JAPLA 研究会資料　2007/10/27

Ｊによるディジタル幾何学－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

西川　利男

０．はじめに

幾何の問題を解くためのＪのグラフィックスーＪ幾何グラフィックスを前回報告したが[1]、先のラーソンの問題以外に、もっといろいろな問題にも汎用的に利用したいと考えた。ところが、Ｊのプログラムはすでに相当大きく、とくにGUIのためにかなり複雑になっている。これをそのまま拡張したのでは、見通しも悪くなり何とかしなくてはならない。

Ｊでもオブジェクト指向プログラミング(OOP)が可能であり、すでに筆者はGridプログラミングを例として何回か紹介した[2,3]。今回、Ｊ幾何グラフィックスを汎用、多機能化するため、OOP手法によりプログラムの再構築を行った。

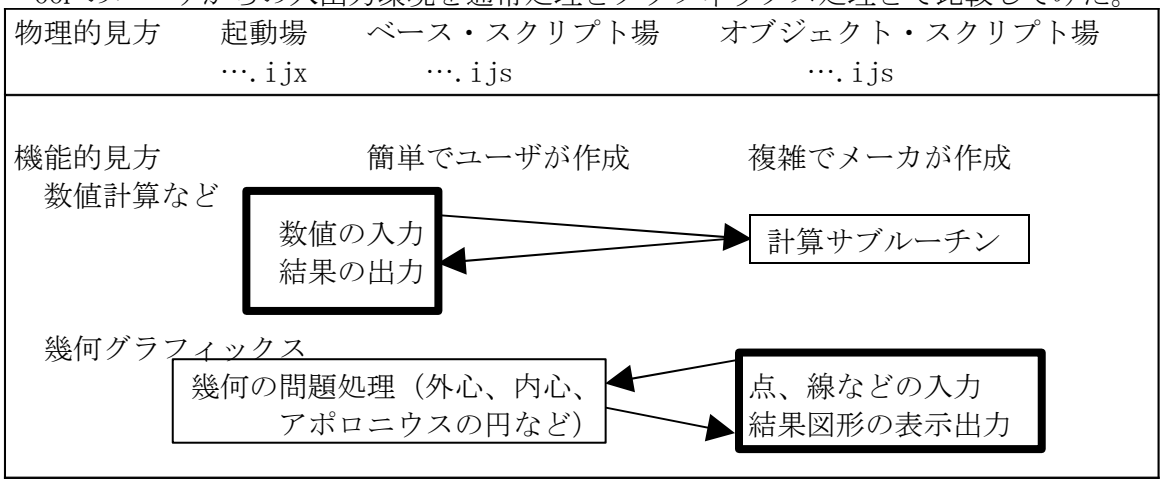
実は先のGridのOOPでは、Ｊシステムに既に備えられているクラスはほとんどそのままで、それをインスタンスとして利用する程度の使い方であった。しかし、今回はクラス・スクリプトのすべてを作ることになり、これは筆者にとってＪのOOPをより深く理解することが必要になり、かえって良い経験と収穫になった。

１．オブジェクト指向とグラフィック・ユーザインターフェース

オブジェクト指向(OOP)とは、ひと言でいうと次のようになる。

> まず、あらかじめ作られた(システム常備またはユーザ作成の)クラス・プログラムを使用するように指定し(登録)、その一例(インスタンス)(一種のコピー)を実行の環境に取り込み、通常のプログラムと同様に使用するやりかたである。

OOPのユーザからの入出力環境を通常処理とグラフィックス処理とで比較してみた。



従来の数値計算などの場合は、ユーザが対面するインターフェース、つまり入出力はキーボードとディスプレイであり、それほどむずかしくないのでユーザが作り、複雑な数値計算はクラスで作られたパッケージを利用する形がふつうである。
一方、幾何グラフィックスでは、点や線の入力にしてもマウスにかかわるかなり面倒なプログラミングを必要とする。結果の表示にしても、GUI の細かい仕様にしたがったプログラミングを行う。そしてこれらは幾何グラフィックスでは共通のものであるのでクラス・プログラムに任せた方が良い。個別の幾何の問題を解くプログラムはユーザが作るのがよい。つまり比較表内の２重のワクで示すように
「幾何グラフィックスの場合には、ユーザが対面するのはクラスで作られた幾何図形入出力環境であり、数値計算など通常の場合とは位置づけがまったく異なる。」

