CC3 Macro

PART 3

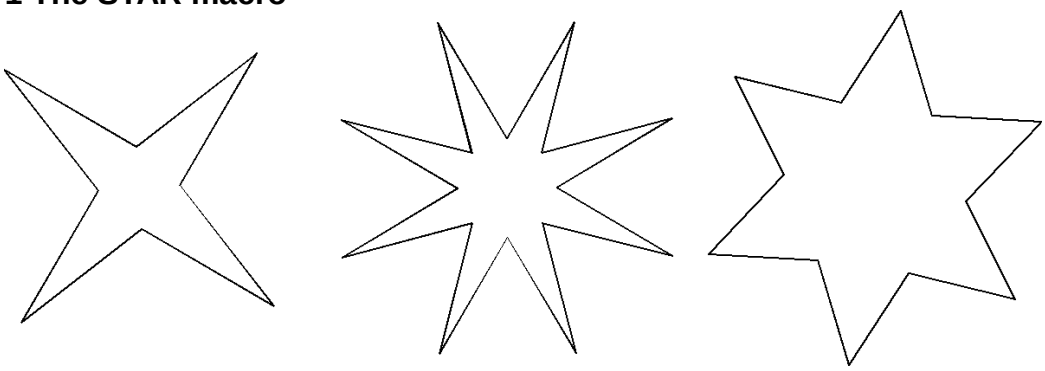
Geometric entities

Introduction

In this part, I'll introduce new commands usefull to create geometric entites or symbols.

Macros will be built step by step as before and can be found on the companion stars.mac file.

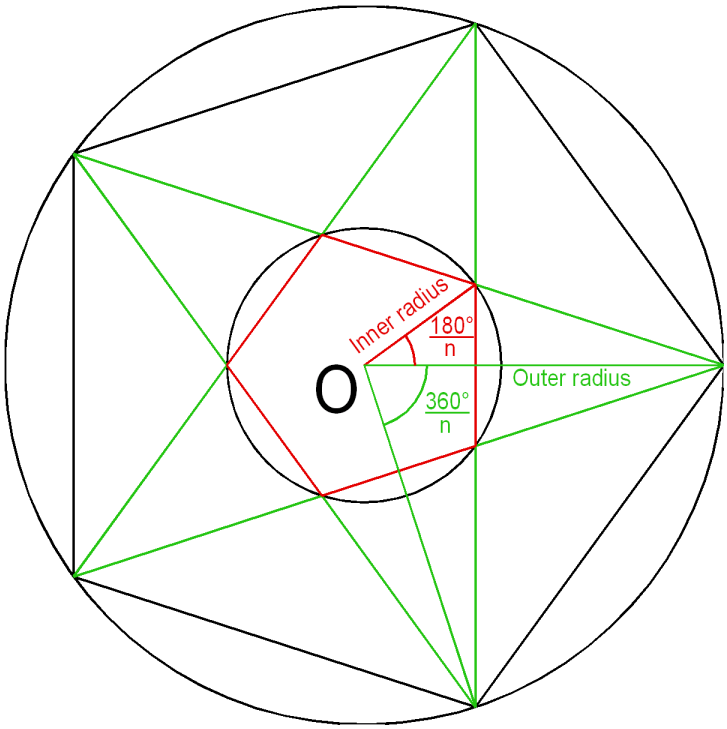
1 The STAR macro



Stars can often be found on maps or drawings. There's no CC3 built in star maker (but numerous symbols are star shaped).

1.1 Anatomy of a star

A regular star as those above is included in a regular polygon itself included in a circle. Inner points of the star are also included in a regular polygon of the same number of nodes but rotated at 180°/n (n is the number of nodes). The inner polygon is included in an inner circle :



So what do we need to define a star ?

