CC3 Macro

What I learned writing macros

Introduction

Writing macros for CC3 can be quite confusing, especially if you're famillier with standard programming.

Macros can however yield marvels and make your map mapping easier.

I'm no expert but I found out some things I'd like to share.

1. Macro file and first macro.

A CC3 macro file is essentially a text file with a .mac extension.
If you navigate in your CC3 folder, you will see a FCW32.mac file. It contains all the *system* macros CC3 already uses as tools.
Please never, never, never modify this file unless you're really sure of what you do.

Open a text editor such as Notepad.
Type in

```
MACRO MYTEST1
GL varSname ^DEnter your name
GP varPname ^DClick a point
TEXTM varSname;varPname
ENDM


```

Warning : those blank lines at the end are vital ! They ensure that the macro is recognised as finished.

Save as *mytest.mac* in the CC3 folder.
No need to close Notepad.

Congratulations, you've created your first macro.

Open CC3, use the LOADMAC command and select *mytest.mac* then type « mytest1 » in the command line.
CC3 asks your name and to click on the map. No big surprise, your name appears at the selected point with the current text properties.

Let's look at those lines

1. `MACRO MYTEST1`
The command MACRO says : hey, this is a macro ! MYTEST is the name of the macro and also the command to launch it. You can write it in capitals or not. MACRO mytest1 will yield the same.

Modifying the FCW32.mac file

Once you have designed a macro and are happy with it, you can add it to the FCW32.mac file provided no other macro shares the same name.

This makes the macro available at runtime without using the loadmac command.

Always keep a separate copy of your macros, though. FCW32.mac may be replaced while updating/upgrading.

Blank lines

Blank lines at the end of a macro are vital : they ensure that the macro is aknowledged as finished.

Blank lines in the macro are sometimes required for some commands as an ending code but **MUST NOT** be used otherwise.

```
MACRO MYTEST

GL varSname ^DName
...
```

Won't work. In fact, I got a crash.

Space

The Space caracter is sometimes used as a separator but often errors occur when there is one at the end of a line, so check...

Variable names

Variable names can be almost anything. Common sense rules however that they aren't too long but clear enough to outside party.

TheLineContainingYourName is obviously too long but *S* could be hard to understand.

2. `GL varSname ^DEnter your name`
GL is a command to store a string of caracters (generally known as a string in most programming languages). I think GL stands for Get Line.

varSname is the name of the variable were the string will be saved. It's up to you to choose the name of the variable. It can contain figures but not at the begining. GL L1 is correct. GL 1L is not.

^D is the code for a prompt, that is, something to allow the user to interact with the macro.

Enter your name is the string the prompt will show. It's up to you to choose the prompt text.

In long macros where a lot of variables occur be sure to name them efficiently. When you'll have to scroll up your text to search for the name of this #### variable, you'll understand.

3. `GP varPname ^DClick a point`
GP is the command to store a point variable, which will save both the X and Y coordinates in a single variable. It stands for Get Point.

varPname is the name of the variable. P would have been enough.

^D is again for the prompt with the line *Click a point*.

The same macro

```
MACRO mytest1
GL L1 ^DName, please:
GP P ^DClick
TEXTM L1;P
ENDM
```

4. `TEXTM varPname;varPname`
TEXTM is the macro command version of TEXT. If you forget the ending « M », CC3 will launch a standard TEXT command with dialog and all, and an error will occur because the rest of the line would not be taken in account.

varSname is the variable containing the answer and varPname is the one containing the coordinates of the target point.
The semicolon between the two variables is important here. Otherwise, CC3 will print « varSname varPname » instead of your name.

5. `ENDM`
ENDM stands for End Macro. It means of course that the macro is ended but be sure to put a carriage return after this line (see sidebar page 1). Most macro writers add more blank lines after ENDM to be sure, and also to separate clearly different macros in the same .mac file.

2. IFERR and labels
Now let's say you want to abort the macro after launching it. The usual way is to rigth-click.

Have a try, right-click after the first prompt.

The macro doesn't stop...
The solution is the IFERR command which stands for IF an ERRor occurs then...

mytest.mac/.txt

The file containing all the macros shown in this article is available for download as mytest.txt because the PF forum doesn't accept .mac files (something to do with MacIntosh computers I guess).

You will have to rename the file *mytest.mac* and place it in the CC3 folder.

You will see that I have included the ZOOM IN and ZOOM OUT macros from FCW32.mac because it is sometines frustating not being able to use those commands while debugging macros...

Aborting a macro is essentially skipping all the commands. To do that, the macro must « jump » over unwanted parts. It's the GO command. To show where you want to jump to, you need to *label* the macro text. A label is indicated by a colon followed by a label name. Try this :

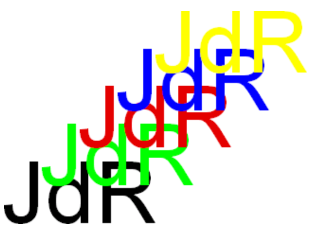
```
MACRO MYTEST2
GL varSname ^DEnter your name
IFERR MacroDone
GP varPname ^DClick a point
TEXTM varSname;varPname
:MacroDone
ENDM
```

No colon is required in the IFERR MacroDone line. In fact, you *must* not write the colon there. CC3 sort of reads the whole macro and knows that the :MacroDone label comes after.