２．Ｊ幾何グラフィックス OOP 版のための基本命令

Ｊ幾何グラフィックス OOP 版は次の３つのモジュールから成る環境で実行される。ここで、Ｊの OOP に必要な基本の命令コードを挙げておく。

・起動場
・ベース・スクリプト …… test_geom.ijs
```
corequire 'Classes¥pgeomgraph.ijs'…… クラスとして登録
ins =: '' conew 'pgeomgraph'  …… インスタンスの作成
```
ここで、ベース・スクリプトからインスタンスの名詞、動詞は次のようにして参照される
```
noun__ins,  verb__ins  (locale の間接参照)
```
・クラス・スクリプト …… Classes¥pgeomgraph.ijs
```
coclass 'pgeomgraph'  …… クラス・スクリプトであることを宣言
create =: 3 : 0       …… インスタンス起動時に実行されるプログラム
 ……
)
```
一方、クラス・スクリプトからベース・スクリプトの名詞、動詞は次のようにして参照される
```
noun_base_,  verb_base_  (locale の直接参照)
```
クラスとベースにおける locale の直接参照と間接参照の書き方の差に注意すること。

３．Ｊ幾何グラフィックス OOP 版への追加部分

３．１　クラス・スクリプトの OOP 対応部分プログラム

ベース・プログラムを実行のために２つのボタン Proc, Proc1 とそれぞれその実行プログラムを追加作成した。
```
geom_Proc_button =: 3 : 0
 ……
RetValue =.  ". PARA, '_base_', ''''''  ベース・プログラム名を実行
plot RetValue                           戻り値を画面にプロットする
)
```
ボタン Proc1 は複数の戻り値に対して、複数の点をプロットするものである。
ここで文字列 PARA は Parameters エディト・ボックスに入力される文字列から、実行したいベース・プログラムの名前を取り出したものである。
なお、グラフィックスにおける点や線の座標値などについて、Ｊにより手軽に計算行える Command/J:なるエディット・ボックスを作った。

３．２　ベース・スクリプトのプログラム

ベース・スクリプトには、いろいろな幾何の問題を解くプログラムを書く。これらはParameters エディット・ボックスへ次の書式で入力して実行される。

(1) ラーソンの問題(重心)　ABC,P=R(gravity)
(2) 三角形の外心を求める　ABC,P=R(circum)
(3) 三角形の内心を求める　ABC,P=R(incenter)

なお、点などの名前は任意の文字でよい。

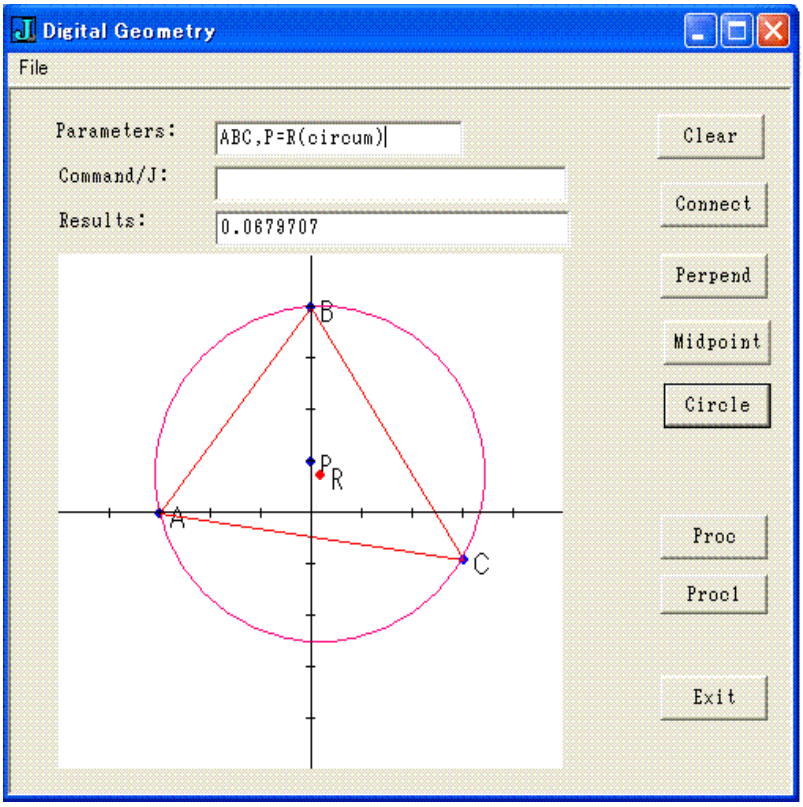
三角形の外心(外接円の中心)を例にとって示してみる。外心とは３つの頂点への長さが等しい点を求めることであり、それをそのままプログラムした。

```
circum =: 3 : 0
P_P =: P_P__ins          NB. 幾何グラフィックスにより入力した点Pの(X,Y)値
PP =: (8, 2) range P_P   NB. 点P_Pの周りに範囲2、区切り点8個の点を生成する
PP1 =: 2{. circ PP       NB. 関数circにより最適値を求めPP1とする
PP2 =: (16, 1) range PP1  NB. 点PP1の周りに範囲1、区切り点8個の点を生成
circ PP2                  NB. 関数circにより再度、最適値を求める
)

circ =: 3 : 0
PP =. y.
A_A =: A_A__ins   NB. 幾何グラフィックスにより入力した点Aの(X,Y)値
B_B =: B_B__ins   NB. 幾何グラフィックスにより入力した点Bの(X,Y)値
C_C =: C_C__ins   NB. 幾何グラフィックスにより入力した点Cの(X,Y)値
DIF =: dif L:0 PP NB. 点Pと点A, B, Cとの距離の差を求め値DIFとする
MinXY =. > ({.@/: , >DIF) { , PP
MinV =. ({.@/: , >DIF) { , > DIF
MinXY, MinV
)

dis =: %:@(+/@:*:)  NB. ２点の距離
dif =: 3 : 0        NB. 点Pと点A, B, Cとの距離の差を求める
a =: dis A_A - y.
b =: dis B_B - y.
c =: dis C_C - y.
Dif =:  | (a-b), (b-c), (c-a)
+/ Dif
)

NB. (4, 2) range (2, 3)=> 点(2, 3)を中心に区分4、範囲2の複数の点を生成する
range =: 3 : 0
:
ra =: (((-@-:) + i.@(>:)) % ]) ({.x.)
px =: ({.y.) + ({:x.)*ra
py =: ({:y.) + ({:x.)*ra
```