i) Center point
ii) Outer radius
iii) Inner radius
iv) Number of nodes

Here we go :

```
MACRO Star1
SELSAVE
SAVESETTINGS
GOLAYER JDRSTARS
SHEET MYSTARS
GP pctr ^DClick center point
IFERR MacroDone
GN vnn 8
GN vnn ^DEnter number of nodes: (8)
IFZ vnn MacroDone
IFN vnn MacroDone
GV vor 10
GV vor ^DEnter outer radius: (10)
IFZ vor MacroDone
GV vir 2
GV vir ^DEnter inner radius: (2)
LWIDTH 0.2
COLOR 1
CIRR vor pctr;
COLOR 2
CIRR vir pctr;
:MacroDone
GETSETTINGS
SELREST
ENDM
```

pctr

point, center

vnn

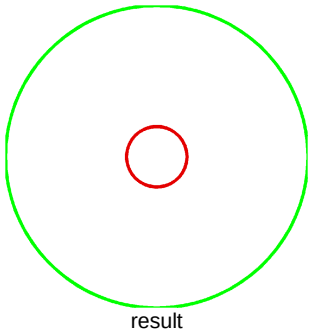
variable node number

vor

variable outer radius

vir

variable inner radius



result

To test that, I've added a new command : CIRR which is the standard CC3 Circle Radius and Center. Syntax is :

```
CIRR real point;
```

Remember part 2 : colors and linewidth are only for debug purpose.

1.2 Regular polygon

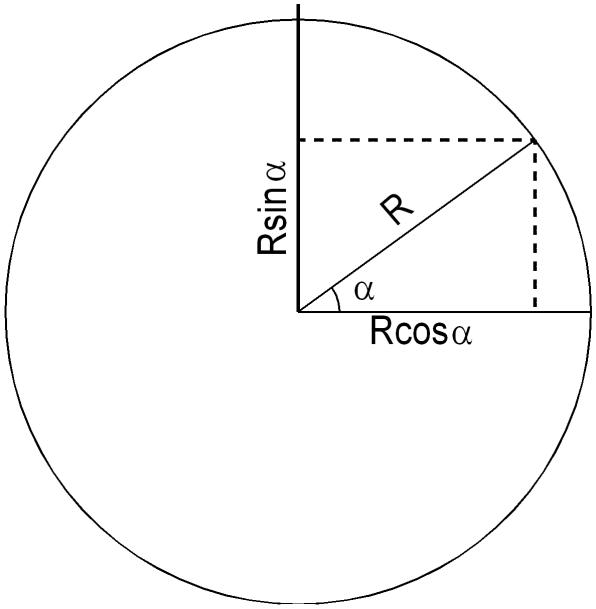
CC3 has already a regular polygon command, but that's not the point here. If we manage to draw a regular polygon of our own, drawing the star will only be one step away.

(math teatcher mode on)

What IS a regular polygon anyway ? Amongst various definitions, one could say that it is a polygon (surprise!) included in a circle so that all its points are at radius distance from the center point AND the angle defined by a point, the center and next point is always the same. In other terms, every point is the rotated last point by an angle of 360°/n at the center.

Ever heard the word *trigonometry* ?

A point on a R radius circle centered at 0,0 and at bearing α , has the x, y coordinates $R\cos\alpha, R\sin\alpha$.



(math teacher mode almost off)

How does CC3 computes cosine and sine ? Two new commands : GCOS and GSIN for GET COSINE and GET SINE. Syntax :

```
GCOS varName AngleInDegrees
```

To draw the polygon, we will need a loop from point 0 to point N. Each time we will have to draw a line from last point to next point. Each time, next point becomes last point so we have to store it. Look at that :

```
GV vfa 0
GETX vcx pctr
GETY vcy pctr
:Loop
GV vna vfa+360/vnn
GCOS vfx vfa
GSIN vfy vfa
GCOS vnx vna
GSIN vny vna
GP vfp vcx+vor*vfx,vcy+vor*vfy
GP vnp vcx+vor*vnx,vcy+vor*vny
GV vfa vna
COLOR 3
LINE vfp;vnp;
IFN vfa-359 Loop
```

(available as star2 in the stars.mac file)

note last line

```
IFN vfa-359 Loop
```

It means that if `vfa` is over 359, the loop ends. Why not 360 ?
Once again, we're dealing with rounded numbers.
If `vnn` is equal to 8 (default), the angle increment is 360⁄8 = 45°, a whole number. What if `vnn` is set to 7 ? 360⁄7 has no decimal value and must be rounded.

vcx,vcy coordinates of center point.

vfa,vna

variable first angle, variable next angle

vfp,vnp

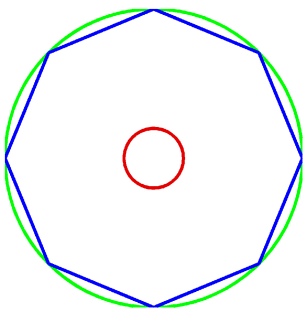
variable first/next point.