3. Loops

In most programming, loops are achieved by a *for index=0 to max* or *while condition.*
CC3 doesn't understand that so you will have to simulate it.
An index variable is generally needed, so you must initialise one, and test when the number of occurances has been reached.
Looping is going backwards to repeat again and again the same kind of commands. Therefore you must insert a label at the beginning of the loop and one at the end. Here is an example :

```
MACRO MYTEST3
GL varSname ^DEnter your name:
IFERR MacroDone
GP varPname ^DClick point:
IFERR MacroDone
GN Index 0
:Loop
IFZ Index-5 LoopDone
COLOR Index
GP varP2 ref varPname @5*Index,5*Index
TEXTM varSname;varP2
GN Index Index+1
GO Loop
:LoopDone
:MacroDone
ENDM
```



Result with text heigth set to 10, line width to 0 and fill style to solid.

New revelant lines are

6. `GN Index 0`
GN is the Get Number command which needs a variable name : Index, and an integer value : 0. This line initialises the index variable used to test the end of the loop.

7. `:Loop`
This is the label at the beginning of the loop. One common error is to

GN and GV

Till now I only used whole numbers (i.e. integers). To set a real number (whole or decimal) you need the GV : Get Value command.

Note that a GN used on a real number will yield only the integer part, ie the number rounded down. For example

GN 15/8 will yield 1 even if it's almost 2.

place the label before index initialisation. In that case the index number will each time be at 0 and the loop will have no end.

8. `IFZ Index-5 LoopDone`
IFZ means if zero. Zero will occur in the following Index-5 when Index is equal to 5, thus making the loop occur five times. After an IFZ and its test there is a label. No need to add GO because an IFwhatever is always linked to a label. GO is implicit.

IFZ, IFN, IFP

IFZ test if the result is equal to 0 (Zero), IFN is it is less than 0 (Negative value) and IFP if it is more than 0 (Positive value)

9. `COLOR Index`
Not a really revelant line (yet!), juste changes the color of the text after each iteration.

10. `GP varP2 ref varPname @5*Index,5*Index`
GP is our old Get Point command with a variable I names varP2

`ref` is a code saying that following modifications are applied relative to the varPname point. Without it, each point would have been relative to the last point used.

`@` is the usual CC3 relative move/copy. Without that, all the points would have been relative to the origin though the ref indicator.

This line thus creates a second point varP2 that lies 5*Index units to the rigtht and 5*Index units above the clicked varPname point.

Feel free to try it without ref varPname and without @.

@

I use @ a lot while drawing maps or dungeons.

If I want to copy an entity 5 units to the right and 2 units above, I select whatever point is available for copy origin and then type @5,2 for copy two.

On the other hand, I almost as often use 0,0 for the origin, and 5,2 (without @ but it doesn't matter) for copy two even if those two points are far away a zoomed part.

12. `GN Index Index+1`
VERY important line. Without it, your index will obviously always stay at 0 and you're loop will be never ending...

Index = Index +1

This was an old debate amongst programmers. The whole point was that Index could never mathematically be equal to index+1, thus creating tricks to avoid writing that.

What it meant was that the variable Index is incremented, by one ie assigned it's value plus one.

13. `GO Loop`
Moves backward to the begining of the loop.

14. `:LoopDone`
Label indicating where to jump once the loop has reach it's limit.

Note that here it is not really important, I could have jumped to MacroDone instead.

Some comments about loops

- the test to end the loop can occur just after the beginning of the loop, just before the GO Loop or even in the middle if you want part of the loop done at the end.
- some loops don't need an index or have an index calculated by the macro. In that case, add anyway a foolproof index in case of. When writing a macro I often use a lot of trials and errors. Due to an error, I made a macro once place 10000+ symbols on a map. In that case, put a limit such as (both tests)
  ```
  IFZ whateverTest EndLoop
  IFZ index-limit EndLoop
  ```

Trials and errors

Thats the way I learn macro writing. You might however need to refresh your hair style after the session though :))

4 Leaving things as they were...

You will notice that after completion, last macro leaves the color yellow (or color 4 if you modified your palette).
The next thing you'll draw will be yellow unless you place a non varicolor symbol.

It's of course easy to click on the color square and changing it but it's even better to make the macro do it.

Two commands are handy here : SAVESETTINGS and GETSETTINGS

SAVESETTINGS stores the current settings and GETSETTINGS restores them to what SAVESETTINGS stored.

Try this :

```
MACRO MYTEST4
SAVESETTINGS
GL varSage ^DEnter your name:
IFERR MacroDone
GP varPage ^DClick point:
IFERR MacroDone
GN Index 0
:Loop
IFZ Index-5 LoopDone
COLOR Index
GP varP2 ref varPage @5*Index,5*Index
TEXTM varSage;varP2
GN Index Index+1
GO Loop
:LoopDone
:MacroDone
GETSETTINGS
ENDM
```

(first change the color, of course, since it would be color 4 otherwise).

5 GE

Till now we've worked with string and number variables. In a CC3 file you work a lot with entities and that's where we are : the GE stores an entity in a variable. Try this after drawing a smooth line :

```
MACRO MYLINES1
SAVESETTINGS
GE varEnt ^DSelect entity
IFERR MacroDone
GOLAYER MYLINES
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
GP varPperp ref varPOnEnt <90,10
LINE varPOnEnt varPperp;
GN varPerc varPerc+10
IFZ varPerc-100 EndLoop
GO Loop
```

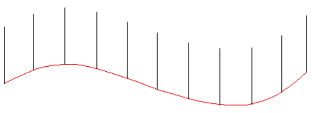
GE

To be honest, GE doesn't select an entity. It selects a point an refers to the entity at that point. The subtility will be explain later.

Fractal entities

Fractals are great tools and I love to use them.

