
DeCaf 1.0

# Introduction


Welcome to the wonderful world of programming.

Well, maybe I shouldn't be starting the text with this sentence. After all, if you want to learn DeCaf it is probably because you are NOT a programmer. And, for non-programmers, the "world of programming" is not a very attractive place.

I hope DeCaf will allow you to start giving your first steps in that — at least for now — "non-attractive" world of programming.

●

Nowadays the number of available computer programming languages is overwhelming.

And you have computer programming languages that suit all tastes, from the very powerful and complex to the very limited and simple. Unfortunately, they usually suit the tastes of... guess who? Programmers. What I mean by this is that there aren't many computer programming languages that are suited for people whose mere mention of the word "programming" induces spasms or severe rashes.

Generally, people that are not programming-biased tend to consider programming as something that requires huge amounts of mathematical knowledge, a very organized memory and lots of logical thinking. Well, all of these are useful but not required. At least not in large doses. For those non programming-biased people, any computer programming language looks or sounds like an extra-terrestrial gibberish. Because of that, some computer programming languages started emulating regular written English. Such is the case of AppleScript, the scripting language of MacOS© or ActionScript, the scripting language of Adobe Flash©.

Those languages allow a much more flexible construction of a program. They are also much simpler to read and code. If they are so much more accessible to the masses, why aren't all programming languages based on the paradigm of regular written languages? Well, those types of languages have their own drawbacks. They execute much slower and, usually, they are not as efficient or powerful as the other types of languages. That happens because they rely on a syntax that is much more ambiguous than a strict, rule-based syntax common to most programming languages. Also, the regular written language lacks the effectiveness and structural cleanness needed to define and manipulate certain types of events, algorithms or data structures.

Wow! I've already started using those alien gibberishes: Events? Algorithms? Data structures? These are terms common in most computer languages. But I really don't want to scare you, so I will refrain from using such terms. Well, at least for now.

As you may be wondering, DeCaf is a computer programming language that is based on regular written English. Kind of... I mean, it is much easier to read and understand than most other computer programming languages. But if you show it to an English teacher, telling him that it is something you wrote, he will promptly contact a mental health institute.

When I first started creating DeCaf I had several purposes:

- to create a language that would be simple enough to allow people that know nothing about programming start coding.
- instead of using abbreviations or mnemonics for commands and functions, like most other computer programming languages I wanted to use expressions similar to regular written English.


- to avoid strict and/or complex syntax, very common to most computer programming languages.
- to allow for flexible scripting with the possibility of using several variations of expressions for the same purpose.
- to allow the use of abbreviated expressions.
- to build an editor that would help users learn DeCaf while coding.
- to present very clear and informative error messages.

Fortunately, all of those purposes were fulfilled. Nevertheless, while creating DeCaf I stumbled upon several ambiguous and doubtful situations that forced me to make some surveys among non-programmers (potential DeCaf users) and make some decisions.

For example, I decided not to deviate too much from some common programming paradigms. This means that DeCaf is not like proper written English, like I said before, but it is a whole lot more comprehensible than COFFEE or C++ (the other two programming languages used in Cinema 4D).

Keeping DeCaf close enough to other programming languages — while still making it simpler — had two advantages: it was a little easier for me to code it and it served as a stepping stone for DeCaf users to be able to learn enough programming paradigms so that they could, in the future, evolve to more complex programming languages.




Let me show you some examples of what I mean. First, an example of the simplicity of DeCaf.

In COFFEE, if you want to create a new Null Object you need to use the following code:

```
main()
{
var null_obj,document;

document=GetActiveDocument();
null_obj=new(NullObject);
document->InsertObject(null_obj,NULL,NULL);
}
```

In DeCaf, to do the same you would use the following code:

```
create null
```

Simpler, isn't it? Almost everything — if not everything — has this degree of simplicity, specially when compared with its counterpart in COFFEE or C++.


Now an example of where I could not keep DeCaf away from regular programming paradigms.

In COFFEE (and in all other programming languages) we have a way to create cycles (groups of instructions that are executed a specific number of times). Actually there are many ways to create cycles, but one of the simplest is:

```
main()
{
var f;
for(f=1;f<=10;f++)
 {
 println("Iteration:");
 println(f);
 }
}
```

For the non-code-biased people (the majority of the readers of this manual) this snippet of code is as intelligible as the scribbles of a two year old child. I will not even try to explain what each line does (for that you can buy my book about the COFFEE programming language).