Why bother ?

Attentive readers might note that those cosine and sine could be prevented by

GP vfp ref pctr <vfa,vor

Well. my poor excuse is that it wouldn't have introduced new commands...



result

Give a try, edit star2, put 360 in instead of 359 and run the macro with 7 nodes. You will get a nice enough regular heptagon (7 nodes polygon) but if you look carefully or list the first line, you will notice that it is drawn twice. That could pose serious trouble if you want to multipoly your star !

The lesson is that when you write a macro, try to think to every possible case. A macro is not designed to be used only once, so severals thing might not work as planned if you didn't check everything...

1.2 The star !

Almost done! We just need to add the intermediates points on the inner circle.
They are computed like the others. Note that the angles made by those points are `(vfa+vna)/2`.

```
GCOS vix (vfa+vna)/2
GSIN viy (vfa+vna)/2
GP vip vcx+vir*vix,vcx+vir*viy
...
LINE vfp;vip;vnp;
...
```

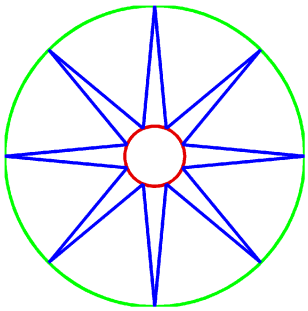
(see star3 macro)

Here we are! The star has been drawn, now we can get rid of the circles and colors that were only debugging tools.

vix, viy, vip

stands for variable intermediate x,y and point.

vip is of course also a very important point :)



result

1.3 The star #2

What if you want to fill the star ? You would have to select all the lines and then use a LINE TO PATH command.

CC3 should have a macro version of LINE TO PATH calles LTPM (Line To Path Macro). Couldn't found out how it works.

Standard LTP is not allowed in macros unless it's the last line. In that case, you will have to select all the lines anyway.

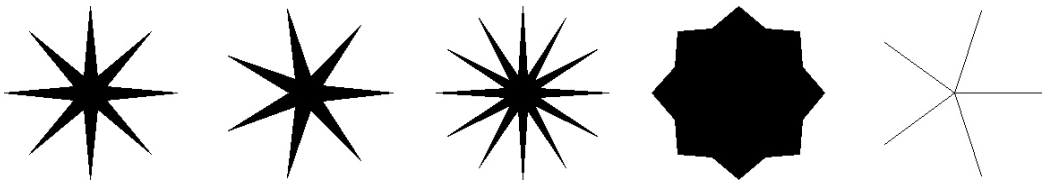
There is the last resort : MPOLY that yields a multipoly entity. Multipolies entities are tricky and should be used with care... as I found out the hard way.

```
GOLAYER TEMPSTARS
...
HIDEA
SELBYA
MPOLY
CHANGEL JDRSTARS
SHOWA
...
```

TEMPSTARS layer

Assuming you have more than one star to create in the same drawing, each one must be placed in a separate layer before multipoly-ing. Otherwise each new multipoly would include the other stars and with multipolies of multipolies... mama mia !

Therefore each star is drawn on the tempstars layer and moved to the JDR(vanity, vanity)STARS layer after multipoly-ing.



From left to right :

Nodes number	Outer radius	Inner radius
8	10	2
7	10	2
12	10	2
8	10	8
5	10	0*

* noticed I never tested Inner radius against 0 ?
Radiuses can even be negative !

That's the star4 macro. I've added something more to the star macro. Let's see if you can see what it is **before** lauching it!

2 Spirals

There are many ways of drawing a spiral and so many kinds of spirals.