A fractal entity does however not exist as such here because fractals consist in a bunch of lines. Selecting a fractal here would only select the particular line at selection point.


Result

```
:EndLoop
GP VarPOnEnt % 100 varEnt
GP VarPperp ref varPonEnt <90,10
LINE VarPOnEnt VarPperp
:MacroDone
GETSETTINGS
ENDM
```

3. `GE varEnt ^DSelect entity`
That's the entity selection stored in a varEnt variable.

5. `GOLAYER MYLINES`
This a new command GOLAYER that creates an new Layer and makes it current layer. It's easier to delete the lines after if they don't please you.

GOLAYER and LAYER

There is a difference between those commands.

LAYER sets the current layer amongst **existing layer**. Thus using LAYER MYLINES if the MYLINES layer does not exist will yield an error.

GOLAYER creates the layer if it does not exist AND sets it as the current layer.

8. `GP varPOnEnt % varPerc varEnt`
Here it's time to use the entity stored in varEnt. The GP gets a point named varPOnEnt (point on entity) defined par the % along modifier. After the % you see first the varPerc variable that gives the percentage and at the end only the entity variable. syntax is :

```
GP PointVariable % PercentageAmount EntityVariable
```

9. `GP varPperp ref varPOnEnt <90,10`
I use again the *ref* saying that the following is relative to the point taken on the entity. The < is the usual CC3 indicator for polar coordinates. <90,10 means a point at 90° (vertical, above) and 1 0 units from the reference point. Note that's not 90°to the entity.

10. `LINE varPOnEnt varPperp;`
This is the common LINE command. It is followed by any numbers of points but must be closed by a semicolon (;) otherwise you will be ask for further points.

The last line is drawn after the loop. I could have set the test to `IFZ varPerc-110` instead but that's not the point.

6 GBRNG

Now let's say we don't want those lines vertical, but normal to a smooth line (SPLINE).
To achieve that, we need to know the line tangent to the curve, or the angle (bearing) this tangent makes versus x axis (0°for CC3).
GBRNG (Get Bearing) does almost that. This command stores the angle made by a line defined by two points.

Here's the new macro next page :

```
MACRO MYLINES2
SAVESETTINGS
GE varEnt ^DSelect entity
IFERR MacroDone
GOLAYER MYLINES
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
IFZ varPerc NoBefore
GP vBefore % varPerc-0.1 varEnt
GO EndBefore
:NoBefore
GP VBefore % 0 VarEnt
:EndBefore
GP vAfter % varPerc+0.1 varEnt
GBRNG varBrng vBefore vAfter
GP varPperp ref varPOnEnt <varBrng+90,10
LINE varPOnEnt varPperp;
GN varPerc varPerc+10
IFZ varPerc-100 EndLoop
GO Loop
:EndLoop
GP vBefore % 99.9 varEnt
GP vAfter % 100 varEnt
GP VarPOnEnt % 100 varEnt
GBRNG varBrng vBefore vAfter
GP VarPperp ref varPonEnt <varBrng+90,10
LINE VarPOnEnt VarPperp
:MacroDone
GETSETTINGS
ENDM
```

A bit of math...

The tangent of the angle made by a tangent to a curve is the limit of $\frac{y2-y1}{x2-x1}$ where (*x1,y1*) and (*x2,y2*) are the coordinates of two points running along the curve toward the target point.

To simulate that we choose two point near the POnEnt, one a 0,1% less, the over at 0,1% more

However the first point has no *before* point and the last has no *after* point.

9.`IFZ varPerc NoBefore`
If varPerc is equal to zero, there's no point on the entity before, so special care must be taken. In that case we make the macro jump after assigning that point.

10. `GP vBefore % varPerc-0.1 varEnt`
11. `GO EndBefore`
Standard points get a point slightly (0,1%) before. Jump over the NoBefore block.

12. `:NoBefore`
13.`GP VBefore % 0 VarEnt`
When there is no point before, the macro chooses the first point instead.

14.`:EndBefore`
15. `GP vAfter % varPerc+0.1 varEnt`
The point before has been assigned in every case so the macro continues.
Because the last point is treated outside the macro, there's no need to test for that here so the point after is assigned directly (0.1% again).

16. `GBRNG varBrng vBefore vAfter`
Now we use that new command GBRNG. Like all the GET command, it is followed by the name of the new variable : varBrng. The syntax wants then two points : the one just before and the one after.

17. `GP varPperp ref varPOnEnt <varBrng+90,10`
This line has been modified in ordre to take the angle (varBrng) in. varBrng+90 is the normal (perpendicular) line.

22. `:EndLoop`
23. `GP vBefore % 99.9 varEnt`
24. `GP vAfter % 100 varEnt`
As I mentionned above, the last point is not treated in the loop. The last point has no *after* so it is choosen instead, as was the case for the *before* of the first point.

You can change the length of those line by adding for example the following between lines 16 and 17 :

```
GV varLmodif 50-varPerc
IFP varLmodif Absolute
GV varLmodif -varLmodif
:Absolute
```

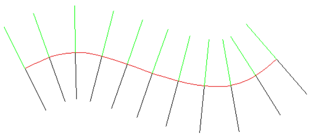
and modifying lines 17 and 27 thus :

17. `GP varPperp ref varPOnEnt <varBrng+90,10*(1-varLmodif/100)`
27. `GP VarPperp ref varPonEnt <varBrng+90,5`

+/- 90

Adding 90°to an angle yield the perpendicular counter-clockwise. -90° would have given the other side. The macro does not ask for a choice but the end of the entity selected will change the order of the points as is always the case with the % modifier.

