

Morrowind Scripting for Dummies

(6th Edition)

A manual for the TES Construction Set Scripting Language by
GhanBuriGhan
(and many others)



Table of Contents

Foreword to sixth Edition.....	8
Foreword to fifth Edition.....	8
<i>Introduction</i>	9
Using this guide	9
What is a script?	9
What can scripts do?	10
What scripts can't do:	10
<i>Scripting tutorial</i>	11
Let's get going!	11
The scripting window	11
What do we want?	12
Writing a script	12
Naming a script: Begin and End	12
Detecting an action by the player.....	13
Writing text and obtaining decisions from the player	13
How local scripts are executed.....	14
Your first bug	18
Putting a spell on the player	18
<i>General Information: Scripts, Commands and Syntax</i>	22
Types of scripts	22
Local scripts	22
Global scripts.....	22
Syntax	23
Beginning and ending scripts	23
General syntax for functions:	23
General syntax: Commas, parantheses and spaces.....	23
Tabulators.....	24
Variables	24
Declaring variables.....	24
Referencing variables in other local scripts	25
Using variables in functions.....	25
Mathematical calculations.....	26
Testing conditions	27
Use of if... elseif conditions.....	27
While conditions	27
Constructing Boolean operations in TES Script	28
<i>List of TES-script functions</i>	29
Explanation of the format:.....	29
<i>Scripting NPC's: AI and Movement</i>	29
Make an NPC walk to a new location	29

Checking whether an NPC has performed his movement.....	30
Facing	31
Random Actor movement.....	32
Activating Objects.....	32
Following.....	34
Checking which AI package is currently executed	34
Forcing sneak movement.....	35
<i>Text and Dialogue</i>	35
Brief dialogue how-to	35
The concept of MW dialogue.....	35
How dialogue works.....	36
A few golden rules:	37
Dialogue-related functions.....	37
Displaying messages	37
Displaying variables and text defines in a message box	38
Adding a dialogue topic	39
Initiating and ending dialogue.....	40
Multiple choice – asking questions	41
Adding to the journal and testing journal entries	41
Special dialogue-only functions	42
<i>Faction, Race and Rank.....</i>	43
Determining the PC's status:.....	43
Modifying faction standing and reaction	44
<i>Combat.....</i>	46
Initiating and ending combat	46
Detecting Attack	46
Keeping track of kills and knockouts.....	48
<i>Crime.....</i>	49
Jailing the PC.....	49
Clearing the PC of crime	49
<i>Magic</i>	50
Limiting the use of teleport	50
Checking and managing souls and soulgems.....	51
Casting spells	52
Managing and testing for spells	54
Managing and testing spell effects	55
Testing disease	55
<i>Sound</i>	56
Letting Actors talk	56
Playing music	56

Playing sounds	56
Controlling sound.....	57
Sound file formats:.....	57
<i>Keeping track of time</i>	57
Timer	58
Morrowind's time related global variables.....	58
Keeping track of days passed.....	59
Moon phases.....	59
<i>Working with objects.....</i>	59
Working with inventories	59
Adding and removing items from the inventory	59
Dropping items.....	60
Checking for presence of items in inventory	61
Monitoring inventory activities.....	61
UsedOnMe	62
Repairing objects.....	62
Equipping an Item – equip and PCSkipEquip.....	62
Placing an item near the PC	63
Checking activation of an item and activating it.....	64
Locking and Unlocking doors or chests	64
Enabling and disabling objects	65
Don't save changes to an object	66
Moving and rotating objects.....	66
Moving along an objects axis.....	66
Moving along the world axis.....	67
Rotating objects.....	67
Positioning an object in the world or in an interior cell	68
CellUpdate.....	69
Setting position or facing of an object for each axis.....	69
Resetting an object to its original position	70
Animating Objects.....	70
Determining location.....	72
Determining the players Cell.....	72
Determining when the PC leaves a cell.....	72
Distance of one object to another.....	73
Determining an objects position and facing	73
Line of Sight:.....	74
Determine whether an Actor is detected by another Actor	75
Triggers for Actors standing on an object.....	76
Hurting an actor standing on an object.....	76
<i>Changing and testing Skills, Attributes, AI and other Stats.....</i>	77
Get, Set, and Modify stats.....	77
Full list of Get-, Set-, and Mod-“Stat” type functions:	78
Attributes:.....	78

Health, Magicka, Fatigue:	78
Skills:.....	78
Magic:.....	79
Other:.....	79
AI Settings:.....	80
Resurrecting a dead actor.....	81
<i>Player Controls</i>	<i>81</i>
Player sleeping	81
Enabling and disabling player control	82
Disable control functions:	82
Enable control functions.....	83
Check control status	83
Force first or third person view	84
Functions for character generation menus	84
<i>Weather</i>	<i>85</i>
Changing weather	85
Changing weather settings for a region.....	85
Determining current weather.....	85
Detecting wind speed.....	86
<i>Miscellaneous functions and variables</i>	<i>86</i>
Making someone fall	86
Determining if player has menus open.....	86
Using MenuTest to close inventory.....	86
Fading the screen in and out	86
Adding a location to the map	87
Detecting if player is indoors or outdoors.....	87
Breaking of script processing	88
Assigning random values to variables	88
Playing videos	88
Controlling global scripts	88
<i>New functions that come with TRIBUNAL.....</i>	<i>90</i>
Changes / fixes to Morrowind scripting :.....	90
Index of new Tribunal Script Functions:.....	90
Enable equipment sharing.....	91
PC Action functions	91
Scale Functions	92
Water Level Functions.....	93
Levitation Functions	95
Object Creation	96

Worn / equipped Object Information.....	98
NPC Movement Functions.....	100
Object Collision Functions	101
Leveled List Functions	102
Miscellaneous Functions	103
Square root	103
Explosion.....	103
Deleting a reference completely	103
Combat Readiness Functions	106
<i>New functions that come with BLOODMOON.....</i>	<i>107</i>
Index of new Tribunal Script Functions:.....	107
GetPCTraveling and GetPCInJail.....	108
Place items near an object	108
Set the werewolf attributes	108
Change the color of Secunda	109
Determine how many kills a werewolf has.....	109
Check to see if the creature is in werewolf form	109
Change to a werewolf.....	109
<i>Tips and tricks</i>	<i>110</i>
Script with style for safer scripting	110
Cleaning up your mod.....	110
Limits of the Script Editor.....	111
Making Actors switch between weapons.....	112
Little helpers	113
Testing for the presence of another Mod	113
Safely starting global scripts- avoiding the main script.....	114
Saving CPU time.....	114
Use sound to detect events	115
Large battles:	116
Targeted scripts: running "global" scripts tied to an object	116
A guide to making rideable objects	117
Selecting objects.....	118
Creating/Deleting objects	118
Falling off from objects.....	119
Collision detection.....	119
Savegame issue	120
Mannequins.....	120
Is she looking at me?	122

Cinematic sequence.....	124
<i>Troubleshooting.....</i>	<i>125</i>
General hints.....	125
The Console.....	125
Using the Console to check variables:	125
Using the Console to quickly test scripts:	125
Error messages, malfunctions and common causes	126
In the game, when script executes:.....	126
In game error messages:.....	126
In the editor	127
<i>Appendix</i>	<i>128</i>
Undocumented functions	128
Game units:.....	128
Magic Effect List	129
List of console commands	130
Variable-type functions:	132
Local variables that get set by the game:	132
Local variables that you can set as a flag:.....	132
Special Globals.....	132
Game Settings	132
<i>Index</i>	<i>140</i>

Foreword to sixth Edition

This latest edition corrects a number of errors and adds some additional info. I am sorry it took so long. This work was mostly done by other people on the forums and people who mailed me. A great amount of work was done by Striker, who compiled the info on the Bloodmoon functions. Many thanks to all of you! Many thanks also to all the users of MSFD who have given me feedback and made the whole work feel worthwhile, and to everyone who keeps making all these amazing mods for Morrowind!

Foreword to fifth Edition

It was a bit premature when I claimed to have "almost everything" covered in the last edition. I guess there were still about 15% of the functions missing. Ouch. This time I should really have all (well, lets say 99.5% ☺) of the functions listed, except for possible undocumented ones that I may have missed. Dave Humphreys online list of functions has been a great help in finding the missing ones (If you don't yet know his most venerable and still unrivaled TES site, go there now: <http://m0use.net/~uesp/index.shtml>). This means that until a major update comes out in the form of a patch or a new expansion, this may well remain the last update for a while (depending of course also on feedback to GhanBuriGhan@gmx.net).

What's new in the fifth edition? Well firstly as mentioned, many previously unlisted functions and a lot of additional info on the ones that were already listed. I would like to point you specifically to the new information added on the use of variables and text defines with MessageBox, the new companion share feature in Tribunal and the new findings on the AIActivate function, but there is a lot more. Since it's a problem for many newcomers, I extended the dialogue section, which is now almost a tutorial of its own. I expanded the coverage on the Get/Set/ModStat type functions, I think many people missed the list that was in the Appendix before. I have significantly extended the *Tips and Tricks* and *Troubleshooting* sections, which now cover many frequently encountered problems and some amazing tricks. The *Index* should now be much more complete. Several useful lists have been added to the Appendix, most notably a (still incomplete) list of Morrowinds Game Settings – doesn't have much to do with scripting, but it should be a very useful resource anyway. With one word, even long time readers and scripting veterans should find a lot of new and interesting information – I myself didn't know a lot of this stuff until recently, when I started collecting all this info from the many helpful souls on the forums.

Many people indeed have contributed to this issue; I thank all those who responded to my questions and pleas for input on the forums. Special thanks to Shoujo, Riiak, Rhyek, Horatio, Ragnar_GD, Nazz, Eldar Mayiere, Striker, B, Shadowsong, Raptormeat, MadMax_001 and many others for their input and encouragement. There can be no doubt that the MW community is an excellent bunch! Specifically I would also like to thank Iudas, maxpublic, Wakim and LDones who have pooled their knowledge to compile the game settings list. Last but not least, I am very grateful to my proofreaders, Sheikizza, Iudas and Riiak – they have helped to greatly improve this manuscript, whatever mistakes and spelling errors still remain are (of course) entirely my fault (since I keep fiddling with the text)!

I hope that this guide will remain the number one resource for all those interested in scripting for Morrowind, and I wish all of you lots of fun while using it,

GhanBuriGhan

Disclaimer: All things TES, Morrowind, Daggerfall, Arena, Tribunal, the TES Construction Set etc. are property of Bethesda Softworks, a ZeniMax Media company. The author of this manual makes no claims to these names and trademarks. In all other respects I maintain the copyright for this document. This guide is fully unofficial and in no way am I, nor is Bethesda, responsible for any damages or loss of data, patience or sanity inflicted through the use of this manual. You can freely distribute this document as long as you keep it intact and unmodified.

Introduction

Using this guide.

If you are new to scripting, you should probably start by reading this introduction and especially do the tutorial. It is my best attempt at explaining scripts, what they do, and how to program one, in simple terms.

Secondly this is a manual, a reference, a handbook. The bulk of this document is a documentation of the available functions. This part is not written for the beginner, you are expected to know the basics of scripting already. I have however tried to provide a lot more info, explanations and sample scripts than the original helpfile, to make it easier to use these functions correctly in your scripts. Use the index at the end of the document to find info on a specific function, or the table of contents to find functions related to a general area of interest. If you run into errors, the Troubleshooting section may help you a bit. In the Appendix you find lists that may also come in handy as a reference.

Thirdly, as an advanced scripter you may also find the Tips and Tricks section interesting that covers both basic advice and advanced scripting techniques.

Finally a word of advice: don't take what's written here as gospel. The info in this guide represent the best of my knowledge and forum wisdom, but that doesn't mean that there aren't mistakes and oversights, etc. E.g. if I write that a function doesn't take variables as arguments it *probably* doesn't – but if its important for your mod, by all means try anyway. It may have changed with a patch or maybe simply nobody checked before. So run a test, and if you find something new, let me know, and I will add the info in a possible future update.

What is a script?

Scripts are basically pieces of code written in a special scripting language (I will call it TES script from here on). These little "programs" will run during the game and can perform certain things in the game, lots of things actually: Trigger events, control time and place, make things and creatures vanish, appear or move, give messages to the player, change stats, even change the weather – the possibilities are great.

TES Script is a unique scripting language, it is not used outside the TES Construction Set. As a scripting language it has certain limitations compared to a "real" programming language, like, e.g. C++:

1. The scope of TES Script is limited– don't expect to be able to program anything that is not already in the game in one way or another – which is not to say you can not achieve new and unusual things with scripting! But you can't use TES script to, say, program a word processor.
2. TES Script is also not an SDK (software development kit) that lets you actually work with and change (parts of) the games source code. That's why you can't use TES script to, for example, add new weather-effects. Those are hardcoded and you would need to change the actual game .exe to do that.
3. It's an interpreted language not a compiled one – the code needs a separate program (in this case Morrowind) to run – unlike compiled code that could be run by itself, like an .exe application.

What can scripts do?

Scripts for Morrowind are a way to have the game dynamically react to what the player does in the game world. You can use scripts to manage complex quests. You can use scripts to create custom items that perform actions beyond what regular enchantments could. You can use scripts to create traps. You can use scripts to change NPC or creature behavior.

Remember the character creation in Morrowind? It's basically controlled by a number of scripts. Seen Fargoth sneak around to his stash in Seyda Neen? That's a script directing his moves. Freed any slaves? That also is handled by a script. So the short answer to the question is: a lot.

What scripts can't do:

The TES script language is limited in its capabilities – there are only so many functions that you can use and sometimes the possible uses are not all that you may wish them to be. In fact some functions are buggy or plain broken. In many cases smart scripters can find workarounds for apparent limitations, but don't expect miracles. Many things are hardcoded and can not or only indirectly be influenced by scripts.