2.1 Archimedes's spiral

The common one is Archimedes's spiral and kind of consist in a circle whose radius is linearly evergrowing.

In math terms, it is a parametric polar curve defined by $r = k.\theta$ with *k* a constant real number.

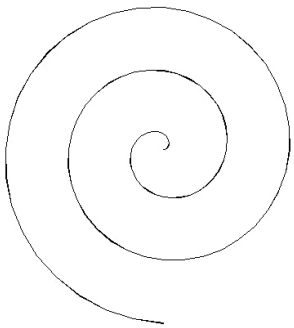
As the radius is everchanging, circles and arcs are not really helpfull here but we can try this:

```
MACRO Spir1
SELSAVE
SAVESETTINGS
GP pctr ^DClick center point
IFERR MacroDone
GV vk 1
GV vk ^DEnter k value: (1)
IFZ vk MacroDone
GV vin 100
GV vin ^DEnter iteration number: (100)
IFZ vin MacroDone
IFN vin MacroDone
GV vas 10
GV vas ^DEnter angle step: (10)
:MacroDone
GOLAYER JDRSPIRAL
SHEET MYSPIRAL
GV index 1
:Loop
```

Fatal error

There is a fatal error in this macro that made me abort CC3. Can you see it ?

Answer in the stars.mac file...



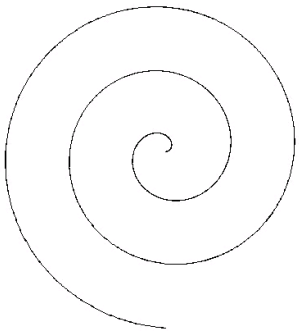
result (after debugging, that is)

```
GV vang vas*index
GP vfp ref pctr <vas*(index-1),vas*(index-1)*vk
GP vnp ref pctr <vas*index,vas*index*vk
LINE vfp;vnp;
IFN index-vin Loop
GETSETTINGS
SELREST
ENDM
```

The result is not bad, eh?
Two low marks, though :

i) there are 100 line entities here.

ii) it's ok at small scale but if you zoom a bit you will see straight parts.


result

ii) could be solved by using a SPLINE command and computing three points per loop :

```
GP vfp ref pctr <vas*(index-1),vk*vas*(index-1)
GP vip ref pctr <vas*(index-0.5),vk*vas*(index-0.5)
GP vnp ref pctr <vas*index,vk*vas*index
SPLINE vfp;vip;vnp;
```

Not much different at small scale and still a lot of entities.

2.2 False spirals

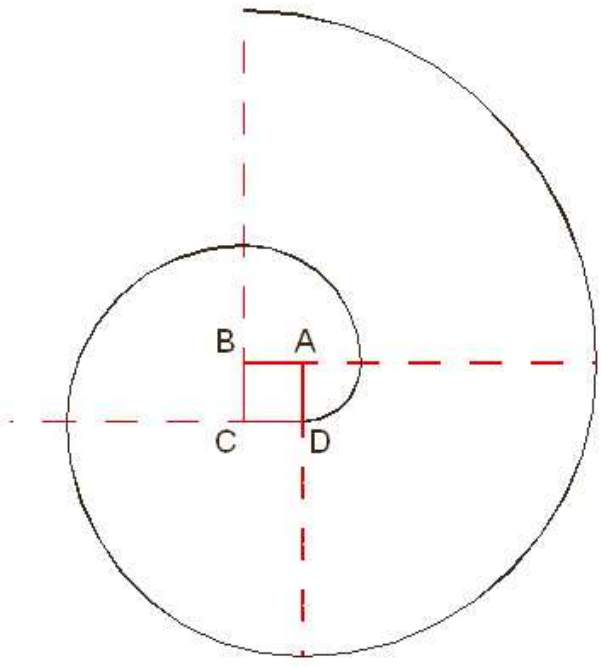
What is known as false spirals can be drawn with a calliper/divider and is made of arcs centered on a fixed number of points. With two points, arcs are half circles, four points quarter circles and so on. Points are generally arranged in a regular polygon but it's no absolute rule.

Here is a false spiral drawn from a regular box (square):

radius

Each radius is greater than the previous one as it should be.