What the code does is to print the following in the Console window:

```
Iteration:
1
Iteration:
2
Iteration:
3
Iteration:
4
Iteration:
5
Iteration:
6
Iteration:
7
Iteration:
8
Iteration:
9
Iteration:
10
```


The same would be done in DeCaf with the following code:

```
variable f
repeat 10 times using f
<<
print "Iteration:"
print f
>>
```

The good news is that DeCaf would also do the same if you wrote:

```
var f
repeat 10 f
<<
print "Iteration:"
print f
>>
```

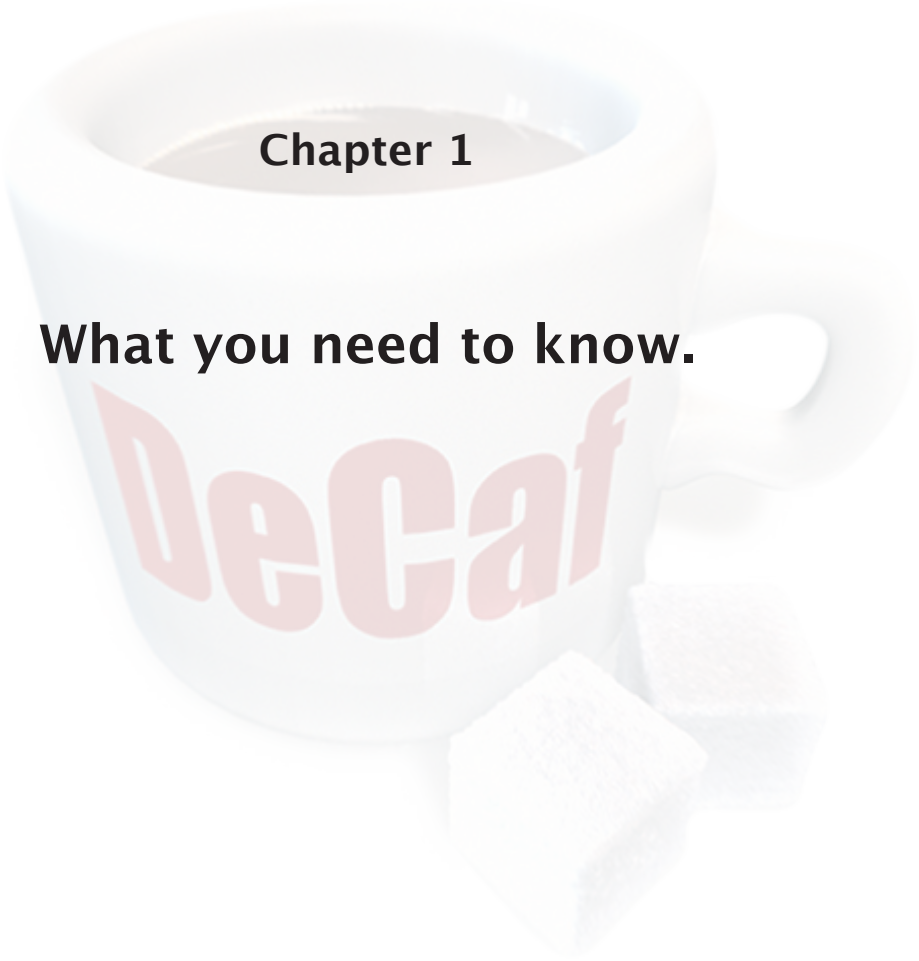
As you can see, DeCaf is versatile enough to allow for different versions of the same commands. You can type them in full, allowing for more clear reading and understanding of your code. But when you are already proficient in DeCaf you may very well use the shorter syntax.

Some commands can even be completely replaced by alternative synonyms. For example, **ask** and **input** will perform the same task.

You may be asking where, in the DeCaf listing, I could not escape from regular programming language paradigms. Well, in the COFFEE listing we define blocks of code by placing commands between **{** and **}**.

In DeCaf we define blocks of code by placing commands between **<<** and **>>**. Later in the manual I will explain why did I decided to use **<<** and **>>**. I will also explain why do I even need to be able to create blocks of code.

This is the type of information that will allow you to give your first steps in the world of programming. After becoming a master in DeCaf you will be a few steps away from learning COFFEE or even C++. But, at least, the learning curve will be much smoother.




# Chapter 1

# What you need to know.

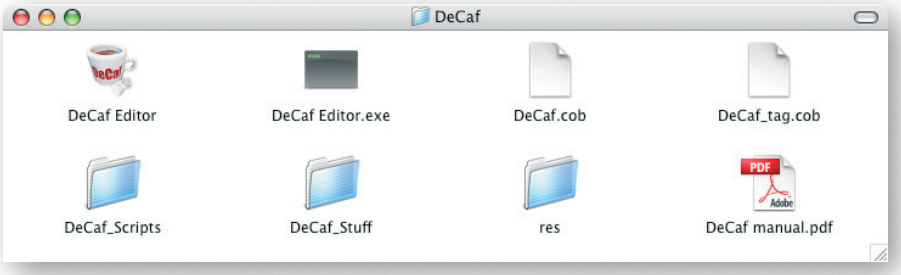
# Chapter 1 - What you need to know.