Scripting tutorial

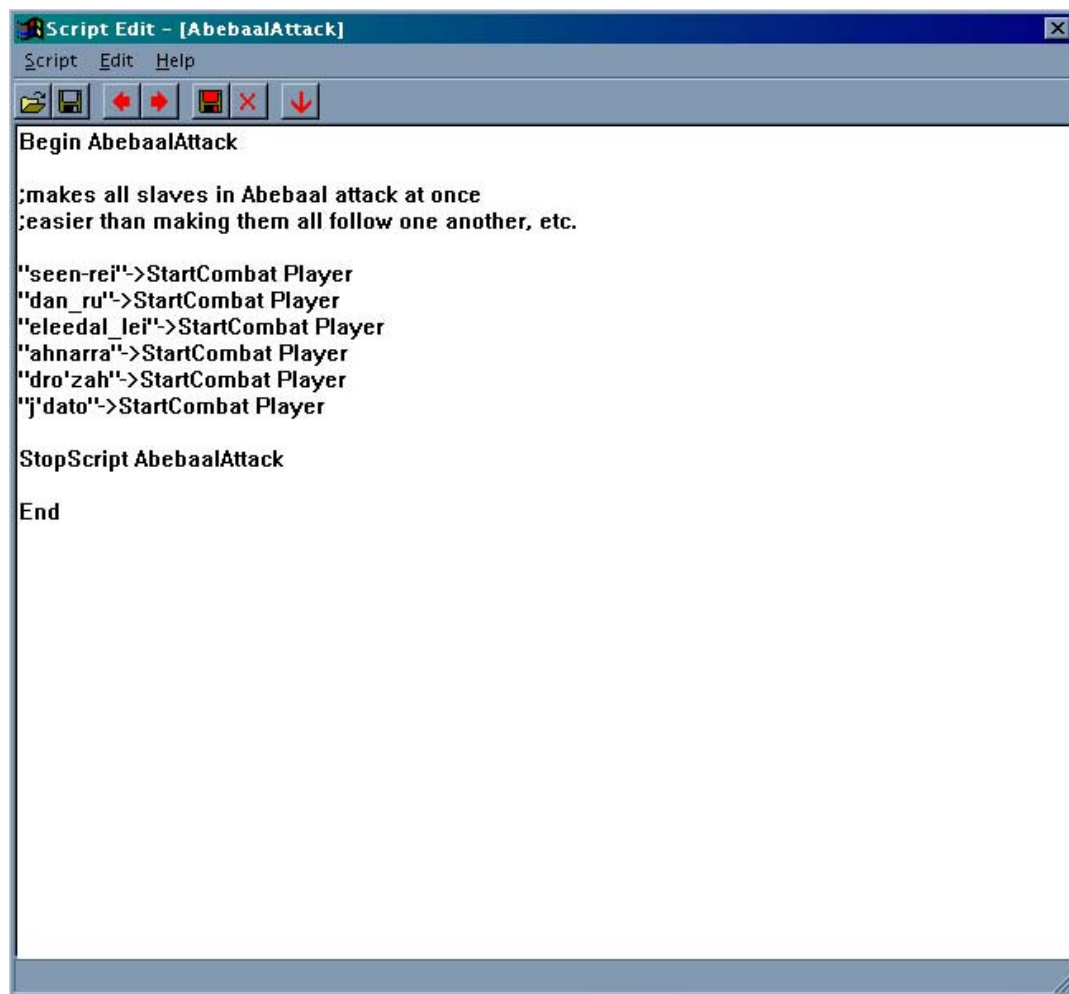
If you are completely new to scripting and programming in general, the thought of using TES script might be a little daunting – I have therefore written an extended tutorial that will walk you through making your first script. I will also explain the main elements of the scripting language as we go. There will be other explanations on the way, but the key instructions will be in **bold print**.

Let's get going!

We start by opening the script editor: **Start up the TES Construction Set, open the Morrowind.esm file and then select Edit Scripts from the Gameplay menu to open the scripting window.**

The scripting window

You enter the script editor either by **selecting Gameplay – Edit Scripts from the menu**, by clicking the edit script button (the pencil) in the taskbar or by accessing it from an Object or NPC dialogue, by clicking the button with the ellipses [...] next to the script field. The editor window is pretty basic:



Let's have a look at the buttons in the taskbar, from left to right: *Open* lets you select a script to edit. *Save* error checks the current script and compiles it or gives out error messages – note, however, that the plugin and thus the script is not really saved to disk at this time. When programming large scripts frequently use the save command in the main TESCS window after you have saved the script here, just in case the TESCS crashes.

Note that if you edit the script and suddenly hit "save plugin" to backup in the middle of the work, your updated script will NOT be saved with it. You must save it manually first. Also, if you just close script window, it doesn't mean that script will be saved. You must take care of it yourself (Thanks to Kir for this tip).

Forward and Backward arrows jump to the next or previous script, respectively (alphabetical order). If you give your scripts a common tag, that will make it easier to jump between the different scripts of your project, e.g. start every script name with AA_Scriptname this will put them right at the beginning of the list and keep them neatly together. *Compile all* recompiles all scripts (what's this good for? I don't really know). Finally, the *delete* button deletes a script and the last "arrow down" button closes the script window.

The help menu gives quick access to the function and command pages of the helpfile (of moderate utility, hence the creation of this manual!)

You can copy and paste into the editor window by using the windows standards ctrl-c for copy, ctrl-x for cut and ctrl-v for paste.

What do we want?

Before we really start writing our tutorial script we should decide what we want it to do. For this tutorial I decided we are going to make a **Riddle Chest: The chest will ask a riddle and only the right answer will open the chest. If the player provides the wrong answer, a trap will go off, hurting the player, and the chest can't be opened.**

That's a fairly complex undertaking, but we will take it step by step.

Writing a script

Ok, once you've got the Edit Script window open, **click into the main part of the window.** That is where you will write your script.

Naming a script: Begin and End

First of all we must give our script a name – every script must start with the declaration of this name. **So please type:**

```
Begin my_first_script
```

into the editor window. Note the underscores: Your name should be one word. Also note that the script language is not case sensitive, so begin could also be written without the capital letter. This name is the handle by which the script will be known in the TESCS. Try hitting the save button now: you will get an error message about "you need to end your script with end scriptname."

So, for the editor to recognize the script we also need to indicate an end: **next, write**

```
End
```

in a new line below the above. As you see we can omit putting the name of the script in this line again, just *end* will do. When you hit *save* now, you will see the name appear in the title bar of the script editor, indicating that the script has been accepted. This is the shortest script possible - and of course it doesn't do anything at all.

Detecting an action by the player

Next, we need a way to determine whether the player tries to open the chest. In TES Script we distinguish between Objects, Functions and Commands –

Objects are all the things in the game world, be they visible objects, creatures, NPCs or just sounds.

Functions are the all the "words" of TES Script that let us either manipulate these objects or let us gather information about them.

Commands are those "words" that structure the scripting language, but do not operate on any Game objects – an example is the word "Begin" we used to tell the script editor about the name of our script.

To tell the game which object it is supposed to perform a given function on we can use the "arrow": -> (really just a hyphen and a greater-then sign). You specify the object for the function on the left (we also call this the calling object) and the function to be performed on the right:

Object_ID -> function, [parameters]

A function may or may not have parameters. Parameters could be other object ID's, numbers and in some cases, variables.

What we need for our riddle chest is the `OnActivate` function: this is an informative function that tells us whether the player has "activated" an object in the game world or not. This function returns a value of 1 (which means "true" in programming terms) if the object has been activated, that means the player targets it and presses the "use" button (space, by default). So what we need to do is check if `OnActivate` becomes "true" anytime in the game.

So edit your script to look like this:

```
Begin my_first_script

If ( OnActivate == 1 )
    ; here we will enter what happens when the chest is opened
endif

End
```

A couple more things need to be explained here: The "*if*" command is there to check a condition – whenever the expression in the parantheses is "true" the following lines of code will be executed until the "endif" command is encountered. The "==" checks if an expression (in our case the "OnActivate" function) on the left of it is equal to the expression on the right of it (in our case to 1). If you forget the *endif* command after an if command, the editor will complain with an error message. The ";" semicolon denotes a comment – whatever you write behind the semicolon will be ignored when the script is run. If you ever write larger scripts you should learn to love this possibility.

Writing text and obtaining decisions from the player

Now we want our trapped chest to ask the player a riddle. For this we use the `MessageBox` function that allows us to display some text on the screen and also to display choices that the player can select from. Unfortunately Morrowind has no option to have the player type in the answer to our riddle, so we will have to give multiple choices. The line for that could read:

```
MessageBox "Voiceless it cries, wingless flutters, toothless bites, mouthless mutters. What is it?", "Bat", "Old woman", "Wind", "Wraith"
```

The first text is the text actually displayed in the box, the other texts, separated by commas tell the game to make "buttons", with the text given displayed.

But how do we avoid that the riddle is asked only the first time we try to open the chest and not every time? We now come to a very central point: the use of do-once conditions and state variables. Most of the problems that beginners encounter with scripting for Morrowind have their roots in misunderstanding how the scripts are actually executed and how scripts should accordingly be structured. So let's have a look at this.

How local scripts are executed

Every script that is attached to an Object or an NPC (local script) is executed *every frame the game displays on screen* while the cell with the object is active (indoors only the cell the NPC is currently in is active, outdoors the PC's cell and all adjacent cells are active). So the *complete script* (not just one line of it) is executed 10-60 times a second or however fast your computer runs the game! It is best to imagine every local script wrapped in a big "while-loop":

```
while (Object is in active Cell)

[Your script code]

endwhile
```

This is the reason why the following script spits out a continuous stream of messages (if attached to an Object or NPC in the same cell as the player). Try it, if you want:

```
Begin Message_script
MessageBox "Thousands of useless Messages"
End Message_script
```

This example is relatively harmless, but imagine what happens if you would use a line of code that adds an item to the players inventory, or places a monster next to him, etc.!

For this reason, "Do Once" constructions are very essential and something you will probably use a lot while scripting for Morrowind. So, let's go on with our tutorial script: we need to declare a variable and use it to make sure the message is only displayed once. **Change the script to the following:**

```
Begin my_first_script

Short controlvar

If ( OnActivate == 1 )
    If ( controlvar == 0)
        MessageBox "Voiceless it cries, wingless flutters, toothless bites, mouthless
mutterers. What is it?", "Bat", "Old woman", "Wind", "Wraith"
        Set controlvar to 1
    endif
endif

End
```

(Please note that the MessageBox command should be in one line in the editor!)

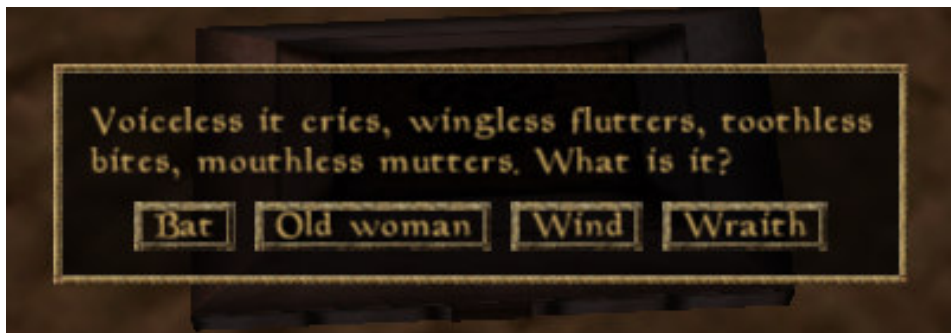
"Short controlvar" declares a new variable I called "controlvar", of type short. For the moment it's enough to know that this is variable that will contain integers (whole positive or negative numbers). A variable is a "placeholder" that can take on different values. The *if*

command we already know, the *set* command is new, but simple enough – it sets our variable that had the value 0 before (all variables start out at zero when declared) to 1. This, in connection with the *if (controlvar == 0)* command provides a do-once condition – the next frame the script is executed after the variable was set to 1 the if condition will be false and the message box will not be displayed again.

Now our script is already capable of being run, so lets test it:

- **Save the script and close the script editor window.**
- **Go to the TES construction set, Object window, select the container tab and open "chest_small_01".**
- **Change the ID of the chest to "tutorial_chest"**
- **In the script dropdown field, select my_first_script**
- **Save the object as a new object, save the mod, and quit the construction set. Start Morrowind and load a savegame.**
- **Now bring down the console (usually the ~ key, or whatever you have to the left of the "1" on the main keyboard) and in the console window, type:
PlaceAtPC tutorial_chest 1,1,1
and hit return.**

Take a step back (err, let your player character step back, that is!); you should now have a little chest sitting on the floor right in front of you. Clicking on it should bring up our message, which should look like this:



Clicking on those buttons will just close the messagebox for the moment, and clicking on the chest again, nothing should happen either – which is good, it means our do-once condition works.

Ok, leave Morrowind and go back to the editor, and load your plugin again.

We now need to figure out which answer the player selects, and script appropriate reactions for right and wrong answers. The function to test the selected answer is

"GetButtonPressed". This function returns a number depending on which of the buttons of a message box has been clicked on with the mouse. It will return "0" for the first button ("bat" in our example) and 1, 2, 3 etc. for the following buttons, in the order you listed them in the messagebox function. While no answer has been selected, the function will return -1, so we have to take care of that, too.

The "Activate" function will make our chest open, in fact activate will simply trigger the standard action that would usually be performed when you "use" the selected object – e.g. doors would swing open, NPCs would initiate dialogue etc.