The point where you click to select the entity will then force the way the lines are drawn.



Result : in green clicking left, in black cliking right

More math

The absolute value of a number is unsigned. If the value is less the 0, it's absolute is the opposite number ie -value.

Here I use a new variable varLmodif that looks how far varPerc is to 50 and take it's absolute, getting a value between 0 (at 50%) and 50 (at 0% and 100%).

1-varLmodif/100 is then between 0.5 (at 0 and 100%) and 1 (at 50%) making the perpendicular lines shorter at both end and following a linear gradient to a maximum at 50%

7 RANDOM

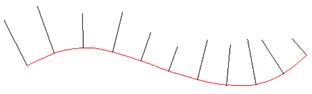
Mappers like randomness. The RANDOM command yields a, err random, number between 0 and 1 and stores it in a real variable. The syntax is :

```
RANDOM VariableName
```

Now instead of a linear gradient as above, this new macro makes all lines ramdon in length, from 2 to 10 :

```
MACRO MYLINES3
SAVESETTINGS
GE varEnt ^DSelect entity
IFERR MacroDone
GOLAYER MYLINES
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
IFZ varPerc NoBefore
GP vBefore % varPerc-0.1 varEnt
GO Before
:NoBefore
GP VBefore % 0 VarEnt
:Before
RANDOM varLmodif
GP vAfter % varPerc+0.1 varEnt
```

```
GBRNG varBrng vBefore vAfter
GP varPperp ref varPOnEnt <varBrng+90,10*(0.2+0.8*varLmodif)
LINE varPOnEnt varPperp;
GN varPerc varPerc+10
IFZ varPerc-100 EndLoop
GO Loop
:EndLoop
GP vBefore % 99.9 varEnt
GP vAfter % 100 varEnt
GP VarPOnEnt % 100 varEnt
GBRNG varBrng vBefore vAfter
RANDOM varLmodif
GP VarPperp ref varPonEnt <varBrng+90,10*(0.2+0.8*varLmodif)
LINE VarPOnEnt VarPperp
:MacroDone
GETSETTINGS
ENDM
```


Result

10*(0.2+0.8*varLmodif)

varLmofif has been generated by a RANDOM command, thus it is between 0 and 1.

It follows that 0.8*varLmodif is between 0 and 0.8. 0.2+0,8*varLmodif is then between 0.2 and 1.

This is a trick to get a random number between something else than 0 and 1.

10 times that yields then a number between 2 and 10 because I didn't want my lines to be to small

8 Multiple parameters and default values.

The last macros draws only 11 lines because I use an increment of 10% by loop and go from 0% to 100%.

It's a bit limitative. The macro could ask how many lines I want to draw, how long, random or not and so on.

It's of course possible but think from the other side : each time you will use the macro, you will have to type all those parameters.

The first solution is to use default parameters and here it's how we do it :

```
GV vLinesNumb 11
GV vLinesNumb ^DEnter number of side lines: (11)
```

It seems that the macro assigns two times the same variable, but that's only the case if you answer the prompt. Hitting return without typing a number first won't modify the default value of 11.

Now that this number is variable, there is something else to change : the amount to increase the percentage after each loop.

Here is the modified macro :

```
MACRO MYLINES4
SAVESETTINGS
GE varEnt ^DSelect entity
IFERR MacroDone
GV vLineNumb 11
GV vLineNumb ^DENter number of side lines: (11)
IFZ vLineNumb-1 MacroDone
GV vPercInc 100/(vLineNumb-1)
GN vLineLimit 100
GN vLoopNumb 0
GOLAYER MYLINES
GV varPerc 0
```

Order of entering

Note that I have inserted the number prompt after the entity prompt. It's not really important, but that way is you mis-click the entity (never happened ?) you won't have to type all the other parameters.

When there is one, I always ask for the entity before anything else.

```
:Loop
GP varPOnEnt % varPerc varEnt
IFZ varPerc NoBefore
GP vBefore % varPerc-0.1 varEnt
GO Before
:NoBefore
GP VBefore % 0 VarEnt
:Before
RANDOM varLmodif
GP vAfter % varPerc+0.1 varEnt
GBRNG varBrng vBefore vAfter
GP varPperp ref varPOnEnt <varBrng+90,10*(0.2+0.8*varLmodif)
LINE varPOnEnt varPperp;
GV varPerc varPerc+vPercInc
GN vLoopNumb vLoopNumb+1
IFZ vLoopNumb-vLineLimit MacroDone
IFP varPerc-(100-vPerInc/2) EndLoop
GO Loop
:EndLoop
GP vBefore % 99.9 varEnt
GP vAfter % 100 varEnt
GP VarPOnEnt % 100 varEnt
GBRNG varBrng vBefore vAfter
RANDOM varLmodif
GP VarPperp ref varPonEnt <varBrng+90,10*(0.2+0.8*varLmodif)
LINE VarPOnEnt VarPperp
:MacroDone
GETSETTINGS
ENDM
```

Fool proof limit

Remember the case of long ending macros page 4 ? vLoopNumb is the variable created for that. Of course, if you think you might draw over 1000 lines, change the limit, it's your time :)

Note the changes :

All the GN have been swaped by GV regarding the % values. That's because now that the number of lines can be anything, nothing ensures that the % step will be an integer.

For the same reason, the `IFZ varPerc-100` has been changed by a `IFP varPerc-(100-vPerInc/2)` because due to rounding, 100 might never be exactly reached and the macro will never end (but for the fool proof index, see sidebar). The `100-vPerInc/2` ensures that no line will appear almost at the end (such as 99.99% for example).

