0.1 Have a start

0.1.1 What is WML?

WML is a shortening for **W**esnoth **M**arkup **L**anguage. WML is a flexible system to program Wesnoth scenarios. Its very easy to learn and can do very much. It is designed from the developers of Wesnoth because nobody wants to write the whole scenario in a real programming language. So the WML was invented to make programming of scenarios more overviewable and easy enough that users can make their own campaigns, with a minimized effort.

0.1.2 How does it basically work?

Like other programming languages, you will have to type a lot of special letters and all kinds of parantheses. Let me explain what all the signs do.

The tag [. . .]

They are the most important signs the WML has. Without them, nothing works. The common commands of Wesnoth all look like this:

[*Command name*]

And then, the tag has to be closed again:

[/*Command name*]

Between you type all the needed options and arguments. More to this later.

The braces { . . . }

You will stumble over them very early and you will ask yourself *What should that be? Why is there no tag and all is written in large capitals?*
–Explaination:
There is something we call macros. A macro is a fragment wich is defined otherwhere and it's not in the scenario file itself because its very long and the author wants to keep the overview, or it defines a very complicated part and it is confusing when it is fully-written everywhere, or it's simply for saving space on the computer. A macro definition looks like this:

```
#define REDRAW[redraw] [/redraw] #enddef
```

Don't try to understand it now. I will explain you whats now important to know. (Altough, it should already give some insights) The definition always begins with a `#define` and then the name of the macro. The command is preluded by a pound (`#`). The pound normally makes the rest of the line to a comment. (by the way, definitions, `#ifdef` stuff and comments are the only reason why you can't write a whole scenario in one line) Everything else than `#define,#enddef,#ifdef,#endif` will be ignored. The definition always ends with an `#enddef`. And when you insert the macro name (always in large capitals) somewhere in your scenario, between braces, the computer inserts the fully-written macro for himself there. There are also some built-in macros you don't have to define.

The pound #

As already explained, it makes the rest of the line to a comment, from the pouind itself on. But you have to write it at the begin of every new sentence to make it to a comment.

Comments are useful because when you program a more fifficult and complicated things, you maybe want to know what it should do later too.

The normal parantheses (...)

That's more difficult. You will learn about it later, in the section "macros.

Thats all nice, but where and how do I write scenarios?

First: The location of your campaign should be:

Linux	*/.wesnoth/data/campaigns/
Windows	/userdata/wesnoth/data/campaigns/
Mac	/Library/Preferences/Wesnoth/data/campaigns/

*your home folder

Second: You write the files in a text editor. I advice you **not** to use MS Word or OpenOffice and these. Better is WordPad (Windows) or Kwrite (Linux) or a similar text editor. The scenarios you write are not formatted. That means no bold text, no colours, no right/left/central flush and all the features you accustomed. The text editors event do not care about the paper format because they don't have such a thing. That's also the problem of Word and similar programs. When you have more complicated scenarios, you will probably need more space for a line than a line of a paper sheet has. That sounds crazy, but you will know soon what I mean.

In the text editor you write your scenario file and then save it as text. But don't put the default ending .txt on the end. You write

```
filename.cfg
```

.

Enough theory! I want to make my first campaign!

Ok, here we go.

0.2 My first campaign

For a campaign, you need a few things.

•A file where you define your campaign. That means you "tell the computer where he finds all needed files for the campaign and wich scenario is the first.

Also it includes the name and the difficulties. I will explain this as the first thing when it goes to programming.

•A folder where you put all the sharing files in. For an example, scenarios. It should have the same name as the campaign file.(again, you will find the reason later) Ready? Go!

0.2.1 The campaign file

Call it however you want, but I really advice you to make clear wich campaign it is.

I give you the code, and then I explain, ok?

```
[textdomain]
name="wesnoth-my_first_campaign"
path=campaigns/my_first_campaign/translations/
[/textdomain]

[campaign]
id=my_campaign
name=_"My first campaign"
define=CAMPAIGN_MY_FIRST
difficulties=EASY,MEDIUM,HARD
difficulty_descriptions={MENU_IMG_TXT2 human-spearman.png _"
(Newbie)" _"(easiest)"} + ";" + {MENU_IMG_TXT human-swordsman.png _
"(Developer)"} + ";" + {MENU_IMG_TXT2 human-general.png _"(Expert)"
_"(hardest)"}
icon=misc/item-darktome.png
description=_"I actually developed my own campaign"
image=misc/disk.png
[/campaign]

[binary_path]
path=data/campaigns/my_first_campaign/external_binary_data/
[/binary_path]

#ifdef CAMPAIGN_MY_FIRST
{@campaigns/my_first_campaign/scenarios}
{@campaigns/my_first_campaign/maps}
{@campaigns/my_first_campaign/utils}
[+units]
{campaigns/my_first_campaign/units}
{~campaigns/my_first_campaign/units}
[/units]
#endif
```

Explainations:

campaign – This will hopefully be clear.