The following update to our script also demonstrates how you can use a control variable to force MW to process functions one after the other, although the complete script is processed every frame of the game: simply increment the control variable and test it in a series of if –

elseif statements. This is a very safe way of scripting for MW – it may not always be necessary, but it's safe.

Please edit the script to the following:

```
Begin my_first_script

Short controlvar
Short button

If ( OnActivate == 1 )
If ( controlvar == 0)
    MessageBox "Voiceless it cries, wingless flutters, toothless bites, mouthless mutters.
What is it?", "Bat", "Old woman", "Wind", "Wraith"
    Set controlvar to 1
elseif controlvar > 1
    activate
endif
endif

if (controlvar == 1)
    set button to GetButtonPressed
    if ( button == -1 )
        return
    elseif ( button == 2)
        MessageBox "The answer was correct"
        Activate
        set controlvar to 2
    else
        MessageBox "The answer was wrong"
        set controlvar to -1
    endif
endif

End
```

Take a look at the part that starts with "*if (controlvar == 1)*". We have set controlvar to 1 as soon as the chest got activated. Now we test for which button is being pressed. We do this by assigning the new variable "*button*" a value returned by *GetButtonPressed*. Since the script is still running, even while the game seemingly pauses to await your decision, we first test if no button has been selected yet – return tells the game engine to stop processing the script for this frame.

Our correct answer was "wind" which corresponds to button number two – if button number two gets pressed, we will tell the player that he gave the right answer, and "activate" will open the chests inventory in the usual way.

All other values of button mean that the player has selected a wrong answer, so we can use the "*else*" command here. In this case we tell the player what a fool he was and the chest is not activated.

Now look at the little addition at the top of the script:

```
If ( OnActivate == 1 )
If ( controlvar == 0)
    MessageBox "Voiceless it cries, wingless flutters, toothless bites, mouthless mutters.
What is it?", "Bat", "Old woman", "Wind", "Wraith"
    Set controlvar to 1
elseif controlvar >= 1
    activate
endif
endif
```

This means that, whenever the chest is activated in the future, it will only open if controlvar is greater than 1. Check above: when the player provides the wrong answer in the riddle, controlvar is set to -1, so he will never be able to open the chest. But if he knew the right answer, *controlvar* is set to 2, and from now on the player can open the chest as often as he likes. Save and run your plugin, and test as described above.

Your first bug

Now, you will probably have noticed that the script does almost – but not quite – what we wanted. After clicking the correct answer, the chest's inventory doesn't open as intended. Now, the logic above seemed fine, so what is wrong? **Let's try the following (change the corresponding part of your script according to the fragment below):**

```
if (controlvar == 1)
    set button to GetButtonPressed
    if ( button == -1 )
        return
    elseif ( button == 2 )
        MessageBox "The answer was correct"
        set controlvar to 2
    else
        MessageBox "The answer was wrong"
        set controlvar to -1
    endif
elseif ( controlvar == 2 )
    Activate
endif
```

See how I moved the activate command to the section that tests for `controlvar == 2`? This provides a cleaner sequence of events, and as I mentioned above, this can be very important when scripting for Morrowind – always try to avoid doing too many things at once! **Well, run and test it.**

Great, now the inventory opens as we wanted, but what is this? The cursor is real slow, and we can't close the inventory! Look above – *controlvar* was set to two, and remains there, we do not change it again – therefore the game now gets continuous "Activate" commands each time the script is processed (every frame)! That's why we can't close the inventory – it gets reopened immediately. **So change the following part of the script:**

```
elseif ( controlvar == 2 )
    Activate
    Set controlvar to 3
endif
```

Test again: now everything works the way we wanted. I hope I have not confused you with the above excursion into the process of debugging, but it is a very important thing to know about – you will constantly have to rethink your scripts and try different ways of doing it to be successful.

What is still missing? The trap effect of course!

Putting a spell on the player

Our chest will put a curse on the player if he fails to answer the riddle.

First go to the spellmaking tab in the editor, right click and select "new". Give the spell the ID "Frost_Curse", name it "Frost Curse" and make it type "curse" then give it a magnitude of e.g. 1-5. It should look like in the picture below.


```

        Set timer to ( timer + GetSecondsPassed )
        if timer > 10
            Player -> RemoveSpell, "Frost_Curse"
            set controlvar to -2
        endif
    endif
End

```

Let's go over this. Player -> AddSpell, "Frost_Curse" puts the curse we created earlier on the player. Note how we need to use "Player -> " to make sure the effect really targets the player. Otherwise we would curse the chest (which is the default object, because the script is attached to it), which wouldn't make a lot of sense...

This bit:

```

Float timer
Set timer to ( timer + GetSecondsPassed )
    if timer > 10

```

...that is how you make a timer in Morrowind. *GetSecondsPassed* is a function that returns the time in seconds that has passed since the last frame. Since this is usually a fraction of a second (as the script gets called every frame), it is only natural that we need a float variable for this purpose – a variable that can store numbers with decimals. So when the timer has been running for 10 seconds we remove the curse again, and make sure we do this only once:

```

        Player -> RemoveSpell, "Frost_Curse"
        set controlvar to -2

```

Ok, save and test your mod. Works fine now, doesn't it? Well, almost. Try the following: let yourself be cursed, and then open your inventory. Wait. See how the curse terminates after some time, without hurting you? Of course: the script is still running, but spell effects are only calculated while in game, not while you are in the menu. We don't want the player to get off the hook so easily, so we need to put something in our script that stops it from processing when we are in menu mode. Luckily there is the *MenuMode* function, which returns 1 when you enter the menu. So we can put this at the start of our script:

```

If ( MenuMode == 1 )
    Return
Endif

```

Remember, return tells the game to stop processing the script for this frame. Ok, now we have our final working script. Congratulations! If you want, experiment a little more with this script. Put the chest into a location in the game and lock it. Then try unlocking it (*Unlock* function) with the script in addition to activating it. Try adding a sound, when the riddle is successfully solved (e.g. *PlaySound3D*, "skillraise"). Try using the *Cast* function instead of "AddSpell"

This is the final script:

```
Begin my_first_script

Short controlvar
Short button
Float timer

If ( MenuMode == 1 )
    Return
Endif

If ( OnActivate == 1 )
If ( controlvar == 0 )
    MessageBox "Voiceless it cries, wingless flutters, toothless bites, mouthless mutters.
What is it?", "Bat", "Old woman", "Wind", "Wraith"
    Set controlvar to 1
elseif controlvar > 1
    activate
endif
endif

if ( controlvar == 1 )
    set button to GetButtonPressed
    if ( button == -1 )
        return
    elseif ( button == 2 )
        MessageBox "The answer was correct"
        Activate
        set controlvar to 2
    else
        MessageBox "The answer was wrong"
        Player -> AddSpell, "Frost_Curse"
        set controlvar to -1
    Endif
elseif ( controlvar == 2 )
    Activate
    Set controlvar to 3
elseif ( controlvar == -1 )
    Set timer to ( timer + GetSecondsPassed )
    if timer > 10
        Player -> RemoveSpell, "Frost_Curse"
        set controlvar to -2
    endif
endif

End
```

General Information: Scripts, Commands and Syntax

Types of scripts

Local scripts

Any script that is running on an object or actor in the game (assigned in the script-dropdown field of the object or actors object window) is a local script. Local scripts are only active if the cell is loaded – this is the current interior cell, or the current and all directly neighboring exterior cells. When the object is outside of this range the script is not running, but the local variables are saved.

Global scripts

Any script that is not attached to any object is a global script, and is by default not executed until you call it (see below). Note that there is no default object for a global script to work on, so objects must always be specified: while the following will work in a local script attached to an NPC:

```
AITravel 1150, 8899, 1110
```

You will have to specify the NPC in a global script:

```
"NPC_ID" -> AITravel 1150, 8899, 1110 ;NPC_ID is the unique identifier for  
;each object in the editor
```

Global scripts are active all the time once they have been activated and until they are specifically terminated. Thus, once activated, they will be processed every frame as described for locals scripts above. That is why they should be used with caution, as too many, or too complicated global scripts can easily slow the game down a lot.

The command to start a non-active script is:

```
StartScript "Script ID"  
or  
StartScript, "Script ID"
```

The function to terminate a global script is:

```
StopScript "Script ID"  
Or  
StopScript, "Script ID"
```

Note: read up on the special case of "targeted scripts" in the Tips and Tricks section.

Syntax

There are some unique features in TES script that should be explained before we go into more detail.

Beginning and ending scripts

```
Begin Script_ID
End
End Script_ID
```

Every script must have the Begin and End tags. The specified name will also be the ID you will reference the script by (be it from other scripts or inside the TES CS object windows). A script may start out with comments, but the first line of real code must be "Begin xxxxxxxxx".

General syntax for functions:

```
"Object_ID" -> Function, [parameters]
```

This "arrow" or "fix" designates which object a function will be performed on. The "object_ID" is the unique ID that is given to each object in the editor (usually the first field in any object window). You need this ID, not the Name! If you call a function without a designated Object-ID, the function will be performed on the default object, which is the object the script is attached to.

This is not to say that there might not be another object referenced as a parameter:

```
"Object_ID1" -> Function, "Object_ID2"
```

Functions are not case sensitive, but using the case sensitive form as suggested by Bethesda (e.g. GetSpellEffects instead of getspelleffects) makes scripts easier to read, so it's recommended to keep it that way.

General syntax: Commas, parentheses and spaces

TES Script is not too picky about syntax. Case mostly doesn't matter, commas can be left out, and spaces are mostly ignored. Nevertheless I would advise adhering to the following principles:

- Use commas between parameters
- If the ID contains a space, you must use quotation marks: "Object ID" better to avoid spaces altogether: Object_ID
- Get used to **always** Leave a space after parentheses and operators, sometimes it seems to cause problems if you don't: if (variable == 1), not: if (variable==1)
Object_ID1 -> Function, Object_Id2, not Object_ID1->Function,Object_ID2
While this doesn't matter most of the time it generates weird and almost untraceable errors sometimes, so you are much better off to get used to always leaving a space.
- (By Dinkum Thinkum:) Leading underscores on Object IDs and variables have erratic results in the script editor: in some places they work fine, in other places the script editor chokes on them (usually with error messages that don't have anything to do with the actual problem...). For example:

```
PlaceItemCell, "_dt_Racial_ClothierNord", "Vivec, Agrippina Herennia: Clothier", -348, 48, -221, 0
```

works fine, but

```
_dt_Racial_ClothierNord -> PositionCell, -348, 48, -221, 0, "Vivec, Agrippina Herennia: Clothier"
```

causes error messages.

At least one of the standard guides or tutorials on modding recommends using leading underscores for Object IDs so they sort to the top of the Object window lists and thus are easy to find. I followed that advice, and so have a lot of other people (from looking

at other people's mods). Leading underscores for variables also are unpredictable in scripts: someplaces they work, other places they don't. So my policy now is to NOT use leading underscores for anything in the Construction Set.

Tabulators

For your own sake, use tabs for if-elseif constructs – makes it much easier to keep track of them, so you don't forget an endif at the end.

```
If ( variable1 )
    If ( variable2 )
        [do something]
    endif
endif
```

is better than

```
If ( variable1 )
If ( variable2 )
[do something]
endif
endif
```

Variables

Declaring variables

There are three types of variables in the TES script language: short, long and float. According to the manual these cover the following data ranges:

Short	-32,768 to 32,767 (signed integer)
Long	-2,147,483,648 to 2,147,483,647 (long signed integer)
Float	3.4E +/- 38 (float, 7 digits)

Apparently the boundaries for Long given here are only partly correct, in the TES-CS you can assign a maximum value of 2147483520 (Forum info / Argent).

Local variables these have to be declared in the script:

```
Float floatvarname
Short shortvarname
Long longvarname
```

Local variables are unique to a specific instance of a local script. This means that the local variables in multiple objects with the same script do not influence each other. The names you use for variables are pretty much up to you as long as they start with a letter, but you have to avoid using function names (this will result in errors during runtime) and reserved characters (e.g. -, +, /, *, =, ",), (, etc.) which will result in compiler errors. E.g. "variable-1" will not work as a variable name. Underscores as in "my_variable" are ok. A dot has a reserved meaning as well (see "*Referencing variables in other local scripts*" below).

Theoretically there should be string variables too, but to my knowledge these are not implemented. Unfortunately there are also no data types to store Object_Id's, which limits the power of the scripting language to a certain extent.

Variables can be grouped into *local* variables (valid only in the script that declares them) and *global* variables (valid in every script). To declare a global variable go to the Gameplay menu and select Globals. Right click for "new", name it, and set the type and starting value for the new global variable. Global variables are very useful for involved quests when you need to

keep track of things over an extended time and space. They are also a simple way to share information between different scripts

Note: if you declare a local variable with the same name as a global variable, the global variable will become invisible for this script. Do NOT declare a global variable as a local!

Referencing variables in other local scripts

If a unique object has a local script running on it you can change variables from outside the script in the following way:

```
Set MyObject.variable to 100  
or  
Set MyObject.variable to local_variable
```

This method changes a local variable in the object's script. The object must have a script on it for this to be valid. Note: The scripting system looks at the first object in the database, thus you should only reference objects that are unique (exist only once).

Note that the reverse does **not** work:

```
Set local_variable to MyObject.variable
```

Use a global variable to transfer information in this way, or set local_variable from the **other** script using the syntax above.

```
if ( anotherobject.x > 0 )
```

apparently works.

Furthermore I realized only recently that this syntax also works for global scripts:

```
set Global_script_name.variable to 1
```

This is useful to avoid using more global variables than necessary or also as a console command to debug global scripts.