There is also a test at the begining checking that there is more than one line. Drawing a single line at the begining of an entity should not involve a macro ! And of course, entering 1 would lead to a divide by zero error the line after.

ECOFF

This command hides the following command that till now appeared in the command line.

the ASKBOX

Blank line again

I have inserted a blank line here that will cause no crash.

It says to CC3 : here is the end of the ASKBOX.

Otherwise, the whole macro body following would have been shown in the box...

9 ASKBOX and MSGBOX

As I said before, if you have a lot of parameters it may take more time to enter those than finishing the rest of the macro (it still less time consuming than doing everything without the macro!)

To shorten even more the delay you can ask only once if all the default parameters should be used.

This implies of course that it is a situation where this would be a general case.

An ASKBOX can do that. The ASKBOX command displays a box containing up to several lines and two buttons : Yes and No. Clicking NO will emit an internal message IFERR responds to.

A MSGBOX, for MeSsaGeBox, is almost the same, but there is only one button. It is of course used to display a message.

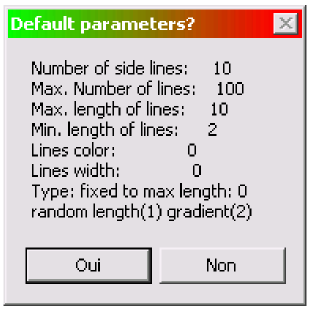
Here is a last example :

```
MACRO MYLINES5
SAVESETTINGS
ECOFF
GE varEnt ^DSelect entity
IFERR MacroDone
GV vLineNumb 11
GN vLineLimit 100
GV vLineWidth 0
GV vmaxL 10
GV vminL 2
GV vcolor 0
GN vtype 0
ASKBOX Default parameters?
Number of side lines:     10
Max. Number of lines:    100
Max. length of lines:     10
Min. length of lines:      2
Lines color:               0
Lines width:               0
Type: fixed to max length: 0
random length(1) gradient(2)

IFERR Choose
GO EndChoose
:Choose
GV vLineNumb ^DENter number of side lines: (11)
IFZ vLineNumb-1 OneLine
GN vLineLimit ^DEnter max. number of lines: (100)
GV vmaxL ^DEnter max. length of lines: (10)
GV vminL ^DEnter min. length of lines: (2)
IFN vmaxL-vminL MaxMinErr
GN vcolor ^DEnter color of lines: (0)
GV vLineWidth ^DEnter lines width: (0)
IFZ vmaxL-vminL EndChoose
GN vtype ^DEnter type 0-fixed 1-random 2-gradient: (0)
:EndChoose
GN vTooMuch 1
IFN vLineNumb-vLineLimit EndFoolProof
GN vTooMuch 0
GN vLineNumb vLineLimit
:EndFoolProof
GV vPercInc 100/(vLineNumb-1)
LWIDTH vLineWidth
COLOR vcolor
GN vLoopNumb 0
GOLAYER MYLINES
SHEET MACROTEST
GV varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
```

ECOFF

This command hides the following command that till now appeared in the command line.


Default parameters?

```
Number of side lines:    10
Max. Number of lines:    100
Max. length of lines:    10
Min. length of lines:    2
Lines color:             0
Lines width:             0
Type: fixed to max length: 0
random length(1) gradient(2)
```

the ASKBOX

Blank line again

I have inserted a blank line here that will cause no crash.

It says to CC3 : here is the end of the ASKBOX.

Otherwise, the whole macro body following would have been shown in the box...

LWIDTH

It's the standard CC3 command to change the width of lines.

SHEET

This command forces all drawings on the MACROTEST sheet.

Sheets do not need to exist (Layers do, see page 5)

```
IFZ varPerc NoBefore
GP vBefore % varPerc-0.1 varEnt
GO Before
:NoBefore
GP VBefore % 0 VarEnt
:Before
GV vLineSize vMaxL
IFZ vtype LengthDone
IFZ vtype-2 LGradient
RANDOM varLmodif
GV vLineSize vminL+(vmaxL-vminL)*varLmodif
GO LengthDone
:LGradient
GV varLmodif 50-varPerc
IFP varLmodif Absolute
GV varLmodif -varLmodif
:Absolute
GV vLineSize vminL+(vmaxL-vminL)*(1-varLmodif/50)
:LengthDone
GP vAfter % varPerc+0.1 varEnt
GBRNG varBrng vBefore vAfter
GP varPperp ref varPOnEnt <varBrng+90,vLineSize
LINE varPOnEnt varPperp;
GV varPerc varPerc+vPercInc
GN vLoopNumb vLoopNumb+1
IFZ vLoopNumb-vLineLimit MacroDone
IFP varPerc-(100-vPercInc/2) EndLoop
GO Loop
:EndLoop
GP vBefore % 99.9 varEnt
GP vAfter % 100 varEnt
GP VarPOnEnt % 100 varEnt
GBRNG varBrng vBefore vAfter
GV vLineSize vMaxL
IFZ vtype LastLength
IFZ vtype-2 LastGradient
RANDOM varLmodif
GV vLineSize vminL+(vmaxL-vminL)*varLmodif
GO LastLength
:LastGradient
GV varLmodif 50
GV vLineSize vminL
:LastLength
GP VarPperp ref varPonEnt <varBrng+90,vLineSize
LINE VarPOnEnt VarPperp
GO MacroCompleted
:MaxMinErr
MSGBOX Max/Min error!
The min Length is greater
than the max Length...

Go MacroDone
:OneLine
MSGBOX You Lazy!
Cannot draw single line?

GO MacroDone
:MacroCompleted
IFZ vminL-vmaxL ForcedType
IFZ vTooMuch ForcedNumb
MSGBOX Completed!

GO MacroDone
:ForcedType
MSGBOX Completed but...
Type set to 0 (fixed)
```

Blank lines...