The increment length is the side of the square.



First arc center is A point, begins at D point and stops after 90°. Second

arc center is B, third center is C, fourth D and fifth A again and so on.
This (false) spiral has only six entities !

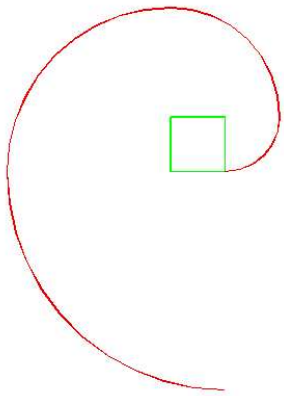
We need points A through D but it's easier to ask for the first point (ie D) and square size.

Let's begin small and place A, B, C and D first.

```
MACRO SPIR3
SELSAVE
SAVESETTINGS
GP vpd ^DClick on first point
IFERR MacroDone
GOLAYER JDRSPIRALS
SHEET MYSPIRALS
GV vsz 1
GV vsz ^DEnter size: (1)
GP vpa ref vpd <90,vsz
GP vpb ref vpa <180,vsz
GP vpc ref vpd <180,vsz
COLOR 1
LWIDTH 0.01
LINE vpa vpb vpc vpd vpa;
:MacroDone
GETSETTINGS
SELREST
ENDM
```

Got a green square ?

As it is better to work slowly, we'll only draw the first cycle

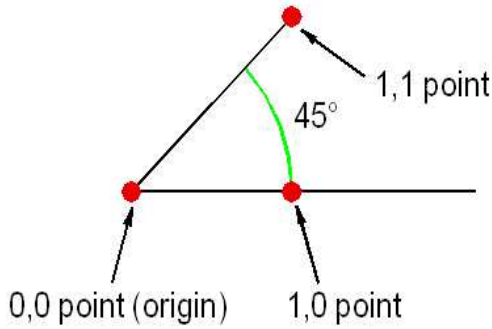


result

```
COLOR 2
GV vrad vsz
ARCS vpa;vpd;0
GP vspi ref vpa <0,vrad
ARCS vpb;vspi;90
GP vspi ref vpb <90,vrad+vsz
ARCS vpc;vspi;180
GP vspi ref vpc <180,vrad+2*vsz
ARCS vpd;vspi;270
GP vspi ref vpd <270,vrad+3*vsz
```

Here I introduce a new command ARCS which is the standard CC3 Arc, Center, Start, End.

This can be a bit confusing because Start is a Point Variable whereas End is either an angle, an not the angle of the arc but an angle relative to x axis, or a point in which case said point defines the angle an not the ending point :



To draw this arc we can use either

```
ARCS 0,0;1,0;45
```

or

```
ARCS 0,0;1,0;1,1
```

But point 1,1 cannot be reached because the radius was set to 1 by the first point one unit rigth to the origin.

The syntax for ARCS is then thus

```
ARCS centerpoint ; StartPoint ; Angle relative to x axis
ARCS centerpoint ; StartPoint ; Point defining angle
```

That being said, we can look at the rest.

The radius is first set to the size of the square (`vsz`) as it should be and each new arc radius is incremented by that radius.

Before drawing next arc we need to know where the last ended, that's `vspi`.

Everything is solved by angles relative to xaxis (bearings) and every step our angles increment by 90° because the spiral is b ased on a square.

Macro is spir4

We can now start the loop. We need to ask how many times the loop will occur :

```
...
GN vin 10
GN vin ^DEnter number of iteration
IFZ vin MacroDone
...
```

Before starting the loop we must initialize `vspi` and the `index`

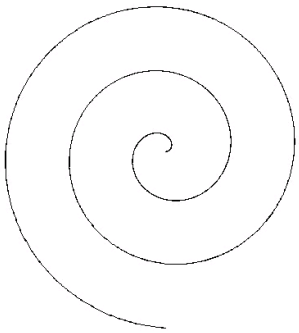
```
...
GP vspi vpd
GN index 1
:Loop
ARCS vpa;vspi;0
...
```

After a cycle (loop), we must increment the radius four times because we draw 4 arcs each loop and never, never, never forget to increment `index`

```
GV vrad vrad+4*vsz
GN index index+1
IFN index-vin Loop
```

The final macro is SPIR. Enjoy.