Using variables in functions

Unfortunately it is one of the limitations of TES script that only certain functions accept variables as parameters, which poses some definite limits. This is indicated in the list of functions below. For example the Position and PositionCell commands do not accept variables, so teleporting works only to predefined locations.

Note: For some functions where both a Get-Function and a Set-Function exist, a workaround can be constructed by using the while function (see below).

Mathematical calculations

You can use the standard operands in set commands (and probably other places, but I never tried) to do calculations in scripts

Addition: +
Subtraction: -
Multiplication: *
Division: /

The syntax is as follows:

```
Set result_var to (var_a + var_b)
```

Instead of variables, literal values are also allowed. I assume that standard mathematical rules are applied. You can also use parantheses according to standard mathematical rules:

```
set ln to ( ln + ( k10 * math_ln10 ) + ( k2 * math_ln2 ) )
```

A warning: there appears to be an issue with very long additions (e.g. adding up more than 20 variables in one line of code) that causes a mod to crash the game upon loading. If this happens, split the calculations to several lines. <I have had VERY bad luck using more than one operator on a line, but maybe that's just me.>

There isn't much in the way of dedicated mathematical functions in MW script. There is the 'Random' function and Tribunal added the 'GetSquareRoot' function (see below). If you need more complex functions, I suggest downloading Soralis' Math Mod (available from Morrowind Summit). It's a collection of scripts that allow you to do complex calculations. Here is a short excerpt from the readme to give you an idea:

"This mod adds the ability to use various math functions within Morrowind's scripts. Specifically, these are the scripts that are added:"

Name	Check/Done	Inputs	Outputs	Accuracy
MathScripts	N/A	N/A	N/A	N/A
MathConstants	N/A	N/A	N/A	N/A
SquareRoot	1	math_sqrt	math_result, math_imag	7
SineScript	2	math_angle	math_sin, math_cos, math_tan	7
ArcsineScript	3	math_arc	math_sin, math_cos	6-7
NaturalLog	4	math_log	math_result, math_imag	4-5
LogScript	5	math_log, math_base	math_result, math_imag	3-4
intPower	6	math_value, math_power	math_result	7
intRoot	7	math_value, math_root	math_result, math_imag	6-7
Modulus	8	math_value, math_mod	math_result	6-7
AntiLn	9	math_log	math_result	4-5
Antilog	10	math_log, math_base	math_result	2-3
AbsoluteValue	11*	math_abs	math_abs	7
PowerScript	12	math_value, math_power	math_result, math_imag	2-3

Testing conditions

Use of if... elseif conditions

Much of TES scripting relies on the use of *if... elseif* conditions and variations thereof. It is very important to fully understand these, and the conditions that can be used in them to test conditions of the game world to trigger events.

In general a condition is "true" when it has the value of 1 (or simply "not 0" (e.g. -1 is "true" in TES Script too) and "false" when the value is 0. There are thus certain functions in the game that return "true" under certain conditions. For example see the `GetAIPackageDone` function further down. It returns "true" (= 1) for one frame when an AIPackage has finished. Thus:

```
If ( GetAIPackageDone )  
    ...  
endif
```

and

```
if ( GetAIPackageDone == 1 )  
    ...  
endif
```

are equivalent. The second syntax tests a condition and returns "true" if the condition is met. You can check for the following:

"Equal to":	==
"Unequal to"	!=
"Greater than"	>
"Smaller than"	<
"Greater than or equal to"	>=
"Smaller than or equal to"	<=

You can also use `else` and `elseif`. `Elseif` tests for a separate condition while `else` will be executed if none of the previous conditions are true

```
If ( foo_var == 1 )  
    [do something]  
elseif ( foo_var == 2 )  
    [do something else]  
else  
    [do something completely different]  
endif
```

In my experience it is safest to use *elseif* instead of multiple separate *if* statements if you test different states of one variable.

There is a maximum of the total *if-elseif* conditions that can be processed in one script. I am not sure but I think it's around 256 conditions.

While conditions

```
While ( condition )  
; things to do  
EndWhile
```

The `while` command differs from the `if` command in that it is repeated within one frame until the condition is fulfilled. This is best explained with an example:

```
Short desiredAmnt  
  
SetStrength 0  
while( GetStrength < desiredAmnt ) ; non-literal value to match  
    modStrength 1  
endwhile
```

This will set strength to the value in variable desiredAmnt after one frame. The following script however would need an undetermined amount of frames to do this, because the if condition is only called once each frame:

```
if (getStrength < desiredAmnt) ; non-literal value to match
    modStrength 1
endif
```

On the other hand the first example can potentially cause "freezing" (if the value would be very high) while the second won't.

Note that this is a workaround for some functions to the problem of functions not accepting non-literal values (variables) as arguments.

Constructing Boolean operations in TES Script

Unfortunately there are no Boolean operators (AND, OR, NOT, XOR, ...) in the scripting language. Thus you need to construct these yourself using if... elseif structures.

Instead of AND:

```
if ( variable1 AND variable2 ); does not exist
    [do something]
endif
```

you have to use:

```
If ( variable1 )
    If ( variable2 )
        [do something]
    endif
endif
```

For OR constructs:

```
if ( variable1 OR variable2 )
    [do this]
endif
```

you can use elseif constructions:

```
If ( variable1 )
    [do this]
elseif ( variable2 )
    [do this]
endif
```

List of TES-script functions

Explanation of the format:

First I will list the function and the arguments it takes:

Code "string", var_enum, var_float, [optional] (returns short)

Arguments of the function: "string" indicates a literal string, such as an object ID. Var_enum indicates a literal value (no variables taken) var_float indicates a variable of the specified type (in this case float). Brackets [] indicate optional parameters. (returns short) or (returns float) indicate that the function returns a value and of what type the value is. I will use the designation (returns Boolean/short) to indicate a function that returns either 1 or 0 (a Boolean variable although its strictly speaking still a float)

Examples of usage follow in italics and are indented:

Code "ID", var_enum, var_float

Example scripts are set in frames:

```
Begin script
[Script functions]
End script
```

Scripting NPC's: AI and Movement

Make an NPC walk to a new location

AiTravel, float_enum_x, float_enum_y, float_enum_z, [reset]

To make an NPC walk between different defined places in the game world you use the AiTravel function.

The variables x, y, z are world coordinates. You can determine these by moving your camera to the desired endpoint of the movement or by selecting a path grid point or an object nearby coordinates are displayed below the object window. The usage of the optional reset flag is unknown.

When using this function in scripts it is important to provide conditions where the AI package is called only once. Consider the following NPC bound script

```
Begin Travel
AiTravel, 1359, 2700, 1045
End Travel
```

The previous script will not work, as the script “fires” continuously, and the effect is that the NPC freezes and never performs the desired movement.

```
Begin Travel
Short do_once

If (do_once==0)
    AiTravel, 1359, 2700, 1045
    Set do_once to 1
endif

End Travel
```

This should work; the NPC will wander to the designated coordinates as soon as his script becomes active, meaning as soon as his cell is loaded.

Checking whether an NPC has performed his movement

`GetAIPackageDone` (returns Boolean/short)

```
if ( GetAIPackageDone == 1 )
    [do something]
endif
```

To check whether an NPC has arrived at his location you can use the *GetAIPackageDone* function. This function returns 1 for one frame when the current AI package has finished.

Use this to perform a check whether a movement has been finished. The following script is an example of how to link several *AITravel* commands within one script, using a state variable and an *if-elseif* structure:

```
Begin TravelLoop

short state
float timer

if ( menumode == 1 ) ; if menu is open don't process
    return
endif

;start walking
if ( state == 0 )
    if ( player -> GetDistance HB_adros_darani < 5000 )
        set state to 5
    endif
endif

;***** He begins his trip
elseif ( state == 5 )
    SetHello 0
    AITravel -8144, -19409, 728 ;new co-ords point 1
    set state to 10
endif

elseif ( state == 10 )

    if ( GetAIPackageDone == 1 ) ;he's reached point 1
        set state to 40
    endif
endif

elseif ( State == 40 )

    AITravel -9147, -19459, 720 ; new co-ords point 2
    set State to 50
endif

elseif ( state == 50 )

    if ( GetAIPackageDone == 1 ) ;he's reached the point 2
        set state to 60
    endif
endif
```

```

endif
elseif ( state == 60 )
    AITravel -8144, -19409, 728 ;new co-ords point 1
    set state to 70
elseif ( state == 70 )
    if ( GetAIPackageDone == 1 )          ;he's reached point 1
        set state to 80
    endif
elseif ( state == 80 )
    AITravel -6640, -18496, 1040 ;new co-ords point 0
    set state to 90
elseif ( state == 90 )
    if ( GetAIPackageDone == 1 )          ;he's reached point 0
        set state to 0
    endif
endif
End TravelLoop

```

Good examples of scripting using *AITravel* are also found in the “lookoutscrip” (Fargoth hiding his treasure) and the CharGenWalkNPC script (The guard that walks you through the ship at the beginning of the game). Or look at my "Traveling Merchants" plugin for true *AITravel* madness ☺ !

If you plan on using this function extensively you should be aware of the following problems: When you leave a cell with an actor that is just performing its *AITravel* command, or if you rest, the script will never detect the *GetAIPackageDone* signal, meaning your NPC gets stuck on its path once you return or finish sleeping. The following simple code can be used to get the script going again (this is for the above script)

```

; ***** Stalled Script rescue - recovers script after leaving a cell or resting
If ( Player -> GetDistance, HB_adros_darani < 5000 )
    if (GetCurrentAIPackage == -1) ; check for idleness
        set timeout to ( timeout + GetSecondsPassed )
        if ( timeout >= 3 ) ; wait some time.
            ; Short instances of idleness always occur
            set state to (state - 10) ; stall will occur at
            ; AIPackageDone - jump to "wander" again.
            set timeout to 0
        endif
    else
        set timeout to 0
    endif
endif
endif

```

Facing

Face x_float, y_float

Makes an NPC face towards a certain x/y coordinate. Apparently interrupts current animations. Using this on wandering NPCs makes them stop, face wherever, then continue on wherever they were wandering to, as soon as the facing movement is over. (Forum info / JOG, Dan_Wheeler).

To make an NPC constantly watch the player:

Example:

```

...
set px to player->getpos x
set py to player->getpos y
face px, py
...

```

Random Actor movement

```
AiWander, range_enum, duration_enum, time_enum, [idle2], [idle3], ...[idle9], [reset]  
"Actor_ID" -> AIWander, 512, 5, 0, 0,20,0,0,10,30,0,0
```

This is the random movement algorithm that most of the NPCs in the game are using. The NPC travels along the path grid, changes direction randomly, and performs idle movements in between.

- **Range:** determines the distance the Actor or creature will roam from his origin.
- **Duration:** probably the time (in hours) the mob will perform the package (before it is reset, which seems to happen if you leave or sleep, not sure?)
- **time:** presumably determines the start time for the package if it has a duration
- **[idle2], ...[idle9]:** chances of idle movements.

The Idles are:

Idle2: Looking around

Idle3: Looking behind

Idle4: Scratching head

Idle5: Shifting clothing or armor on shoulder

Idle6: Rubbing hands together and showing wares

Idle7: Looking at fingers and looking around furtively

Idle8: Deep thought

Idle9: Reaching for weapon

To let an Actor stand in one spot you can use: `AIWander, 0, 0, 0`

Activating Objects

```
AiActivate "Object ID"  
AiActivate , ObjectID, [reset]
```

In the words of Bethesda: "This package tells the actor to activate the specified ObjectID. A powerful and admittedly underutilized and undertested package."

In standard Morrowind this function appeared to be mostly **broken**, except for making an NPC quaff a potion. **With Tribunal** it seems to have been fixed, at least to some extent. I could successfully use it to get an NPC pick up a weapon, open a normal door and go through a load door. LoadDoors only work if the door marker it teleports to is in the same interior cell, or within the loaded (PC's current and surrounding) cells in exterior cells – otherwise the game crashes. I also tested it successfully with an activator (switch to open Ghostgate). The usual precautions with AI functions apply (make sure the actor is not too far away, have a good AI grid in place, make sure nothing can get in the way, etc.). Although not thoroughly tested, I didn't seem to get an *AIPackageDone* signal, but other conditions can be constructed (see examples below) to set the NPC to a different AIPackage again.