- id= – On `id`s the computer discerns "what this is.
- name=_– Whenever you set non-argument text, you have to put it into double-quotes. Whenever the text has to be expanded somewhere, use `~_"(text)`.
- define= – This is not a macro. It's a definition you use for an `#ifdef`.
- difficulties= – This is also a definition for `#ifdef`s. You will see this over and over:
  `#ifdef EASY`
- difficulties= This describes the difficulties. . . . Wait! What's the following horrible stuff!?
  →Please continue the reading.
- MENU_IMG_TXT – Menu-image, text. (This is a macro) Whenever an image is defined anywhere, the computer searches in the ~/wesnoth/images/ directory and in your campaign sharing folder in the section images. (if it exist:it isn't required) If it's not directly in there, or it's in another folder, you have to specify the path. Example: you have a sub-directory in (*Your campaign's name*)/images called "/items. Then you have to write items/*(image)*.png[1]. The linebreaks you see are made by the editor because you can see what you wrote then. If you make some, it's a **bug!**

  Then you write the name of the dificulty in parantheses, and _[2]. Then
  `+ ; +`
  and then the next difficulty.
  †Attention: The WML is strict with the spaces in a macro. When you don't make them between the single arguments, the computer interpretes it as one argument!
- icon= – You have to insert another image there. It is the one wich appears in the campaign menu.
- description= – Again, that's an individual text wich will be expanded, that means, you use = _". . . . Here you give a summary of the campaign.
- image= – The image wich appears under your campaign description or summary.

/campaign A tag should be closed again, eh?

binary_path – Here you have to put a path in if you want something of your campaign folder in somewhere other than in the campaign. Example, in the campaign menu. Then you have to create such a folder, and insert the export-files. And describe the path as seen. When you don't have such files, just insert a path into your campaign folder. [3]

- #ifdef CAMPAIGN_MY_FIRST – Means "When this campaign is chosen, load following files: to the computer. And then you put in the paths. I think this is understandable with logic, isn't it?

[1] All images have the "png extension
[2] With an = you get it on the other end of the menu(difficulty rate)
[3] Without guaranty

- #endif – Whenever there was an `#ifdef`, there *must* be an `#endif`, or you create a horrible bug!

It's important to say now that you can shuffle all this keys as you like it the best. But I advice you to make an understandable order of them.

0.3 After the campaign file is completed,

We look forward to a scenario file. Because the file can be very long, I don't write you an example and then I explain, I will write a small piece each time, and then explain. At the end of this section, you see the whole file. Go!

0.3.1 Basic settings

We start off with [scenario].And then you have to configure a few basic things before you start with writing it in real.

```
[scenario]
id=first
name=_"My first scenario"
map_data="{@campaigns/maps/first}"

victory_when_enemies_defeated=yes
music="wesnoth-1.ogg"
next_scenario=second

{TURNS 36 30 24}
```

0.3.2 That's enough for the moment.

scenario – You write each scenario in this tag. A tag in a tag is allowed.[4]

- id= – The scenario's id. The computer does only look for the id. It must be the same as the term you wrote in first_scenario at the campaign file. You can make a different name and a different filename and the scenario is still found.
- name= The scenario name. It is expanded text, wich means you have to write it as I did.
- map_data= – Specify here where the map is located. And you have to set the path between quote marks. Also you need the @ at the beginning. There is another way to do the map data: You copy the index of the map file in it and treat it as expanded text.
- victory_when_enemies_defeated= – You can set this option "yes or "no. When you set it "yes, you win automatically when you have wiped all enemy leaders off the plartform. It also allows an event[5] name called "victory.

[4] You can nest as deep as you want.
[5] Later!

- {TURNS 36 30 24} – This means the number of turns on the difficulties EASY,MEDIUM,HARD. You don't have to use this macro. Otherwise, write `turns= (number)`. And you can make the difficulties also with `#ifdef`s. But why since we have this macro?

0.3.3 We go fast forward and do the next part

```
[story]
        [part]
        background=campaigns/Heir_To_The_Throne/story/story1.png
        story=_"A new custom campaign was rising upon the world..."
        music="frantic.ogg"
        [/part]
[/story]
```

Explainations:

story – Begin of the background story of the scenario.

part – One section of the story. After each `[part]`, you have to press "Return when playing. You can have as much as you want of them.

- background= – You *may* use this key to put an image in the background. But all of strangety, this has not to be set under quote marks, but you can do it, as with most of the arguments.
- story=_"... – The story text. Expanded text!
- music= The background music during the story. It is not required to put one in, and this example will not make much sense.

/part – I will not explain the tag closes everytime. But, I repeat that this tag close may be followed by a second `[story]`.

- I will present you another thing now: You have probably already seen those world maps with the dots and the battle cross. You have to grab the right path to the map to place it. And for the symbols, {`DOT X Y`} and {`CROSS X Y`} are commonly used. Please mind that you have to insert the pixel coordinates (easy to find out with an art program) for the `X` and `Y`.