All those blank lines have the same function : marking the end of the message box text.

```
because min Length is
equal to max Length!

GO MacroDone
:ForcedNumb
MSGBOX Completed but...
Line number set to Max Number
because Line Number greater
than Max Number...

:MacroDone
GETSETTINGS
ECON
ENDM
```

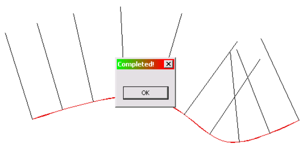
Message boxes

I put a lot of MSGBOX here. Feel free to erase them, they are only to illustrate the command.

Macro Completed

Some macro take a lot of time and the end is not always clear.

A message Macro Completed is sometimes helpfull...



Result with MSGBOX

ECON

As ECOFF hides the commands, ECON makes the next visible. Don't forget it or you won't see what you do AFTER the macro.

Wow. That has grown in quite some text... Remember, the file is available for download...

I won't comment all the additions, with the sidebars and what you've learned now I hope you can analyse it. Feel free to ask any question on the forum.

MACRO TIP

As I said, the text has grown from 20 lines to those two full pages.

It's generally the way I write macros : begining small and adding each step a bit more options/enbellishments.

10 GLEN

Instead of telling how many lines to draw, it migth be better to specify the distance between to lines. That way, if you'd like to use the macro more than once (otherwise it won't be that usefull, eh?) all the entities would look uniform.

The math of it is if you specify the distance you need to calculate the number of lines anyway. It would be easy : divide the length of the entity by the distance interval. For that, of course, we need to know the length of the entity and here comes GLEN, the command that yields said length. The syntax is, as all Gs, GLEN followed first by the name of the variable and only the entity at the end.
`GLEN VariableName EntityVariable`

I won't use the huge previous macro but start again with the vertical lines (could be usefull for an escarpement...)

You should know enough now to add choice of color, line width, random or gradient length...

```
MACRO MYLINES6
SAVESETTINGS
ECOFF
```

```
GE varEnt ^DSelect entity
IFERR MacroDone
GV vSpacing 10
GV vSpacing ^DEnter lines spacing: (10)
GV vLSize 10
GV vLSize ^DEnter line length: (10)
IFZ vSpacing MacroDone
IFZ vLSize MacroDone
GN vMaxLines 100
GLEN vEntL varEnt
GN vLNumb vEntL/vSpacing+1
IFP vMaxLines-vLNumb NumbOk
GN vLnumb vMaxLines
:NumbOk
GV vPercInc 100/vLNumb
GOLAYER MYLINES
SHEET MACROTEST
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
GP varPperp ref varPOnEnt <-90,vLSize
LINE varPOnEnt varPperp;
GN varPerc varPerc+vPercInc
IFP varPerc-(100-vPercInc/2) EndLoop
GO Loop
:EndLoop
GP VarPOnEnt % 100 varEnt
GP VarPperp ref varPonEnt <-90,vLSize
LINE VarPOnEnt VarPperp
:MacroDone
GETSETTINGS
ECON
ENDM
```

Math again ?

Even though we want to specify the distance interval, we still want something evenly spaced.

`GN vLNumb vEntL/vSpacing+1` yields the number of lines to draw (the +1 to place the last pole, remembre those interval exercices at school ?)

This is an integer (GN) so the result is rounded down and will probably result in a different interval at the end.

To circumvent that, the distance interval is calculated again. So, for each entity, the interval would NOT be the same.

It will be noticable only on very short entities. The longer the entity, the less difference.

Putting a symbol in the Symbol manager

The easiest way I found was to insert the symbol on the map first than erase it. The symbol won't appear on the map but will stay in the manager.

To use this macro, you will have first to place the Decid Tree 1 (standard cc3 vegetation catalog) on your map.

11 Symbols

Using symbols is a tricky part I've not yet mastered to my taste.

If you look in the Tome of Ultimate Mapping (pages 134-151) you will notice that almost every CC3 command involving symbols are not usable in macros. A few only have macro equivalent (ibid, pages 153-156).

I can however show what can be achieved with the INSSYM and SYMSORT commands

INSSYM is a command inserting a symbol ***already present in the symbol manager*** on a specific location, with specific scale and angle. Syntax is :

```
INSSYM SymbolName;Xscale;Yscale;RotationAngle;Point1;....PointN;
```

SYMSORT is the usual command to sort symbols on a map.

To use SYMSORT we must first find a way to select the symbols.

For now, we will use the SELBYA (Select by All) command. It forces any selection need to select anything.

We can't leave the macro in that mode. The command SELBYD (Select

BY Dialog) is used to return in a standard state.