Object Type	Activation
NPC	Dialogue
Container	Opens
Door	Opens
Load door	Opens/teleports (same cell only)
Weapon, armor, misc., etc	Picks Up
Book/Scroll	Reads <meaning what to an NPC?>
Activators	Execute as defined by script

Sample Script: these are just a few testing scripts I made. They show how you can set conditions to determine when the NPC has finished his action.

```
Begin TT_opendoor
short doonce
short AIState

if ( doonce == 0 )
    if ( GetDistance, Player < 400 )
        AIActivate TT_door
        set doonce to 1
    endif
elseif ( doonce == 1 )
    set AIState to GetCurrentAIPackage
    MessageBox "Package = %g", AIState
    if ( TT_door->GetAngle, z != 180 ); As soon as door starts rotating
        MessageBox "Done"
        AIWander 30, 5, 0, 0,20,0,0,10,30,0,0
        set doonce to 2
    endif
endif

end
```

```
Begin TT_pickmace
short doonce
short AIState
if ( doonce == 0 )
    if ( GetDistance, Player < 400 )
        AIActivate TT_daedric_mace
        set doonce to 1
    endif
elseif ( doonce == 1 )
    set AIState to GetCurrentAIPackage
    MessageBox "Package = %g", AIState
    if ( GetItemCount, TT_daedric_mace >= 1 ); when NPC has the mace in his inventory
        MessageBox "Done"
        AIWander 512, 5, 0, 0,20,0,0,10,30,0,0
        set doonce to 2
    endif
endif

end
```

```
Begin TT_openloaddoor
short doonce
short AIState

if ( doonce == 0 )
    if ( GetDistance, Player < 400 )
        AIActivate TT_door
        set doonce to 1
    endif
elseif ( doonce == 1 )
    set AIState to GetCurrentAIPackage
    MessageBox "Package = %g", AIState
    if ( GetPos, y > 2000 ); loaddoor target was in same cell for this script.
    ;Other conditions like GetPCCell could be used as well.
        MessageBox "Done"
        AIWander 30, 5, 0, 0,20,0,0,10,30,0,0
        set doonce to 2
    endif
endif

end
```

Following

```
AiFollow, "Actor ID", duration_f_enum, x_f_enum, y_f_enum, z_f_enum, [reset]
AiFollowCell, "Actor ID", "Cell ID", duration_f_enum, x_f_enum, y_f_enum, z_f_enum, [reset]

"MobID" -> AiFollow, "Mob2ID", 0,0,0,0
```

The "Follow" AI-packages makes an Actor closely follow another. You can use this to make an NPC or creature follow the player, but you can also use it to make NPCs and creatures form a caravan. The following excerpt from one of my own scripts shows the unconditional use of the function:

```
elseif ( state == 20 )
    HB_guar_pack_adros_ -> AiFollow, HB_adros_darani, 0, 0, 0, 0
    AiTravel -8144, -19409, 728 ;new coords point 1
    set state to 30
```

since there is no duration or destination location given, the guar will follow the NPC until another command is given. As with other AI-commands, make sure you set conditions so that each AiFollow command is issued only once, not each time the script is executed every frame.

The duration, CellID and x, y, z destination coordinates set conditions that once fulfilled will terminate the AIPackage (which you can test with the GetAIPackageDone function as describe for AiTravel above). The AiFollowCell function allows you to set an interior cell as the destination.

The meaning of the optional reset option is currently unknown.

```
AIEscort, "Actor ID", duration, x, y, z, [reset]
AIEscortCell, "Actor ID", "Cell ID", duration, x, y, z, [reset]
```

This function allows to program an Actor to lead the player to a certain point. The actor will wait for the player if the distance becomes too great, and will resume its path when the player approaches again (Thanks to Kir for this info). You see an **example** of this in character generation -- the guard who escorts you up from the ship's hold to the ramp on the second level. Escorting actors will stop & wait for you if you get too far behind (Thanks to MisterSmileyFaceDude).

Checking which AI package is currently executed

For complex Actor scripting it might be valuable to know which AI mode is currently active and make script actions dependant on it.

```
GetCurrentAIPackage (returns short)

If ( GetCurrentAIPackage == 2 )
    [do something]
endif
```

The returned values are:

None	-1
Wander	0
Travel	1
Escort	2
Follow	3
Activate	4
Pursue	5

Forcing sneak movement

ForceSneak
ClearForceSneak
GetForceSneak (returns Boolean/short)

The *ForceSneak* command puts the actor in sneak mode, all movement will be performed in sneaking. *ClearForceSneak* terminates *ForceSneak* mode. Unfortunately there does not seem to be a corresponding command to force running (added with Tribunal). *GetForceSneak* returns one if *ForceSneak* mode is active on the calling actor. Check the LookoutScript script for an example. Here's a snippet:

```
elseif ( walkstate == 2 )
    Fargoth->ForceSneak ; enter sneak mode
    Fargoth->AiTravel -11468.595,-71511.531,173.728 ;goes to tree
    set walkstate to 3

elseif ( walkstate == 3 )
    if ( Fargoth->GetAiPackageDone == 1 )
        ;Fargoth->Equip "torch_infinite_time_unique"
        set walkstate to 4
        ;MessageBox "SHOULD BE AT TREE"
    endif

elseif ( walkstate == 4 )

    set timer to timer + GetSecondsPassed

    Fargoth->ClearForceSneak ; terminate sneak mode
    Fargoth->AiWander 0 0 0 0 0 0 0 0

        if ( timer > 3 )
            Fargoth->ForceSneak ; reenter sneak mode
            Fargoth->AiTravel -11410.590,-72057.188,133.644 ;goes to wall
            set walkstate to 5
        endif

endif
```

Text and Dialogue

Many of the following functions are not only used in scripts, but also in the result field of the dialogue editor window. Dialogue is an art in itself that can not be fully covered here, but I will give a very brief introduction.

Brief dialogue how-to

The concept of MW dialogue

In Morrowind, dialogue is organized in a database. This database is structured in the following way:

Top level:

The different "divisions" of dialogue are:

Topics: The actual topic words and responses for the dialogue window in the game.

Greetings: The text strings you are greeted with when you initiate dialogue with an actor.

Persuasion: The answers you get for successful or failed persuasion attempts.

Journal: The entries for your journal.

Voices: The .mp3 sound files (and subtitles) that players "say" when you come near, are hit, are fleeing, etc.

Sub level 1:

Each of these top level divisions has sublevels I call topics – for the *Topics* division, these are the actual "keywords" (topics) that actors will have something to say about. These are the words that will be highlighted ("hyperlinked") in in-game text and listed in the right hand

panel of the dialogue window. For *Journal* these are the different journals (usually one per quest). For *Voices* the different categories of sound responses that are "triggered" by the appropriate in-game conditions etc. For *Greetings* these are just general categories of greetings (diseased, quest offers, standard and so on). For *Persuasion* these are things like serices refusal or bribe success/fail messages.

Sub level 2:

For each topic there can be one or several sub-entries – these are the actual responses. For *Topic* and *Greetings* these are the actual answers of actors to the topic phrase. For *Journal* these are the different entries describing the progress of a quest.

These entries represent a linked list, meaning that each entry contains (invisible) info on which is the next response in line and which is the previous. (This leads to a common error message when dialogue gets deleted or scrambled, e.g. by cleaning a mod with TESAME, or by loading several mods that change the same topic. The game doesn't know for sure any more which entry comes after which. This sometimes doesn't matter, but it can also mean that certain responses are cut off, because the order isn't right any more – this depends on the conditions).

These entries come with conditions that you can set in the dialogue editor window. In the topic window you will find a list of general conditions on the left – here you can define which actor (Actor ID) or group of actors (Race, Class, Disposition, etc.) potentially know this response. There are also two conditions for the PC (PC faction and rank). To the right you can set a maximum of 6 "free" conditions that can refer to the actor, the PC or other things like the state of global variables – lots of options here, you will have to see for yourself. Check for journal entries, player stats, local or global variables, items in inventory, and many other functions, some equivalent to script functions, some unique to dialogue (see below).

An important and initially confusing feature of the dialogue editor is the filter option (lower left-hand corner). When you select an actor ID here, you only see the topics that this actor can possibly know (as described above). Remember that when you create a **new** topic (maybe specifically for that NPC) it contains no responses. Thus the actor cannot "know" it and thus the freshly created topic does not appear! Select the empty slot on the very top to see all topics again, make a response for the new topic that your actor can "know" – then you can turn the filter on again, if you wish.

How dialogue works

To decide whether a dialogue **topic** is available during dialogue with an NPC, the game checks:

1. Whether the NPC "knows" the topic – this is determined by the conditions set in the "speaker conditions" field – if the NPC can fulfill the conditions for just one of the responses for that topic, he "knows" it.
2. Whether the Player knows it. The PC knows a topic if it (the topic word) either appeared previously in a response from an NPC or if it was specifically added with the AddTopic function.
3. If both the PC and the actor "know" the topic it shows up in the topic list and gets highlighted in the text – it can now be selected by the player and an appropriate response will be given.

When the game has to select the correct response from the list of responses for that topic, it does the following:

- It starts checking **from the top** of the list, whether the conditions for that response are met, which means **all** conditions that have been defined for that response return "true".

- If not -> Game moves on to the next item in the list and checks again.
- If yes -> this entry is used and printed to the dialogue window.

Special rule for greetings: if none of the items in the list are met, move on to the next level 1 item (e.g. if no greeting in "Greeting 0" returns true, start checking "Greeting 1").

Special rule for Journal: there is only one condition here, the index (called by `Journal` function). It must be exactly met.

Once a response has been selected, the game will

- Output the text string to the dialogue window (or play the mp3 for *Voice* responses)
- Execute any Script commands in the Results field (At the very bottom). You can use all scripting functions here. Conditions (*if* command) can also be used (Thanks to Kir and Manauser for this info).

Be aware that the result field allows you to exchange information with scripts (e.g. by setting variables or adding journal entries) and that scripts can vice versa influence dialogue as well, by setting conditions that can be tested (e.g. you can check for local or global variables in the speaker conditions of dialogue). The simplest script that influences dialogue is the `nolore` script, which is just used as a flag to keep actors from using standard dialogue.

A few golden rules:

- The **most specific** responses should be **on top** of the list, the "catch all" answers should be lowest! Remember the first one that returns true is the one that gets picked. So you can't have a response for everyone in Vivec above one for a specific NPC in Vivec.
- If you want an NPC to be able to talk with the PC about something special, you must introduce the topic word, e.g. in a greeting or in a "latest rumors" response. Alternatively, you can use a script with the `AddTopic` function.
- Topic, Greeting, Journal are actually all in the same database – that's why journal topics use a format like `A1_dreams`. If it were just "dreams" than the journal entry to "dreams" could come up as a dialogue response to the word "dreams".

Dialogue-related functions

Displaying messages

`MessageBox, "Message", [var1], [var2], ["button1"], ["button2"]`

The `MessageBox` command lets you give out information to the player. Normally these appear as a small box with the text on the bottom of the screen that stays there for a few seconds or until the player has clicked a button if the message box has buttons. If a dialogue window is open, `MessageBox` will output to the dialogue window! This will be in a different color so it's a good way to say show that text isn't part of dialog. For example, "Okay I'll take the curse off. *He takes the curse off.*" `MessageBox` has several different modes of operation. The simplest one is just giving out an onscreen message that appears on the bottom of the screen for a few seconds, as in the following script that gives out a message when the item it's attached to is equipped:

```
Begin informplayer
Short OnPCEquip

if ( MenuMode == 1 )
    return
endif
```

```

if (OnPCEquip ==1 )
    MessageBox, "The sword vibrates in your hand"
    Set OnPCEquip to 0
Endif

End informplayer

```

The second mode of operation makes the message stay on the screen until the player presses a button:

```

MessageBox, "Ulyah lifts her hands and speaks the formula. You will now be transported to Sheogorad", "ok"

```

In the third mode of operation you can use the messagebox to demand a decision from the player via a message box with buttons and the *GetButtonPressed* function:

GetButtonPressed (returns short)

Pressed button if a message box with buttons is used, starting at 0. Will return -1 until button is pressed.

Sample Script:

```

Begin choices

Short button
Short status
Short OnPCEquip ;declare as variable - otherwise there will be errors

if ( OnPCEquip ==1 )
    MessageBox, "The sword vibrates in your hand. Do you want to equip it?", "yes", "no"
    Set OnPCEquip to 0 ;display the Message Box only once
    Set status to 1
Endif
If ( status == 1); wait for player decision
    Set button to GetButtonPressed
    If ( button == -1 ); no button selected yet: do nothing
        return
    ElseIf ( button == 0 ); continue normally
        Set status to 0 ; reset for next time
    ElseIf ( button == 1 )
        Player -> drop, "power_sword" ; makes the player drop the item
        Set status to 0
    Endif
Endif

End

```

Displaying variables and text defines in a message box

To display variables in a message box you need to use a syntax describing the format of the number to be displayed. ATTENTION – there is a lot of wrong info in the original helpfile on this!

```

MessageBox "You have %.0f days left", days_left

```

The % symbol indicates the variable. The number after the dot determines the number of digits displayed. "f" signifies a float variable. The helpfile lists several types (f for float, D for short or long and S for string variables), of these I could only get f to work. However %g and %G work fine for short and long variables (thanks Niyt Owl). You can use things like %.3g, but the digit designation will simply be ignored. The designators are not really specific to the variable type, %.3f will also display a short or long variable.

String variables are mentioned in the helpfile but are to my knowledge not implemented, you can however use dialogue text defines in message boxes but do NOT use %: for text defines – In scripts its ^instead (thanks Ragnar_GD):

Text defines:

^PCName The player's name.
^PCClass The player's class.
^PCRace The player's race.
^PCRank The player's rank in the speaker's faction.
^NextPCRank The player's next rank in the speaker's faction.

^Cell The cell the player is currently in.
^Global Any global variable value. Floats display as 1.1, such as ^Gamehour. Note: you can also display a Global variable normally, using the above syntax such as %.1f, which would yield the same result. If you use the ^Global text define in a book, Morrowind will usually crash if you access or change the global variable while the book is open. This should be avoided at all costs! (Forum Info/Chris_K)

^Name The speaker's name.
^Race The speaker's race.
^Class The speaker's class.
^Faction The speaker's faction. If they have no faction, it will be blank.
^Rank The speaker's rank.

Note: These last listed ones will not work quite as they do in dialogue, as the defines default to the PC's values by default, not to the calling actor. So ^Name and ^PCName will both display the PC's name.

Example Script: Stupid sample script demonstrating all possible syntax:

```
Begin test1

short var_1
long var_2
float var_3
; GameHour is a global float variable

set var_1 to 1
set var_2 to 2
set var_3 to 3

MessageBox "^PCName, you have %g head, %G hands, and %.5f gold. One could say the hour is
getting late in ^cell. It's the ^GameHour hour or more exactly the %.2f hour!", var_1, var_2,
var_3, GameHour

End
```

Adding a dialogue topic

```
AddTopic, "Topic"
```

Once you have set up a dialogue topic in the TESCS, you may find that you still can't talk about it with the NPC you have given the dialogue to, because for the game you don't know that particular topic yet. There are two ways to change that condition: either you introduce the topic in another conversation topic (e.g. a custom greeting) or you give it to the player via script, which makes sense when it's an obvious topic the player would ask about without being brought to it by conversation (e.g. if you see and NPC standing under a waterfall, you

might want to ask him about "aren't you getting wet?" even if the NPC doesn't bring up the topic.