|: (<″(0) px) (,L:0)/″(0 1) (<″(0) py)
)

４．いくつかの実際例

４．１　三角形の外心



まず三角形ABCを描いてその内部に点Pを打つ。つぎParametersボックスには

ABC,P=R(circum)

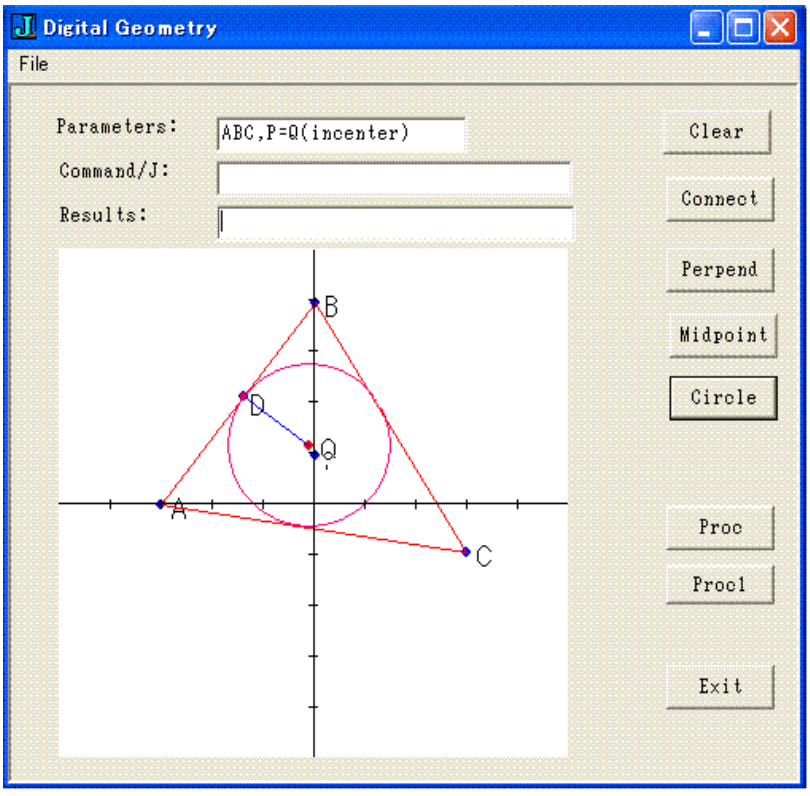
と入力して、ボタンProcを押すと、外心がRとして求められる。

さらにParametersに

R, RA

として、ボタンCircleを押すと中心R、半径RAの円が描かれ、外接円であることが確かめられる。

４．２　三角形の内心



同様にして三角形を描いてから、内部に仮の点Pを打つ。そして

ABC,P=Q(incenter)