Left : size 1 Iterations 5 : 20 entities only!
Center : size 0.5 Iterations 10 : 40 entities
Right : size 0.3 Iterations 15 : 60 entities!



Result of Archimedes's spiral for comparison (100 entities)

3 Rotated boxes

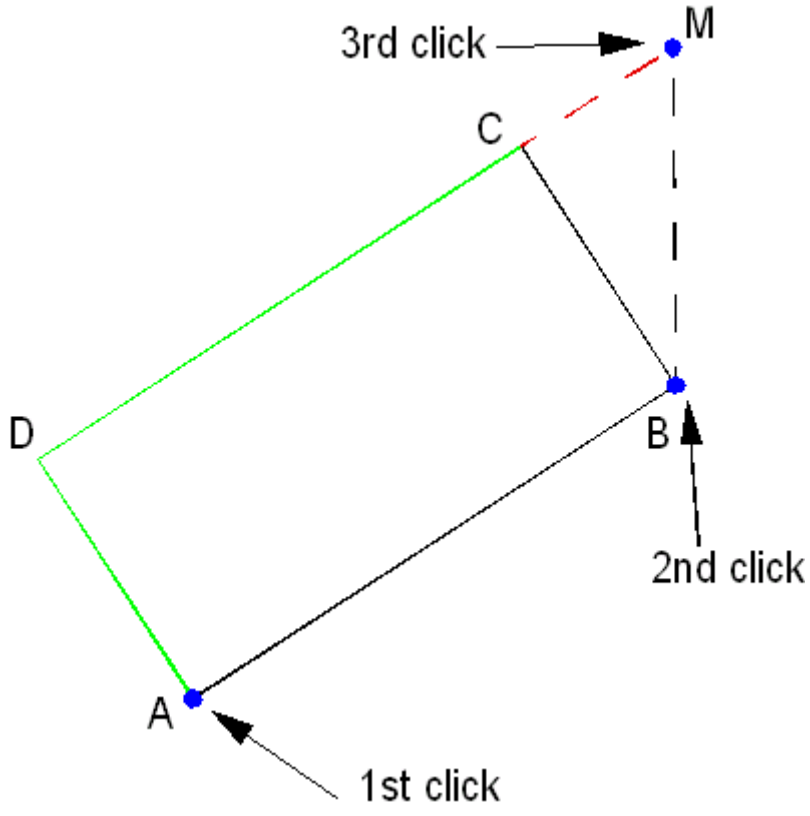
Never wanted to draw a rotated box?
When CD3 came out I marvelled at that beautifull tool that lets you place box shaped houses by clicking three points.

That's what I propose to do with standard boxes.

The trouble is that with macros you don't see what you get as you move your cursor. Anyway, even if the utility might be reduced, let's make it a macro writing exercice.

It will go like that :

1. First click : first point
2. Second click : second point, thus defining one of the sides.
3. Third click : a point not necessarily on the box but used to close the box anyway like this :



It will look a lot like the door macro of part 2 because first thing will be to find C point.

Steps will be

1. Entering A, B and M points.
2. Finding C
3. « Moving » C to get D.

3.1 Starting the macro

```
MACRO RBOX1
SELSAVE
SAVESETTINGS
GP pA ^DClick on first point
IFERR MacroDone
GP pB ^DClick on second point
IFERR MacroDone
GP pM ^DClick on third point
IFERR MacroDone
GDIST vAB pA pB
IFZ vAB MacroDone
GDIST vBM pB pM
IFZ vBM MacroDone
GDIST vAM pA pM
IFZ vAM MacroDone
LWIDTH 1
COLOR 1
LINE pA pB;
:MacroDone
GETSETTINGS
SELREST
ENDM
```

I think we've taken care of anything that could go wrong but the remote possibility that M could be on (AB) line. That will be done after next step.