Like all other plug-ins for Cinema 4D, DeCaf must be installed in the plugins folder that resides inside your Cinema 4D folder. When I say “installed” I mean copied because DeCaf, like most other Cinema 4D plug-ins, doesn’t really require any installation.

It is very important that you check the privileges of the DeCaf folder and all its content (folders and files). The user that is running Cinema4D — probably the administrator — should have read and write privileges to the DeCaf folder and all its content. This is because DeCaf is a very complex piece of software and reads and writes a lot to its folder.

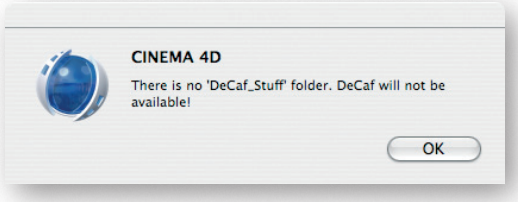
Also, the folder/file structure is very important and should not be changed in any manner. If you change the name or location of any folder or file inside the DeCaf folder, DeCaf will simply stop working. The only folder whose content you can mess with is the **DeCaf_Scripts** folder. Inside it reside all the scripts you create and you can manage them as you feel like it. You can delete them, duplicate them and even rename them. Just don't change or delete the extensions of the script files.

The usual structure of the DeCaf folder is the following:



The first time you run Cinema 4D after *installing* DeCaf, you will be asked to reveal where your Cinema 4D application is. This is just for DeCaf to know the name of your Cinema 4D application as it will need it when switching between Cinema 4D and the external DeCaf editor. If you don't change the location of your Cinema 4D application this will usually only be asked once.

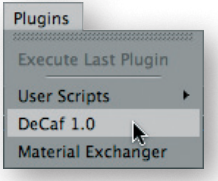
Each time you run Cinema 4D, DeCaf checks for its folder structure to make sure everything it needs is in place. If something doesn't pass the tests, you will get a warning and DeCaf will not be available. For example:




If everything checks out fine you will be asked to input your serial number. Don't worry as this will only happen the first time you run Cinema4D with DeCaf in the plug-ins folder. Just type the serial that you received when you purchased DeCaf and press the **OK** button.

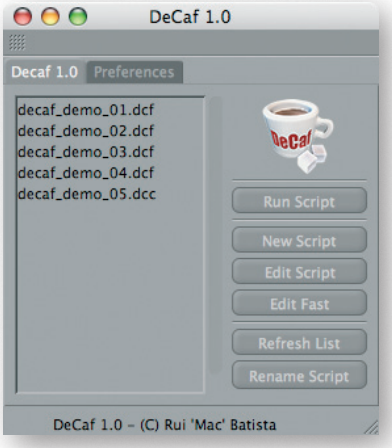
# Chapter 1 - What you need to know.

It is **VERY IMPORTANT** that you click in OK and not just press Enter at the end of inputting the whole serial. DeCaf will only check the serial if you **PRESS THE 'OK' BUTTON!** The serial has five groups of five characters each, separated by hyphens. This is how it is provided to you but, when inputting it, you just have to type the characters. The cursor will automatically advance to the next field as soon as you type five characters and you should **NOT** type the hyphens.



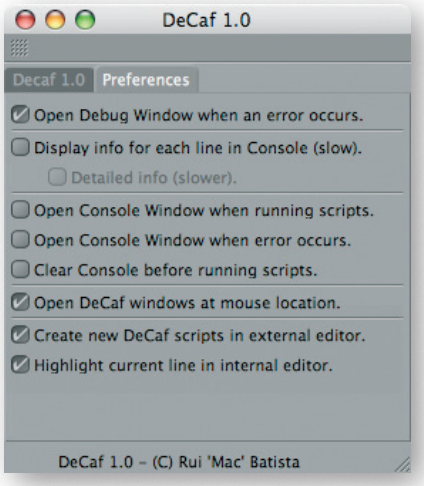

Assuming that everything is ok and that the serial is correct, once Cinema 4D finishes loading, you will get access to DeCaf from the plugins menu.

After choosing DeCaf from the plugins menu you will see its interface.




On the left side of the window there is a list of all available scripts. This list of scripts reflects the content of the **DeCaf_Scripts** folder, showing the regular DeCaf script files — with the extension **.dcf** — and the password protected DeCaf script files — with the extension **.dcc**.