I'll re-use a version of the MyLines6 macro above, but know, instead of drawing lines I will place symbols along the path.

```
MACRO MYSYM
SAVESETTINGS
ECOFF
GE varEnt ^DSelect entity
IFERR MacroDone
GV vSpacing 10
GV vSpacing ^DEnter symbols spacing: (10)
GV vSSize 1
GV vSSize ^DEnter symbol scale: (1)
IFZ vSpacing MacroDone
IFZ vSSize MacroDone
GN vMaxSym 100
GLEN vEntL varEnt
GN vLNumb vEntL/vSpacing+1
IFP vMaxSym-vLNumb NumbOk
GN vLnumb vMaxSym
:NumbOk
GV vPercInc 100/vLNumb
GOLAYER MYSYMBOLS
SHEET MACROTEST
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
INSSYM Decid Tree 1;vSSize;vSSize;0;varPOnEnt;
GN varPerc varPerc+vPercInc
IFP varPerc-(100-vPercInc/2) EndLoop
GO Loop
:EndLoop
GP VarPOnEnt % 100 varEnt
INSSYM Decid Tree 1;vSSize;vSSize;0;varPOnEnt;
SELBYA
SYMSORT
SELBYD
:MacroDone
GETSETTINGS
ECON
ENDM
```

Selction methods

SELBY1 lets you click once only. You can get multiple entities anyway if they are near.

SELBYA : selects all

SELBYC: by Color

SELBYD : by dialog

SELBYL : by Layer

SELBYW : by window (rectangular selection)

SELBYE : by each, means that you may click several times to select.

SELBYP : by prior, selects all prior selected entities.

Result. Note the lonely tree used to put the symbol in the manager.

12 ERA and layer play

This last macro would even be better if the red line would dissapear at the end. You might keep it on a hidden layer or erase it, the process is the same.

Remember the GE command ? It puts an entity in a variable. The truth of it is that sometimes the variable is considered as an entity (GLEN for example) and sometimes as a point.

So if I add something like
`ERA varEnt`
at the end, it will not only select the line, but any symbol, line, whatever

located near the same point.

To circumvent that we must use layers. The method is this :

1. Move the entity to a specific layer such as NOSEEN or TOERASE.
2. Hide all other layers with a HIDEA
3. Select all with SELBYA
4. ERA
5. Show all layers by SHOWA.
6. HIDE NOSEEN if erase is not an option.

The modified macro is :

```
MACRO MYSYMERA
SAVESETTINGS
ECOFF
GE varEnt ^DSelect entity
IFERR MacroDone
SELBY1
GOLAYER TOERA
CHANGEL varEnt TOERA
GV vSpacing 10
GV vSpacing ^DEnter symbols spacing: (10)
GV vSSize 1
GV vSSize ^DEnter symbol scale: (1)
IFZ vSpacing MacroDone
IFZ vSSize MacroDone
GN vMaxSym 100
GLEN vEntL varEnt
GN vLNumb vEntL/vSpacing+1
IFP vMaxSym-vLNumb NumbOk
GN vLnumb vMaxSym
:NumbOk
GV vPercInc 100/vLNumb
GOLAYER MYSYMBOLS
SHEET MACROTEST
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
INSSYM Decid Tree 1;vSSize;vSSize;0;varPOnEnt;
GN varPerc varPerc+vPercInc
IFP varPerc-(100-vPercInc/2) EndLoop
GO Loop
:EndLoop
GP VarPOnEnt % 100 varEnt
INSSYM Decid Tree 1;vSSize;vSSize;0;varPOnEnt;
SELBYA
SYMSORT
ASKBOX CC3
Erase entity after completing ?

IFERR MacroDone
LAYER TOERA
HIDEA
SELBYA
ERA
SHOWA
:MacroDone
SELBYD
GETSETTINGS
ECON
ENDM
```

Revelant blocks are :

```
GE varEnt ^DSelect entity
IFERR MacroDone
SELBY1
GOLAYER TOERA
CHANGEL varEnt TOERA
```

The entity is moved to the TOERA layer that is created by a GOLAYER for this purpose.
The SELBY1 is there to ensure no dialog is launched (remember, it means select once, not select ONE entity)
CHANGEL is the classic CC3 command to change a level, it is given the parameter of the entity we want to move and the target layer.

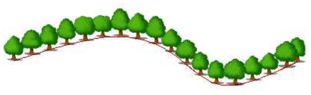
NOTE : if your map is encumbered their might be several entities changing layer ! Be carefull.

This change must occur before we add anything ! Best is to do it just after selecting the entity.

```
ASKBOX CC3
Erase entity after completing ?

IFERR MacroDone
LAYER TOERA
HIDEA
SELBYA
ERA
SHOWA
```

Here I use an ASKBOX to make erasing the line an option.
If erasing is asked for, LAYER TOERA makes this layer active again and HIDEA hides all the other layers.
SELBYA is here academic because it's already the selection state (see line above block)
ERA doesn't need any selection because ALL is erased due tu SELBYA.
SHOWA shows again everything.

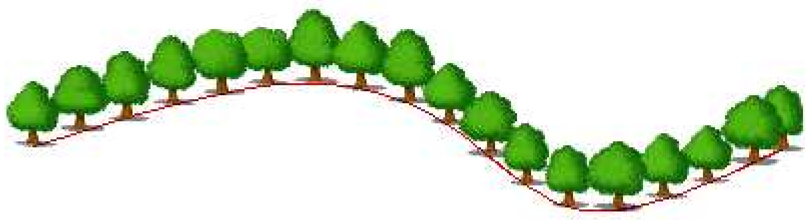

Result

Note : if you had hidden layers before they aren't hidden anymore...