To do that, just attach a small script to the NPC:

```
Begin AddSpecialDialogue

;add possibility to ask about travel
AddTopic, "aren't you getting wet?"

End AddSpecialDialogue
```

You must already have the topic with this topic ID set up before you make this script, otherwise the script compiler will complain.

You can not remove a topic via script, you can however set a condition that refers to a local variable of the script as a speaker condition in the Dialogue editor, which can be used to achieve the same effect.

Initiating and ending dialogue

ForceGreeting

ForceGreeting can be used to make actors initiate dialogue. When *ForceGreeting* is called the dialogue window will open, and the actor will use a greeting according to his dialogue settings. Therefore, if you want a special greeting by the actor, you have to provide it via the dialogue window in the TES CS. It does not matter where the NPC is, this function will always work, so its usually best used in connection with a *GetDistance* or *GetPCCell* condition

Example Script: this script shows a nice set of condition being checked before initiating the *ForceGreeting* command

```
Begin balynScript

float timer
short doOnce

if ( GetJournalIndex "DA_Mephala" < 40 )
    Return
endif
if ( GetJournalIndex DA_Mephala >= 60 )
    Return
endif
Set timer to ( timer + GetSecondsPassed )
if ( timer < 5 )
    Return
endif

Set timer to 0

if ( doOnce == 0 )
    if ( GetDistance Player <= 1024 )
        if ( player->GetDistance "hlaalu_loaddoor_ 02_balyn" <=256 )
            if ( GetLOS Player == 1 )
                ForceGreeting
                Journal DA_Mephala 55
                set doOnce to -1
            endif
        endif
    endif
endif

End
```


Goodbye

Goodbye forces the end of dialogue. After calling this function (usually this function is used in the result section of a dialogue topic, not in scripts) the PC can only choose the goodbye option and thus closes the dialogue window.

Multiple choice – asking questions

```
Choice, "choice 1", choice1_enum ["choice 2", choice2_enum, ...]  
Choice "yes", 1, "No, certainly not!", 2
```

This is used in dialogue result fields to ask a decision of the player or can be called just to "continue" a longer speech. After the PC makes his choice, the same topic will be checked again, and you can provide the correct response by using function / choice / = / choice_enum in the speaker conditions of the dialogue window. Not meant for scripts as far as I know. Some more info from Riiak: *Unlike MessageBox this one can be stacked. By this I mean you could have 2 choice calls in the same result box for a max of 10 choices or 3 Choice calls for a max of 15 choices etc... I don't know the stack limit for this, but I'm pretty sure there is one. (I think it's limited to 5 choices per call).*

Adding to the journal and testing journal entries

```
Journal, "Journal_ID", Index_enum
```

```
Journal, MG_BCSroomsCombat, 10
```

This adds a journal entry to your in-game journal, the journal entry must have previously been set up in the dialogue editor. Index references which part of a journal topic is added. Beware of using simple names for journal topics, adhere to Bethesda's two letter standard (see above example) – otherwise the journal entry might show up as a regular conversation response, just like any other, if the topic title shows up in a conversation!

```
SetJournalIndex "Journal_ID" index_enum
```

I have not used the function (nor have Bethesda) and I am not sure where it is different from *Journal* function. According to the helpfile: "Sets the Journal to that index. Can move up or down". This means probably that you can set the journal index to a certain value without triggering a journal entry – so could this maybe be used to make quest be given over and over again, like in Daggerfall, by resetting the journal to 0?

```
ClearInfoActor
```

This is a function that is used in the results section of a dialogue topic – using this function will stop the topic from appearing in the PC's journal (under "Topics"). Useful to avoid cluttering the topic section with useless information.

```
GetJournalIndex, "JournalID" (returns short)  
  
If (GetJournalIndex, MG_BCSroomsCombat == 10)
```

This function returns the index of the highest (or the last?) journal entry for that journal topic that the player has received. This is very convenient for keeping track of quest advancement, and to have a script react according to what parts of the quest have already been performed.

Sample Script: Here is a short script demonstrating the use of both functions from the game:

```

Begin attack_slave

short nolore

If ( GetDeadCount "Vorar Helas" > 0 )
    return
endif

if ( GetJournalIndex "MV_SlaveMule" < 102 ); if PC has not yet reached a certain point
    If ( GetDistance, "Rabinna" < 512 )
        Rabinna->AiWander 0 0 0 0 0 0
        StartCombat, "Rabinna"
        Journal "MV_SlaveMule", 102 ; add journal entry
    endif
endif

End

```

Special dialogue-only functions

Among the functions available for defining dialogue conditions in the dialogue editor there are a few that do not have a direct equivalent in scripting functions. By using dialogue (e.g. using ForceGreeting, voice-dialogue or strategically placed quest NPC's) and the dialogue result field, you can still make use of these functions for scripting purposes, e.g. by setting a global variable in the result field. Examples of such functions are:

PC Sex (dialogue)

This is 0 if the player is male and 1 if the player is female.

Talked to PC (dialogue)

This is 1 if the speaker has ever talked to the player and 0 otherwise. You can use this to have someone say something the first time you speak with them – or for our scripting purposes to mark this person as "known" by the player.

It seems that Talked to PC will reset to 0 if the player has spent 72 game hours out of the NPC's cell. This time limit also applies to other NPC-related functions such as "forceGreeting": the NPC is available for a forceGreeting reference from a different cell only for 72 game hours. As a workaround, if you use PositionCell on the NPC once per day (even without changing their location), the 72 hour time limit no longer applies (Foum info /Time limit info from Cortex, thanks to Srikandi for bringing it to my attention). This trick to get around actors breaking their connection to you after 72 hours seems to require the cell you send them to to not be the cell where you initially met them (Forum info / Cortex).

This either implies it must not be their editor starting cell or that it must be a cell that you have not visited. In my fix I have an interior I send them to for this purpose so either explanation could be why it works.

So basically after you have met them they get sent there each day even though they are already there after the first sending.

Formatted: English (U.K.)

Formatted: English (U.K.)

Formatted: English (U.K.)

Rank Requirement (dialogue)

This checks to see if you "qualify" for the next rank in the speaker's faction.

This returns 0 if you do not have enough Faction Reputation and do not meet the skill requirements.

This returns 1 if you meet the skill requirements, but do not have the Faction Reputation.

This returns 2 if you have the Faction Reputation, but do not meet the skill requirements.

This returns 3 if you qualify.

Formatted: English (U.K.)

Formatted: English (U.K.)

Formatted: English (U.K.)

PC Clothing Modifier (dialogue)

This is the total value of all the clothing and armor the player is wearing. The value of your equipment changes the disposition of people in the game.

Friend Hit (dialogue)

Used in dialogue for when you attack a member of your group (like a follower)

The return values are:

0 = never been hit

1 = hit by pc 1st time

2 = hit by pc 2nd time

3 = hit by pc 3rd time

4 = hit by pc 4th time and the npc/creature is no in combat with the pc

Faction, Race and Rank

Determining the PC's status:

GetPCRank, FactionID_enum (returns short)

Returns PC's rank in faction. This will default to the actor's faction if FactionID is not defined. Returns 0-9 and -1 if not a member.

Sample Script: An actor/object with the following script is only enabled if the PC is not a member of House Redoran:

```
Begin bandenIndarysScript

if ( CellChanged == 0 )
    Return
endif

if ( GetPCRank "Redoran" == -1 )
    Enable
else
    Disable
endif

End
```

GetPCFacRep, [FactionID] (returns short?)

Probably returns reputation with the faction. Not tested.

SameFaction (returns Boolean/short)

Returns 1 if PC is in the faction of the calling object (NPC).

PCExpelled ["factionID"] (returns Boolean/short)

Returns 1 if PC has been expelled once from calling object (NPC) Faction, or a faction can be defined to get a specific one. For an example script look below, PCClearExpelled function.

GetRace, "RaceID" (returns Boolean/short)
Player->GetRace "Dark Elf"

Returns 1 if the object is of RaceID.

Sample Script: This is a global script Bethesda uses to set a variable that can be used to determine the PCs race in dialogue:

```
begin RaceCheck
```

```

;global script that gets run once to check the PC's race, so it can be used in dialogue

if ( Player->GetRace "Argonian" == 1 )
    set PCRace to 1
elseif ( Player->GetRace "Breton" == 1 )
    set PCRace to 2
elseif ( Player->GetRace "Dark Elf" == 1 )
    set PCRace to 3
elseif ( Player->GetRace "High Elf" == 1 )
    set PCRace to 4
elseif ( Player->GetRace "Imperial" == 1 )
    set PCRace to 5
elseif ( Player->GetRace "Khajiit" == 1 )
    set PCRace to 6
elseif ( Player->GetRace "Nord" == 1 )
    set PCRace to 7
elseif ( Player->GetRace "Orc" == 1 )
    set PCRace to 8
elseif ( Player->GetRace "Redguard" == 1 )
    set PCRace to 9
elseif ( Player->GetRace "Wood Elf" == 1 )
    set PCRace to 10
endif

StopScript RaceCheck

end

```

Modifying faction standing and reaction

PCJoinFaction ["FactionID"]

Makes the PC a member of the specified faction. FactionID is optional if it is not added it will use the faction of the NPC who called the function

LowerRank
RaiseRank

Raises or lowers the object's rank in its current faction. In the original game, this was only used in dialogue.

PCLowerRank
PCRaiseRank

Raises or lowers the PC 1 rank in the NPC's faction. If PC is not part of the faction, it will set the rank to 1

```
PCExpell ["FactionID"]
```

Expels PC from NPCs faction.

```
PCClearExpelled ["FactionID"]
```

Clears currently expelled flag.

Example script:

A script by Bethesda, which clears the Players expelled status after some time:

```
Begin expelledMG
;this is just a model
;it is supposed to be on an item in each of the Mages Guilds.

short myDay
short temp

if ( PCExpelled "Mages Guild" == 0 )
    return
endif

if ( ExpMagesGuild == 0 )
    Set ExpMagesGuild to 1
endif

if ( myDay == 0 )
    Set myDay to Day
endif

if ( myDay == Day )
    return
endif

if ( Day > myDay )
    Set temp to ( Day - myDay )
else
    Set temp to ( myDay - Day )
endif

Set myDay to Day

Set temp to ( temp + 2 )

Set ExpMagesGuild to ( ExpMagesGuild + temp )

if ( ExpMagesGuild > 30 )
    Set ExpMagesGuild to 0
    PCClearExpelled "Mages Guild"
    return
endif

End
```

```
ModPCFacRep, var_enum, ["FactionID"]
SetPCFacRep, var_enum, ["FactionID"]

ModPCFacRep, 5, "Imperial Legion"
ModPCFacRep, 5, "Temple"
```

Modifies or defines the reaction modifier for members of the specified faction (towards the PC).

```
ModFactionReaction, "factionID1", "factionID2", var_enum
SetFactionReaction, "factionID1", "factionID2", var_enum
```

Modifies or defines the reaction of one faction towards members of another faction.

Example: This is part of the MoonAndStar script. This part first makes the PC part of the faction "Nerevarine" and then sets two factions to react particularly to this change:

```
;faction reaction and journal stuff
Journal "A2_6_Incarnate" 50
player->modReputation 5
PCJoinFaction, Nerevarine

if ( GetPCRank, Redoran >= 0 )
    modFactionReaction, "Redoran", "Nerevarine", 4
endif

if ( GetPCRank, Temple >= 0 )
    modFactionReaction, "Temple", "Nerevarine", 4
endif
```

Combat

Initiating and ending combat

```
StartCombat, "ActorID"
StopCombat
```

```
"Actor_ID1" -> StartCombat, "ActorID2"
"Actor_ID1" -> StopCombat
```

StartCombat and StopCombat are used to set an actor into combat mode or back into normal mode. Start combat will make the calling actor attack the actor supplied as the argument. While StopCombat seems to be "safe" to use "every frame", you should supply a do once condition of some sort when issuing the StartCombat command, otherwise the actor might not do anything. Nevertheless continuous StopCombat is very dangerous to use, because it makes the NPC completely helpless: it will not retaliate when attacked (which however allows you to create a real pacifist...).

Once in combat mode the AI settings of the actor apply normally, e.g. if the actor has a high flee setting, he will flee despite the StartCombat command. For this reason, you will often see that the Fight rating is also changed when initiating combat in many scripts:

```
If (GetDeadCount, "My Friend" > 0 )
    StartCombat, Player
    SetFight, 100
endif
```

Detecting Attack

```
OnPCHitMe      (is local short variable)
```

```
Short OnPCHitMe
If (OnPCHitme == 1)
```

A local game variable (not a function, you must declare it as a variable as shown above) that gets set to 1 when the player hits the calling actor. Must be manually reset. It seems the use of the variable "short-circuits" normal NPC behavior in that an NPC with a script that uses this variable will not attack on its own accord. If you don't want the actor to remain passive you have to manually *StartCombat* (see example below). Once the actor is in combat mode, *OnPCHitMe* does not report any further hits by the PC. Except, according to information on the forum, *OnPCHitMe* gets reset (to 0) if another Actor hits the calling Actor after the PC did, then the variable gets reset to 1 if the player hits again.