3.2 Computing C

C is THE point found on the parallele line to (AB) going trough M AND on the perpendicular line to (AB) going through B. Thus :

(math mode on)

$\overrightarrow{MC} = k\,\overrightarrow{AB}$ (1) (MC) is parallel to (AB)
$\overrightarrow{BC}.\overrightarrow{AB} = 0$ (2) (BC) is perpendicular to (AB)

Analytical or geometrical analysis yields the following (for details, see part 2 where a similar work has been done)

$$\overrightarrow{MC} = (\overrightarrow{MB}.\frac{\overrightarrow{AB}}{AB}).\frac{\overrightarrow{AB}}{AB}$$

Because CC3 doesn't know vectorial operations (here we have a dot product, a vector divided by it's length and a vector times a number also known as a scalar) we have to go analytical after all as we need

separate coordinates of A, B and M along with the length of AB.

```
GETX vxa pA
GETY vya pA
GETX vxb pB
GETY vyb pB
GETX vxm pM
GETY vym pM*
```

Let's do the dot product :

```
GV vdot ((vxb-vxm)*(vxb-vxa)+(vyb-vym)*(vyb-vya))/vAB
```

vAB

Is already defined as AB length.

And compute C point

```
GV vxc vxm+vdot*(vxb-vxa)/vAB
GV vyc vym+vdot*(vyb-vya)/vAB
GP pC vxc,vyc
```

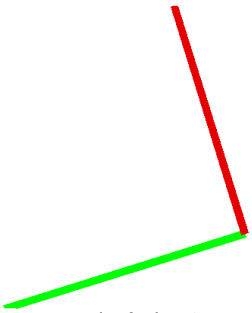
$$x_C = x_M + (\overrightarrow{MB}.\frac{\overrightarrow{AB}}{AB}).(x_B - x_A)$$

Same for ys....

Some debugging trick

```
COLOR 2
LINE pB pC;
```

And macro RBOX2 is done!



Result of Rbox2

3.3 So what about M point being on the (AB) line ?

As the dot product of two vectors tells us if vectors are perpendicular when it yields zero, another tool will tell us if vectors are parallel : the cross product.

Now, the cross product is generally used in 3D because the cross product of two vectors in another vector, perpendicular to the plane defined by the firts two and if those two are parallel (we say then colinear), no plan is defined.

That won't trouble us because we're interested in the length of the cross vector, not the vector itself.

If we go analytical again in the $z=0$ plane (which is the standard 2D) cross product of $x,y,0$ vector by $x',y',0$ vector has a signed length of $x.y'-y.x$.

We're interested by M, A and B. If M point is on (AB) line then $\overrightarrow{AM}$ and $\overrightarrow{AB}$ are colinear.

```
GV vcross (vxb-vxa)*(vym-vya)-(vyb-vya)*(vxm-vxa)
IFP vcross AbsDone
GV vcross -vcross
:AbsDone
IFN vcross-vAB/1000 Mabort
...
GO MacroDone
:Mabort
MSGBOX Macro Aborted
Because third point is on line
defined by the two firsts
(or almost)
```

```
:MacroDone
...
```

Why bother at all ? Having M point on (AB) line won't yield an error because unlike the dot product in part2 if the cross product length isn't involved in a division.

Well, I got this traumatic case with multipolies. It is quite right to say that if M point is on (AB) line CC3 won't crash or show any math error BUT you will have two entities one on another and that's never a good thing...

Modified macro is Rbox3. Note I moved the drawing of (AB) line after the test...

3.4 D point

D point is the result of moving C point by $\overrightarrow{BA}$ (and not $\overrightarrow{AB}$ eh? Even if it sounds better!)
Thus

$x_D = x_C + (x_A - x_B)$ and $y_D = y_C + (y_A - y_B)$ (brackets only academic).

```
GV vxd vxc+vxa-vxb
GV vyd vyc+vya-vyb
GP pD vxd,vyd
GOLAYER BOXES
SHEET JDRBOXES
GOLAYER TEMPBOX
HIDEA
LINE pA pB pC pD pA;
SELBYA
MPOLY
CHANGEL BOXES
SHOWA
```

LINE pA pB PC pD pA;

One common pitfall when drawing quadrangular shapes on a computer is to forget to close the shape. Therefore, even if there are only four points, first point must be repeated.