として、ボタンProcを押すと、内心はQとして求められる。

Q,AB = D

としてボタンPerpendを押すと点Qから辺ABへの垂線が引かれその足がDと記される。

Parametersに

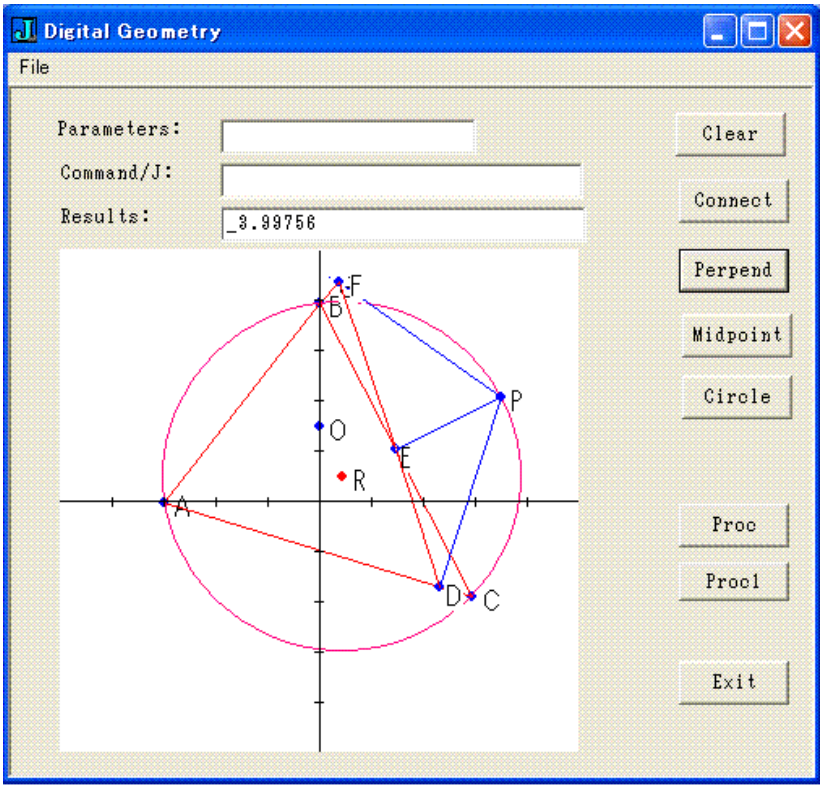
Q, QD

としてボタンCircleを押すと、内接円であることが確かめられる。

４．３　シムソン線

シムソン線とは、三角形の外接円の周上の任意の点から各辺またはその延長線上に下ろした垂線の足は同一線上にある、という定理である。

１７世紀のイギリスの数学者シムソンにより発見されたといわれるが、実はウォーレスだという説もある。



４．４　オイラー線

オイラー線とは、三角形の外心、重心、垂心は一直線に並ぶ、という定理である。

外心は

ABC,P=O(circum)

で、重心は

ABC,P=G(graviy)

