.(a.&i.)) C
". (C,'_',c), ' =: ', (":r)
)

NB. Midpoint ====================================================
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
bb =: ',' second PARA1
if. ',' = {.bb do. bb =: }.bb end.
dd =: PARA9
". (dd,'_',dd),' =: ',": ((". aa,'_',aa) + (". bb,'_',bb))%2
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 ''
)



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,' =: ', (": Z_x),',', (":Z_y)
)


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. Maximiz Product of Distances of Perpendicular Foot ==========
NB.   call subprograms calc_LM, calc_L, calc_M, range
geom_Larson_button=: 3 : 0
PP =: range P_P
PMax =: calc_LM PP  NB. Point for Maximum Product
glrgb 255 0 0
glpen 1 0
red_dot (val2pixel 2{. PMax), 10
glshow ''
wd 'set Result ', (": {: PMax)
)

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)

range =: 3 : 0
px =. ({.y.) + -: -: ((-@-:) + i.@(>:)) 8
py =. ({:y.) + -: -: ((-@-:) + i.@(>:)) 8
|: (<"(0) px) (,L:0)/"(0 1) (<"(0) py)
)

calc_L =: 3 : 0
:
'a_a b_b'=: x.
'a_x a_y' =: a_a
'b_x b_y' =: b_b
p_p =: y.
'p_x p_y' =: p_p
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)
)

calc_M =: 3 : 0
LAB =. | ( (A_A;B_B) calc_L y.)
LBC =. | ( (B_B;C_C) calc_L y.)
LCA =. | ( (C_C;A_A) calc_L y.)
NB. LAB =. | (_3 0;0 4) calc_L y.
NB. LBC =. | (0 4;4 _1) calc_L y.
NB. LCA =. | (4 _1;_3 0) calc_L y.
```

```
LAB * LBC * LCA
)

calc_LM =: 3 : 0
LM =: calc_M L:0 ,y.
NB. wr (, y.),. LM
MaxXY =. > (({.@¥:) >LM) { , y.
MaxV =. (({.@¥:){]) >LM
MaxXY, MaxV
)


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)
)
```