ExampleScript: An example from my traveling merchants mod, to make a guar handler defend his charge while not in AIFollow mode, I attached this to the guar:

```

Begin _HB_Adros_GuarDefend

float timer
short attackstate
short OnPCHitMe

if ( OnPCHitMe == 1 )
    set attackstate to 1
    set OnPCHitMe to 0
endif

if (attackstate == 1)
    StartCombat, player
    set timer to ( timer + GetSecondsPassed)
    if ( timer >= 1 )
        set timer to 0
        if ( GetLOS, HB_adros_darani == 1)
            HB_adros_darani -> StartCombat, Player
            set attackstate to 0
        endif
    endif
endif

End

```

GetAttacked (returns Boolean/short)

Returns 0 if the actor has never been attacked and 1 if he has ever been attacked.

Example script: Uupse protects Yagrum Bagarn:

```

Begin uupse_Bagrum

short doOnce

if ( doOnce == 0 )
    if ( "yagrum bagarn"->GetAttacked == 1 )
        StartCombat player
        SetFight 90
        SetDisposition 0
        set doOnce to 1
    endif
endif

end uupse_Bagrum

```

GetTarget, "Actor ID" (returns Boolean/short)

If (GetTarget, Player == 1)

Returns 1 if object's combat target is ActorID (otherwise 0). This can be used to determine if an actor is in combat with the player (or another actor).

Example Script: this shows how an actor can be programmed to "not fight against the law". This is from the "frelene acquies script" – she is a Redoran follower and will not help the Player to fight Redoran guards.

```

[...

if ( GetCurrentAiPackage == 3 )
;if follow is the current package, set followNow and continue...

    set followNow to 1
    SetHello 0

    if ( GetTarget "ordinator wander_hp" == 1 )
        StopCombat
    endif

    if ( GetPCCell "Vivec, Hlaalu Prison Cells" == 0 )
        ;if follow is done, NPC has arrived...
        Journal HR_Stronghold 144
        set followNow to 0
    endif
endif

```

```

        AiWander 256 0 0 40 20 20 0 0 0 0 0
        ForceGreeting
        SetHello 30
    endif
[...]
```

```

HitOnMe, "Weapon ID"          (returns Boolean/short)
HitAttemptOnMe, "Weapon ID"   (returns Boolean/short)
```

These functions return true (1) for 1 frame if the calling actor is successfully hit or if it was attempted to hit it with a specified weapon. *HitOnMe* is used only in the *LorkhanHeart* script (only look at that if you have finished the game or don't mind severe spoilers). I guess it could be a nice function to script any kind of fight of the "you need this special weapon to kill this particular monster" type.

Keeping track of kills and knockouts

```

OnDeath          (returns Boolean/short)

    If ( OnDeath == 1 )
```

Returns 1 for 1 frame when the actor is killed.

```

OnMurder          (returns Boolean/short)

    If ( OnMurder == 1 )
```

Returns 1 for 1 frame when the actor is murdered. The conditions for *OnMurder* are not entirely clear to me – from the context of its use in the game however, it seems that *OnMurder* gets set when you are reported as a murderer to the law ("your crime has been reported"). So a murder only happens when you kill someone illegally AND are seen.

Sample Script: this sets a variable that is used in the "Redoran Hortator" dialogue topic to determine if the player has killed a councilor:

```

begin RedoranCouncilor

;no lore...
short noLore

;for HT_Monopoly
short mageMonopolyVote

;for Hortator dialogue...
if ( OnDeath == 1 )
    if ( OnMurder == 1 )
        Set RedoranMurdered to 2
    else
        Set RedoranMurdered to 1
    endif
endif

End
```

```

OnKnockout        (returns Boolean/short)
```

Returns 1 for 1 frame when the actor is knocked unconscious (e.g. in hand-to-hand combat)

```

GetDeadCount, "Actor ID"      (returns short?)

    If ( GetDeadCount "divayth fyr" > 0)
```

The function returns the number of references (individuals) of type "Actor ID" that have been killed. A useful function for quest scripting to keep track of which NPCs are still alive. Note that there is an equivalent function for dialogue as well. Other uses are imaginable, e.g.

building a reputation with certain monsters that might flee you instead of fighting after you killed more than 100 of them, etc.

Sample Script:

GetDeadCount is often used to check if a certain NPC is dead. It is advisable to use "> 0" in such cases, as you never know if another mod might add another instance of that ID, so it's better to play it safe.

```
Begin araraUvulasScript

short noLore

if ( CellChanged == 0 )
    return
endif

if ( GetDeadCount "Neloth" > 0 )
    Disable
endif

End
```

Crime

Jailing the PC

GotoJail

Sends the PC to the (closest available) prison, more exactly speaking to a PrisonMarker (Door object) and applies the usual prison penalties.

SampleScript:

Here is a cool little scripted item by B from the Modern Adventurer mod. The cursed Holiday Pants that send you to prison:

```
Begin Holiday_script

Short OnPCEquip
Short message

if ( OnPCEquip == 1 )
    if ( MenuMode==1 )
        return
    else
        Set message to Random 2
        if ( message==0 )
            MessageBox "The holiday pants contain a mighty enchantment of happiness.
Lots of happiness. When the guards found you doing the Can-Can on top of the nearest silt
strider port, they were not very amused.", "ok"
        elseif ( message==1 )
            MessageBox "The holiday pants make you scream and shout with joy as you
relive the happiest days of your childhood. The guard that brings you back to your senses is
in stark contrast to this experience", "ok"
        endif
        Player -> GoToJail
    endif
Set OnPCEquip to 0
endif

End holiday_script
```

Clearing the PC of crime

PayFine

The PayFine function removes the stolen items from the PCs inventory; it does not remove any gold. Call after paying a crime fee to clean AI. Also puts the PCs hands down (that is not ready to cast or fight).

PayFineThief

Like PayFine function but *does not* remove stolen items from the PCs inventory. Call to "clean AI". May have incorrectly removed stolen items before one of the patches.

PC crime level

GetPCCrimeLevel (returns short)

Reports the current crime level of the PC. Can be used to detect whether a crime the PC has committed has been seen. See the "Bill_MT_writxxxxx" scripts for examples of its use.

Magic

Limiting the use of teleport

DisableTeleporting
EnableTeleporting

Rather self explanatory, these functions turn the ability to use teleporting magic on or off. Nice to keep those magic user types from wimping out of your dungeon ☺. In the original game it's only used when the player encounters Dagoth Ur.

I won't show the whole script as it would be quite a spoiler, but here is the part that uses the function:

```
short teleportDisabled

if ( teleportDisabled == 0 )
    DisableTeleporting
    Set teleportDisabled to 1
endif
```

This is later reset in the EndGame script.

Note: when Tribunal is installed this function is effectively **broken**: One of the start-up scripts in Tribunal overrides all other teleport commands and forces teleporting on except within one specific area in Mournhold (thanks to Slink and Riiak for the info). I am sure this could be fixed easily enough by editing the script in question.

Here is the culprit:

```
Begin TribunalMain

;check for teleporting
if ( GetPCCell "Sotha Sil," == 1 )
    DisableTeleporting
else
    EnableTeleporting
endif

;check levitate
if ( GetPCCell "Sotha Sil," == 1 )
    DisableLevitation
elseif ( GetPCCell "Mournhold" == 1 )
    DisableLevitation
else
    EnableLevitation
endif

end
```

and here a completely untested suggestion for a fix:

```

Begin TribunalMain

Short switch1
Short switch2

;check for teleporting
if ( GetPCCell "Sotha Sil," == 1 )
    DisableTeleporting
    Set switch1 to 1
Else
    If ( switch1 == 1 )
        EnableTeleporting
        Set switch1 to 0
    endif
endif

;check levitate
if ( GetPCCell "Sotha Sil," == 1 )
    DisableLevitation
    Set switch2 to 1
elseif ( GetPCCell "Mournhold" == 1 )
    DisableLevitation
    Set switch2 to 1
else
    If ( switch2 == 1 )
        EnableLevitation
        Set switch2 to 0
    endif
endif

end

```

Checking and managing souls and soulgems

```
HasSoulgem, "CreatureID"
```

```
    If HasSoulGem, "golden saint"
```

This function checks if the player has a soul gem containing the specified soul in his inventory. A little used function that could allow some fun quests and new uses for soulgems.

Sample: This is part of the StrongSoulCheck script:

```

if ( Player->HasSoulGem "atronach_storm" > 1 )
    Set counter to ( counter + 2 )
elseif ( Player->HasSoulGem "atronach_storm" > 0 )
    Set counter to ( counter + 1 )
endif

```

```
RemoveSoulgem, "CreatureID"
```

Removes a soulgem with the specified soul from the players inventory.

Sample: this is the complementary part from RemoveStrongSoul script to the example above:

```

if ( counter > 0 )
    if ( Player->HasSoulGem "atronach_storm" > 0 )
        Player->RemoveSoulGem "atronach_storm" 1
        Set counter to ( counter - 1 )
    endif
endif

```

Note, the player will not be happy if they get Azura's Star taken away by this. Here's a sample solution:

```
short StarCount ;They could have more than one I guess.

if ( OnActivate )
    if ( Player->HasSoulGem "Golden Saint" > 0 )
        set StarCount to ( Player->GetItemCount "Misc_Soulgem_Azura" )
        Player->RemoveSoulGem "Golden Saint" 1
        if ( ( Player->GetItemCount "Misc_Soulgem_Azura" ) < StarCount )
            Player->AddItem "Misc_Soulgem_Azura" 1
        endif
        Player->AddItem Gold_001, 10000
        MessageBox "Thank You, Come Again."
    else
        MessageBox "You have no Golden Saint souls."
    endif
endif
```

```
AddSoulGem "creature ID", "soulgem ID"
```

```
AddSoulGem "atronach_storm", Misc_Soulgem_Grand
```

AddSoulGem adds a soulgem of the specified type and with the specified soul to the players inventory.

```
DropSoulgem, "Creature ID"
DropSoulGem "atronach_storm"
```

I didn't test this yet – I assume it makes the calling object drop a filled soulgem with the given soul.

The soul gems in the game have the following ID's:

Soul Gem ID's:

Misc_SoulGem_Azura

Misc_SoulGem_Grand

Misc_SoulGem_Greater

Misc_SoulGem_Common

Misc_SoulGem_Lesser

Misc_SoulGem_Petty

Formatted: English (U.K.)

Formatted: French (France)

This function was not used in the original game.

Casting spells

```
AddSpell, "SpellID"
RemoveSpell, "SpellID"
Cast, SpellID, "TargetID"
```

The AddSpell function will add the spell to the calling object. This can mean two things: normal spells are added to the players spell list. Curses, diseases etc, however will affect the calling object. The same is true for the RemoveSpell function: Normal spells are removed from the list, curses or diseases are removed as effects.

The Cast function makes the calling object cast the spell "SpellID" on the target "TargetID", and Target will suffer or benefit from the effects normally. **Note:** It was believed that Cast would only work on the PC. At least with Tribunal (not sure about earlier versions) you can use cast to cast a spell from an activator on an actor – probably other combinations would work too.

Sample Script: The Cast function can be used for traps, as in the following example attached to a Container. Note that there is a do once condition here, so that the effect is not cast continuously on the player.

```
Begin Trap_script
short done
if ( OnActivate == 1 )
    if ( done == 1 ) ;do-once condition
        Activate
        return
    else
        Cast, "flame", Player ;damage to player
        set done to 1
        Activate
    endif
endif
End trap_script
```

The next example script uses the *AddSpell* function:

```
begin Item_Cast

short OnPCEquip
short CurseAdded
float Timer

if ( CurseAdded )
set Timer to ( Timer + GetSecondsPassed )
    if ( Timer >= 25 ) ; after 25 s remove the spell. 25 damage done.
        set Timer to 0
        Player->RemoveSpell "ItemFlame"
        set CurseAdded to 0
    endif
endif

if ( OnPCEquip );when item is equipped
    if ( MenuMode )
        return
    elseif ( CurseAdded == 0 )
        Player->AddSpell "ItemFlame" ; a spell of type curse!
        ;custom spell one point of fire damage / s
        set Timer to 0
        set CurseAdded to 1
    elseif ( CurseAdded ) ; Add the spell only once
        Player->RemoveSpell "ItemFlame"
        set CurseAdded to 0
    endif
endif

end Item_Cast

(script by Patrin, edited)
```

The added spell is a custom-made curse spell doing one point per second flame damage. Note that there is again a do once condition implicit in this script. **Failure to have a do once condition can crash the game! Also, it appears that creatures killed with curse spell effects on them cause all other creatures of that type to have the same curse on them. This can be avoided by using 'RemoveSpell' in an 'OnDeath' section of the script. (Forum Info / Argent)**

Managing and testing for spells

```
GetSpell, "Spell_ID" (returns Boolean/short)
If ( Player -> GetSpell, "heal companion" == 1 )
```

Returns true if object has Spell_ID in inventory. However, this does not seem to work on spells of types. "Powers" and other race/birthsign related spells don't seem to register with this function, only the ones listed in the main part of the spellbook window. Sample script see below.

```
GetSpellEffects, "Spell_ID" (returns Boolean/short)
```

Returns true if Spell_ID is affecting calling object. The following could be added to the "trap_script" discussed under "casting spells" above:

```
if ( Player -> GetSpellEffects, "flame" == 1 )
    MessageBox "You have been flamed"
endif
```

This is the favorite possibility of adding new "spell effects". A dummy spell is created that does some minimal effect, e.g. raising luck by 1 point for 1 second. The GetSpellEffects function is used to detect if that spell has been cast on the player, and the script handles

everything else. Sample script see below. Seems to work for Abilities and Diseases as well. Probably Curses and Blight Diseases for that matter but I didn't check.