MPOLY

Even if we draw all four lines with only one commands, they still stays four entities vs standard box which is one.

3.5 Last embellishment

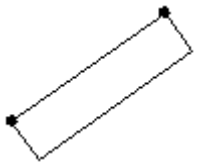
We cannot draw « phantom » lines as CC3 does while drawing shapes so the placement of A, B and M points might be hazardous.

What we can do is visualise those point by drawing some circles around them. Which radius to choose ?

We can try this : 1% of drawing width. This can be accessed with the GETEXTX for GET EXTents width (and it's fellow command GETEXTY yields the heigth of the drawing).

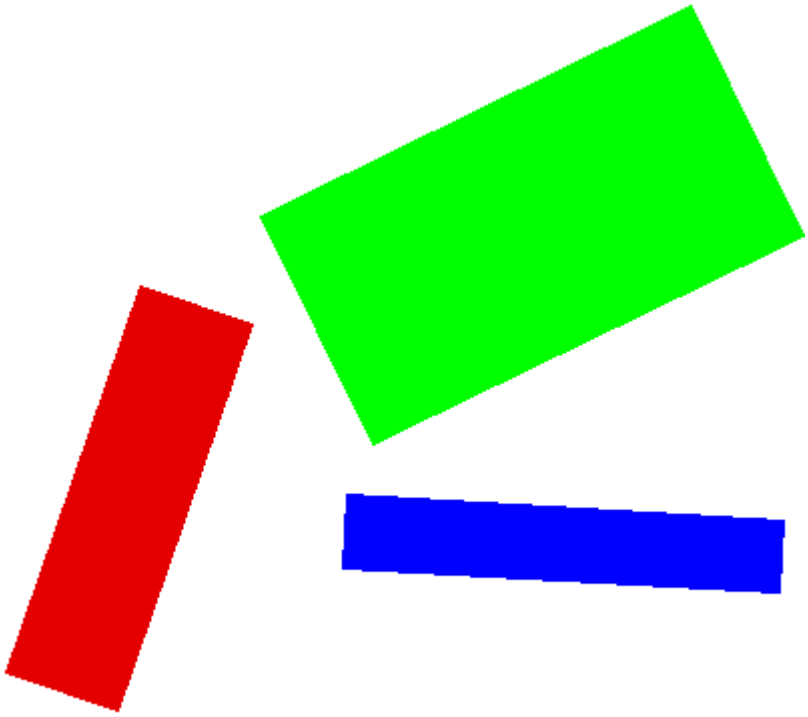
```
GETEXTX vdrw
GV vrad vdrw/100
GOLAYER TOERASE
CIRR vrad pA
...
:MacroDone
LAYER TOERASE
HIDEA
SELBYA
ERA
SHOWA
LAYER BOXES
```

There's no need to plot M point...



Intermediate result (use CC3 UNDO to get this after rbox macro)

Don't forget to put those circles on a specific layer to be erased at macro completion.



New commands			
Command	**Effect**	**Syntax**	**Page**
CIRR	Draws a circle defined by a radius and a center point	`CIRR radius point;`	2
GCOS	Stores the cosine of an angle in degrees in a variable	`GCOS varName AngleInDegrees`	3
GSIN	Stores the sine of an angle in degrees in a variable	`GSIN varName AngleInDegrees`	3
MPOLY	Creates a multipoly out of selected entites	`MPOLY`	4
SPLINE	Draws a smooth line	`SPLINE point1 point2 ..... point N;`	6
ARCS	Draws an arc defined by center point, start point and end bearing	`ARCS centerPoint StartPoint BearingInDegrees` `ARCS centerPoint StartPoint PointDefiningAngle`	7
GETEXTX	Stores the width of your drawing in a variable	`GETEXTX varName`	12
GETEXTY	Stores the heigth of your drawing in a variable	`GETEXTY varName`	12

Part 3 is now ended.

Thanks to every macro writers! Your examples were invaluable.

Special thanks to Linda Kekumu, Allyn Bowker, Simon Rodgers, & Ralf Schemmann for their help, either through the TOUM, the Technical Support or the PF forum.