On the right side, there are six buttons that perform operations on the script that is selected in the list. The top button, named **Run Script** will execute the script that is selected in the list. The button named **New Script** will create a new empty script. Depending on the preferences it will automatically open the external DeCaf editor or ask you for the name of the new script and open a new window for editing it. Pressing the **Edit Script** button will open the script selected in the list in the external DeCaf editor. Pressing the **Edit Fast** button will open the script selected in the list in a window,




without leaving Cinema 4D. Pressing the **Refresh List** button will reload all the DeCaf scripts that are inside the **DeCaf_Scripts** folder into the list, sorting then alphabetically. This is specially useful if you organized your DeCaf scripts manually. The last button is named **Rename Script**. As its name implies, you use it to rename a script — it has to be selected, of course — without leaving Cinema 4D.

You always have the choice to rename the scripts using the Mac Finder or Windows Explorer. Just don't forget that you may need to refresh the list — pressing the **Refresh List** button — if you edit something inside the **DeCaf_Scripts** folder while Cinema 4D is still opened.

Deleting and duplicating scripts always has to be done using the Mac Finder or Windows Explorer. Once again, if you do any of these operations while Cinema 4D is still opened you have to refresh the scripts list by pressing the **Refresh List** button.

As you may have noticed, there are two tabs in the DeCaf interface. The one described earlier is named **DeCaf 1.0** and it will be the one you will be using more often. The other one, named **Preferences** contains all the available options that allow you to customize DeCaf behavior.

In the following pages there is a detailed explanation about all the options available in the **Preferences** tab.

**Open Debug Window when an error occurs.** — **Default: ON**

When turned on, DeCaf will open the Debug Window whenever an error occurs. The Debug Window shows the line where the error occurs, the error message and a list of all variables an their content. It is specially useful when you are still learning DeCaf or when you are debugging your DeCaf code.

**Display info for each line in Console (slow).** — **Default: OFF**

When turned on, DeCaf will display the content of each line of code in the Console Window, as it is being executed. It states that this will slow down the code but, in fact, it is just a few milliseconds slower. Just turn this off if you want your code to run as fast as possible. This option is specially useful when you are still learning DeCaf or when you are debugging your DeCaf code.

**Detailed info (slower).** — **Default: OFF**

This option is available only when the previous option is turned on. When turned on, DeCaf will display additional information about the content of each line of code in the Console Window, as it is being executed. It states that this will slow down the code but, in fact, it is just a few milliseconds slower that the previous option. Just turn this off if you want your code to run as fast as possible. This option is specially useful when you are still learning DeCaf or when you are debugging your DeCaf code.


**Open Console Window when running scripts.** — Default:OFF

When turned on, DeCaf will automatically open the Console Window when you press the **Execute Script** button. This way you don't have to, manually, choose Console from the Window menu or press Shift+F10. It is specially important to keep the Console Window opened when you are starting to learn DeCaf or when you are debugging your code because very important information is displayed there.

**Open Console Window when error occurs.** — Default:OFF

When turned on, DeCaf will automatically open the Console Window whenever an error occurs. This is very useful because, sometimes, your code may not be doing what you want and you don't know what may be happening. If you have this option turned off and some important event occurs, DeCaf will open the Console and display a message asking you to turn on this option.

**Clear Console before running scripts.** — Default:OFF

When turned on, DeCaf will automatically clear the Console Window when you press the **Execute Script** button. This is very useful because, sometimes, the Console Window is full of "garbage" and you may want to have a clean display as you run each of your scripts.

**Open DeCaf windows at mouse location.** — Default:ON

When turned on, DeCaf will open its windows (editor and Debugger) at the mouse location. Otherwise, the windows will open at the top-left corner, always at the same location (128 pixels from the left of the screen and 128 pixels from the top of the screen).

**Create new DeCaf scripts in external editor.** **Default:ON**

When turned on, new DeCaf scripts will be created using the external editor (opening it if it is not already open). Otherwise, new scripts will be created in Cinema 4D, using the simpler editor. When you are still giving your first steps in DeCaf it is advisable to use the external editor. When you get more confident with DeCaf you can start using this internal, simpler editor.

**Highlight current line in internal editor.** **Default:ON**

When turned on, the line where the cursor currently is located, in the internal DeCaf editor is highlighted. It may be useful, specially for people with screens set to a very high resolution since the blinking cursor is not particularly visible.

If you ever decide to revert the preferences to their default values in one go, simply delete the **decafpref.txt** file that is inside the **DeCaf_Stuff** folder. DeCaf will create a new **decafpref.txt** file with the default values the next time you run Cinema 4D.

If you plan to use DeCaf very often you may want to dock the DeCaf window somewhere in your layout. Just drag the dotted square that is on the top-left corner of the DeCaf window (above the DeCaf tab) and dock the window wherever you prefer. You may even turn the docked DeCaf window into a tabbed window. This way you can associate it with other windows and save some space since DeCaf is not something that you need to have accessible all the time.