This one uses random to choose Decid Tree 1 to Decid Tree 6 and modify slightly the scale :

```
MACRO MYSYMR
SAVESETTINGS
GE varEnt ^DSelect entity
IFERR MacroDone
GV vSpacing 10
GV vSpacing ^DEnter symbols spacing: (10)
GV vSSize 1
GV vSSize ^DEnter symbol scale: (1)
IFZ vSpacing MacroDone
IFZ vSSize MacroDone
GN vMaxSym 100
GLEN vEntL varEnt
GN vLNumb vEntL/vSpacing+1
IFP vMaxSym-vLNumb NumbOk
GN vLnumb vMaxSym
```

```
:NumbOk
GV vPercInc 100/vLNumb
GOLAYER MYSYMBOLS
SHEET MACROTEST
GN varPerc 0
:Loop
GP varPOnEnt % varPerc varEnt
RANDOM vrTree
RANDOM vrSize
GV vrSize vSSize*(0.9+0.2*vrSize)
GN vrTree (6*vrTree+1)
IFZ vrTree-2 T2
IFZ vrTree-3 T3
IFZ vrTree-4 T4
IFZ vrTree-5 T5
IFZ vrTree-6 T6
INSSYM Decid Tree 1;vrSize;vrSize;0;varPOnEnt;
GO TreePlaced
:T2
INSSYM Decid Tree 2;vrSize;vrSize;0;varPOnEnt;
GO TreePlaced
:T3
INSSYM Decid Tree 3;vrSize;vrSize;0;varPOnEnt;
GO TreePlaced
:T4
INSSYM Decid Tree 4;vrSize;vrSize;0;varPOnEnt;
GO TreePlaced
:T5
INSSYM Decid Tree 5;vrSize;vrSize;0;varPOnEnt;
GO TreePlaced
:T6
INSSYM Decid Tree 6;vrSize;vrSize;0;varPOnEnt;
:TreePlaced
GN varPerc varPerc+vPercInc
IFP varPerc-(100-vPercInc/2) EndLoop
GO Loop
:EndLoop
GP VarPOnEnt % 100 varEnt
INSSYM Decid Tree 1;vSSize;vSSize;0;varPOnEnt;
SELBYA
SYMSORT
SELBYD
:MacroDone
GETSETTINGS
ECON
ENDM
```



Result

All commands seen so far

Command	Effect	Syntax	Page

ASKBOX	Shows a yes/now question in a message box	`MSGBOX HeaderText`
		`Text Line`
		`Text Line`
		`....`
		`<blank line>`
			10

CHANGEL	Changes layer of selected entities	`CHANGEL`	15

COLOR	Changes current color	`COLOR integer`
		`COLOR integerVar`
			3

ECOFF	Does not show following commands in the command line	`ECOFF`	10
ECON	Makes following commands visible in the command line	`ECON`	12

ENDM	Delimits macro	`ENDM`
		`<blank line>`
			1

ERA	Erases selected entities	`ERA`
		`ERA EntityVar`
			15

GBRNG	Stores the angle in a variable	`GBRNG Point1 Point2`	6
GE	Stores an entity in a variable	`GE varName Entity`	5
GETSETTINGS	Makes settings stored by SAVESETTINGS current	`GETSETTINGS`	4
GL	Stores a string in a variable	`GL varName String`	1
GLEN	Stores length of entity in a variable	`GLEN varName EntityVar`	12

GN	Stores an integer in a variable	`GN varName integer`
		`GN varName math.expression`
			3

GO	Go to label	`GO Labelname`	3
GOLAYER	Makes the layer current layer and creates it if necessary	`GOLAYER LayerName`	5

GP	Stores an x,y point in a variable	`GP varName Point`
		`GP varName x,y`
			1

GV	Stores a real number in a variable	`GV varName real`
		`GV varName math.expression`
			3;9

HIDEA	Hides all layers excepted current	`HIDEA`	15
IFERR	If error occurs go to label	`IFERR LabelName`	2

IFN	If expression is less than zero go to label	`IFN expression LabelName`
		`IFN NumbVar LabelName`
			3

IFP	If expression is more than zero go to label	`IFP expression LabelName`
		`IFN NumbVar LabelName`
			3;9

IFZ	If expression is equal to zero go to label	`IFZ expression LabelName`
		`IFZ NumbVar LabelName`
			3

INSSYM	Inserts a symbol **defined** in symbol manager	`INSSYM`
		`SymbolName;Xscale;Yscale;RotAngle;Point1;...;PointN;`
			13

Label (not a command)	Localises label	`:LabelName`	2
LAYER	Makes the layer current. Layer **must** exist.	`LAYER LayerName`	5
LINE	Draws lines	`LINE Point1 Point2...PointN;`	6

LWITDH	Sets line width	`LWIDTH RealNumber`
		`LWIDTH variable`
			11

MACRO	Defines new macro	`MACRO macroName`	1

MSGBOX	Shows a message box	`MSGBOX HeaderText`
		`Text Line`
		`Text Line`
		`....`
		`<blank line>`
			10

RANDOM	Stores a random number between 0 and 1 in a variable	`RANDOM varName`	8
SAVESETTINGS	Stores current settings	`SAVESETTINGS`	4
SELBY1	Selects with one click or one selection	`SELBY1`	14
SELBYA	Selects all entities visible and not frozen	`SELBYA`	14
SELBYC	Selects by color	`SELBYC ?`	14
SELBYD	Selects by dialog	`SELBYD`	14
SELBYE	Selects by multiple clicks	`SELBYE`	14
SELBYL	Selects all entites on a layer	`SELBYL`	14
SELBYP	Selects prior entities selected	`SELBYP`	14
SHEET	Makes the sheet current sheet. Sheet **does not** need to exist	`SHEET SheetName`	11
SHOWA	Shows all layers	`SHOWA`	15
SYMSORT	Sorts symbols selected **before**	`SYMSORT`	13
TEXTM	Macro equivalent to TEXT : puts a text on the map.	`TEXTM text;point`	1

This ends part1. See you soon!