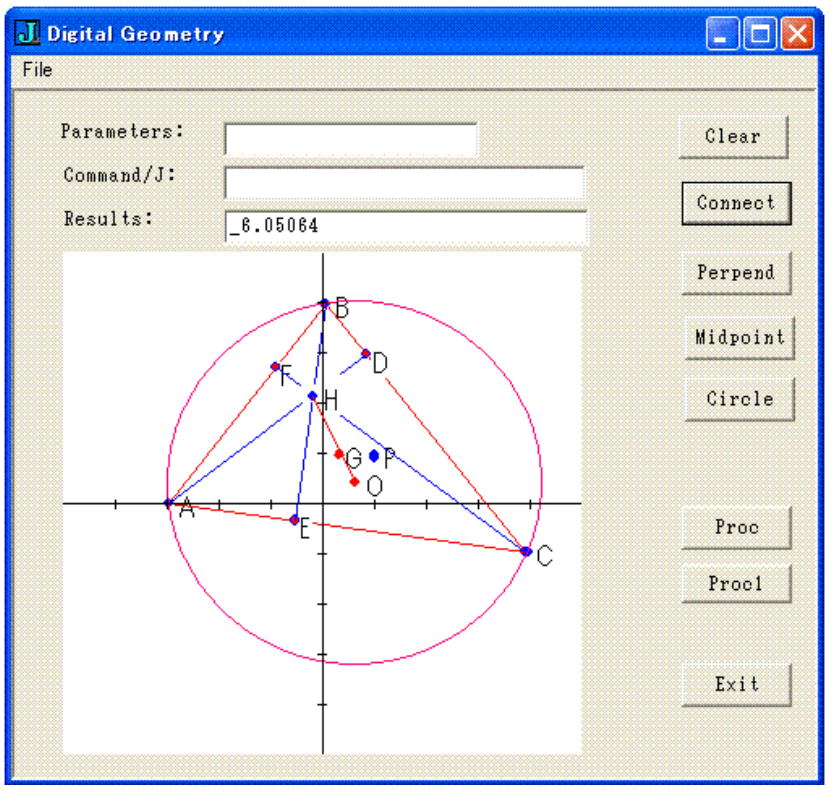
で求められる。

垂心（各頂点からその対辺に下ろした垂線の交点、、つまり

A,BC=D

でボタンPerpendを押す、を３回行って得る。

これらの３点が一直線に並ぶ。



[1] 西川利男「Ｊによるディジタル幾何学ーその１」JAPLA研究会資料 2007/9/22
[2] 西川利男「Ｊのオブジェクト指向プログラミングー簡単な例でやってみる」
[3] 西川利男「同上ーＪのスプレッドシート(Grid)と数独パズルへの適用」

NB. Base Script : test_geom.is in J4.02

```
NB. geometrical graphics in OOP
corequire 'user¥Classes¥pgeomgraph.ijs'

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

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. (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. Find Circum_center ===================================
circum =: 3 : 0
P_P =: P_P__ins
NB first search
PP =: (8, 2) range P_P
PP1 =: 2{. circ PP
NB. second search
PP2 =: (16, 1) range PP1
circ PP2
)

circ =: 3 : 0
```

```
PP =. y.
A_A =: A_A__ins   NB. global point data
B_B =: B_B__ins   NB. ,,
C_C =: C_C__ins   NB. ,,
DIF =: dif L:0 PP
MinXY =. > ({.@/: , >DIF) { , PP
MinV =. ({.@/: , >DIF) { , > DIF
MinXY, MinV
)

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

dif =: 3 : 0
a =: dis A_A - y.
b =: dis B_B - y.
c =: dis C_C - y.
Dif =:  | (a-b), (b-c), (c-a)
+/ Dif
)

NB. Inscribed Circle Center ================================
NB. 2007/9/26
incenter =: 3 : 0
P_P =: P_P__ins
PP0 =: (8, 2) range P_P
PP1 =: 2{. incen PP0
PP2 =: (16, 1) range PP1
incen PP2
)

incen =: 3 : 0
PP =. y.
A_A =: A_A__ins   NB. global point data
B_B =: B_B__ins   NB. ,,
C_C =: C_C__ins   NB. ,,
calc_CM PP
)


calc_C =: 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.)
+/ | (LAB-LBC), (LBC-LCA), (LCA-LAB)
)
```

```
calc_CM =: 3 : 0
CM =: calc_C L:0 ,y.
MinXY =. > (({.@/:) >CM) { , y.
MinV =. (({.@/:){]) >CM
MinXY, MinV
)


NB. Larson Problem/Graphical Solution ======================
NB. program called from pgeomgraph OOP
gravity =: larson
larson =: 3 : 0
PP =: (8, 2) range P_P__ins NB. ,,
A_A =: A_A__ins     NB. global point data
B_B =: B_B__ins      NB. ,,
C_C =: C_C__ins      NB. ,,
PMax =: calc_LM PP
)

NB. Larson subroutines

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.)
LAB * LBC * LCA
)

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

```
MaxXY, MaxV
)
```

NB. Class Script - should be stores as ‘user¥Classes¥pgeomgraph.ijs’
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 206 157 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_', ' '''''
R_R =. 2{. Ret
RMax =. 2}. Ret
glrgb 255 0 0
glpen 1 0
red_dot (val2pixel R_R), 10
glrgb 0 0 0
glpen 1 0
gltextxy 20 + val2pixel R_R     NB. write point name
gltext P2
glshow ''
wd 'setfocus Param'
wd 'set Result ', (": RMax)
NB. ". ('R_R'), ' =: ', ": R_R
". (P2,'_',P2), ' =: ', ": R_R
```

```
)

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

geom_Proc1_button=: 3 : 0
NB. select. proc
NB.     case. 'A' do. Ret =: apollo_base_ ''
NB. end.
Ret =: apollo_base_ ''
R_R =: Ret
NB. R_R =: 2{. Ret
NB. RMax =: 2}. Ret
NB. R_R =: (0, 1);(1, 2);(2, 2)
glrgb 255 0 0
glpen 1 0
RR =: val2pixel L:0 R_R
red_dots L:0 RR
glshow ''
wd 'setfocus Param'
)



NB. 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
)

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. 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, X1, 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)
)
```