Managing and testing spell effects

```
GetEffect, Effect_ID (returns short)

If ( GetEffect, sEffectRestoreHealth == 1)
```

This function returns TRUE if the calling Actor is being affected by the effect. Important: Effects are not spells, but the elements spells are made of. In the Appendix you can find a list of all spell effects

```
RemoveEffects, Effect_ID#_enum
Player -> RemoveEffects, 75
```

Removes all spells on the actor that include the Effect. For this function you need the **number** of the effect-ID unlike the GetEffect function where you need the effect ID itself (Bravo, Bethesda!). Both can be found in the appendix.

Important: Effects are not spells, but the elements spells are made of. In the Appendix you can find a list of all spell effects.

Sample script: This is a demonstration script that lets you check if a spell is in inventory, if it's active on the player, if the effect it causes is on the player and then removes the effect. Start it in the console using "StartScript Magicscript" to try it out.

```
Begin Magictest

short var_1
short var_2
short var_3

if ( Player -> GetSpell, "hearth heal" )
    set var_1 to 1
else
    set var_1 to 0
endif

if ( Player -> GetSpellEffects, "hearth heal" )
    set var_2 to 1
else
    set var_2 to 0
endif

if ( Player -> GetEffect, sEffectRestoreHealth )
    Player -> RemoveEffects, 75 ;delete this line to see what happens normally
    set var_3 to 1
else
    set var_3 to 0
endif

MessageBox "GetSpell: %.0f   GetSpellEffects, %.0f   GetEffect: %.0f ", var_1, var_2, var_3

End
```

Testing disease

```
GetBlightDisease (returns Boolean/short)
GetCommonDisease (returns Boolean/short)
```

Both functions return 1 if the calling actor has the appropriate type of disease, otherwise 0. These are used in the disease scripts that give diseased or blighted creatures their disease:

Sample Script:

```
Begin diseaseBlackHeart

DontSaveObject

if ( CellChanged == 0 )
    return
endif

if ( GetBlightDisease == 0 )
    AddSpell "black-heart blight"
endif

End
```

Sound

Letting Actors talk

Say, "file name", "text"

Make subject "say" the sound file, only works on animating objects. The .mp3 voice sound files can be found in "Data files\Sound\Vo\" folder and are ordered in subfolders by race and gender. You can browse through most of them in the Dialogue/voice window as well. Text is what is displayed as a subtitle as the file is played.

SayDone

Returns true if the object is not saying anything.

Playing music

StreamMusic, "filename.ext"

Plays the sound file "filename.ext", usually an mp3 file, as the current music file.

Playing sounds

PlaySound, "sound ID"

PlaySoundVP, "sound ID", volume_enum, pitch_enum

PlaySound3D, "sound ID"

PlaySound3DVP, "sound ID", volume_enum, pitch_enum

"ex_gg_portcullis_02"->Playsound3DVP "Dwemer Door Open" 1.0 1.0

The PlaySound function plays a sound without any modification. It does not matter to which object the script using the function is attached, the sound will always play at full volume, directly in the player's ear, so to speak.

The PlaySound3D function plays a directional sound source. The sound will seem to be emitted by the object to which the script with this function is attached.

The "VP" variants of each of these commands allow setting volume and pitch for the sound that is played. Bethesda has not made much use of this, it seems and it's only ever used with

1.0 set for both, which appears to be the standard anyway. My own experiments showed the sound not playing when I set volume to a variable, but this is no final verdict. The "sound ID" is the ID listed in the sound window accessed via Gameplay – sounds menu. You can add sounds there (place the .wav file somewhere in Data files/sounds). A good source for sounds on the web is <http://www.findsounds.com/>. Sounds should be in a certain format (see below) so you might have to change the format in a suitable program if you don't hear a sound in game.

Controlling sound

StopSound, "Sound ID"

stops the sound "SoundID" if it is currently playing in the calling object.

GetSoundPlaying, "sound ID" (returns Boolean/short)

Returns 1 when the specified sound is currently playing on the calling object. The sound ID's can be found in the Gameplay menu /sounds and /sound gen, where you can also set up your own (see below for formats). This function can be used to control sounds, but also to gain information, because certain sounds are tied to certain events in game, e.g. the "Critical Damage" sound or the "Disarm trap" sound.

▲ **Sample Script:** This simple script assures that lava always has its rumble sound playing:

```
begin lava
if ( menumode == 1 )
    return
endif

if ( CellChanged == 0 )
    if ( GetSoundPlaying "lava layer" == 0 )
        PlayLoopSound3DVP "lava layer", 1.0, 1.0
    endif
endif

HurtStandingActor, 20.0          ;20 pts of damage a sec

end lava
```

Formatted: English (U.K.)

Sound file formats:

Note: Not all sound files seem to play correctly in the game (although all play in the editor). To be safe, use the formats used by Bethesda (thanks to random name) :

"Cr" and the "Fx" folder
Windows PCM (.wav)
22050 Hz, 16-bit, Mono

"Vo" folder format
MPEG Layer-3, 64 Kbps
44100 Hz, 16-bit, Mono

Keeping track of time

There are a number of possibilities to follow the passage of time in scripts, including some that are not, or only poorly documented in the help file.

Timer

GetSecondsPassed (returns float)

A simple timer can be scripted with the GetSecondsPassed function. It returns the seconds passed *since the last frame* as a float value. To use this for a timer use something like the following example:

```
Begin TimerScript

Float timer
Short state

Set timer to (timer + GetSecondsPassed)

If (timer > 10)
    MessageBox "Displayed every 10 seconds"
    Set timer to 0
Endif

End TimerScript
```

Morrowind's time related global variables

GameHour (float global variable)
Day (short global variable)
Month (short global variable)
Year (short global variable)

These globals get set by the game and contain the current date and time.

The MW calendar is a little bugged (thanks to samois for the info): MW starts on Day 16, Month 7, Year 427. (16 Last seed)

The months are as follows, with the days in each month.

(Morning Star ???)

Suns Dawn 31

First Seed 28

Rain's Hand 31

Second Seed 30

Mid Year 31

Sun's Height 30

Last Seed 31

Heart Fire 31

Frost Fall 30

Suns Dusk 31

Evening Star 30

So there are 334 days in a MW year!?! Basically it seems that Bethesda screwed up their code and lost a month, Morning Star / January... It seems the mistake was simply making it "wrap" to the wrong month from Evening Star. If you manually set month to 0 it will correctly display Morning Star in the rest menu. So this could possibly be scripted around.

Sample Script: Checking the time of day with the GameHour function:

```
Begin AfternoonTea

If ( GameHour >= 17 )
    If ( GameHour <= 19 )
        "Cup of Tea" -> Enable
    Endif
Endif

End AfternoonTea
```

Formatted: English (U.K.)

```

endif
elseif ( GameHour < 17 )
    if ( GameHour >19 )
        "Cup of Tea" -> Disable
    endif
endif

End AfternoonTea

```

Keeping track of days passed

Day (short global variable)

Use the global "Day" variable. It contains the current "day of the month" – so on "17, last seed" it is 17. This can be used to keep track of the number of days passed:

```

Short localdaysPassed
Short currentDay

if ( currentDay != Day ) ;whenever Day changes (presumably increasing)...
    set currentDay to Day
    set localdaysPassed to localdaysPassed + 1 ;add one to the counter
endif

```

This would usually be used for time-limited quests in a global script, to make sure the passage of time is correctly measured. Innovative scripting might also make use of it to trigger events after an item has been in the player's possession for some time, etc.

Moon phases

Indispensable for any potential werewolf mod.

```

GetMasserPhase (returns short)
GetSecundaPhase (returns short)

If (GetMasserPhase == 4)
    [enable werewolf monster]
endif

```

Note: The helpfile lists *GetSecundusPhase*, the above syntax *GetSecundaPhase* is the correct one. Also note that GetMasserPhase and GetSecundaPhase return the value of the moon phases for the last exterior cell you visited (Forum Info / Elim).

I only made a quick test with these, but they seem to work.

Both functions return short with these values:

0 = MOON_PHASE_NEW (this is the default)

1 = MOON_PHASE_WAXING_CRESCENT or MOON_PHASE_WANING_CRESCENT:

2 = MOON_PHASE_WAXING_HALF or MOON_PHASE_WANING_HALF:

3 = MOON_PHASE_WAXING_GIBBOUS or MOON_PHASE_WANING_GIBBOUS:

4 = MOON_PHASE_FULL

Working with objects

Working with inventories

Adding and removing items from the inventory

```

AddItem, "ObjectID", count_enum
RemoveItem, "ObjectID", count_enum

Actor -> AddItem, "item_ID", 1

```

These functions are simple enough, adding or removing items from the player's inventory, A *RemoveItem* call makes the object "vanish". Apparently these functions can accept global variables, but only in dialogue results, and only if you don't set the global variable in the same dialog result (Forum info / Argent; According to Argent, the maximum amount he has been able to add using AddItem, var was 65534 (using a long var=2147483520)). The *Drop* function will remove it from the inventory and drop it into the world, at the calling actors feet. Removing items that are not present in the inventory does not crash the game, but the 'RemoveItem' function will subtract the removed item's weight from the character's encumbrance, EVEN IF the item is NOT in the character's inventory. So if a script uses 'RemoveItem' to remove a 4 lb. item that the character doesn't have, the character's encumbrance will wind up 4 lbs. lower than it should be. The workaround for this bug is to always check for the presence of an item before using 'RemoveItem' to delete it (Thanks DinkumThinkum for this info). Also beware of removing an item that is executing a script, that will crash the game.

Example script:

The following script was discussed on the forums (sorry don't remember who first made it). It was supposed to ask the player if he wanted to recycle an item when it was equipped, and then replace that item with the "recycled" version.

```
Begin scr_thing

short button
short OnPcEquip
short state

if ( MenuMode == 1 )
    return
endif

if ( OnPcEquip == 1 )      ; when the item is equipped
    set state to 1
    set OnPcEquip to 0      ; do this once per equip event...
endif
if ( state == 1 )
    MessageBox "recycle?" , "yes", "no"
    set state to 2
elseif ( state == 2 )
    set button to GetButtonPressed
    if ( button == 0 )
        PlaySound "mysticism cast"
        player->RemoveItem "item_a", 1 ;this line crashes the game!!!
        player->AddItem "item_b", 1
        set state to 0
    elseif ( button == 1 )
        set state to 0      ; once done, reset everything
    endif
endif
endif

end
```

It worked nice enough without the *RemoveItem* function in it, but with that line it would crash. The reason was that this script was *attached* to item_a, and thus the running script would be removed with it, which apparently crashes the game. So the above idea had to be handled via a global script.

Dropping items

```
Drop, "ObjectID", count_enum

"Actor_ID" -> Drop, "ITEM_ID", 1
```

This function is supposed to drop the item from the inventory to the calling actor's feet. This seems to work correctly only for the player character, dropping the item to his feet. When I

used it for NPC's, the items would get removed correctly from the NPC, but still dropped at my own player character's feet.

Note: An interesting note on this is that if they really have that item, they will drop it, with charges and condition intact. If they don't really have that item, a new instance will be created.

Checking for presence of items in inventory

```
GetItemCount, "ObjectID"      (returns short)

Short objectcount
Set objectcount to ( "Mob_ID" -> GetItemCount, "Object_ID" )

If ( GetItemCount, "Object_ID" >= 1 )
```

This function checks the inventory of the calling object and returns the number of objects of type "Object_ID" it owns.

Monitoring inventory activities

Certain inventory activities can be checked through scripts using local variables. These local variables are automatically set when the Player does the action. They are NOT functions, but variables that are true or false.

```
OnPCEquip      (is local short variable)
Short OnPCEquip
If ( OnPCEquip == 1 )
```

The PC has the object equipped (remains true while object is equipped)

This game variable (needs to be declared) gets set to 1 if the player is equipping the calling object. It will remain "true" while the item is still equipped, but gets reset to 0 if the item is unequipped. So, in some cases you might want to manually reset it:

```
if ( OnPCEquip == 1 )      ; when the item is equipped
[do something]
set OnPCEquip to 0      ; do this once per equip event...
endif
```

The next time the item is unequipped and equipped again, the functions in [do something] will be performed again. You can also use a status variable to control when an effect will be executed. Note that this can also be processed while in Menu mode:

```
If (MenuMode ==1)
    if ( OnPCEquip == 1 )      ; when the item is equipped
        [do something]
        set OnPCEquip to 0      ; do this once per equip event...
    endif
endif
```

This script will execute while you are in the menu, as soon as the item is equipped, while the following will be executed only after you leave the menu:

```
If (MenuMode ==1)
    Return
Endif

if ( OnPCEquip == 1 )      ; when the item is equipped
[do something]
set OnPCEquip to 0      ; do this once per equip event...
endif
```

An additional **Sample Script** can be found below with the Equip function.

▲ OnPCAdd (is local short variable)

The PC added the object to inventory

Formatted: English (U.K.)

OnPCDrop (is local short variable)
The PC dropped the object

OnPCSoulGemUse (is short variable)
the Object is a soulgem and it has been used in either recharging or item making

UsedOnMe

UsedOnMe, "Object ID" (returns Boolean/short)

According to helpfile:

```
if ( UsedOnMe, Misc_pot_redware_01 )
```

"Returns true if the "Object ID" has been used on the calling object. This is used for scripts that make objects do certain things of the player uses an object on it."

According to current knowledge this function is **broken**.

Repairing objects

OnPCRepair (is short variable)

A game variable that gets set to 1 when the PC repairs the calling object.

RepairedOnMe, "Object ID" (returns Boolean/short)

```
if ( "daedric_mace"->RepairedOnMe, "repair_journeyman_01" == 1 )
```

This function returns 1 if the calling item is repaired by an item of type "Object ID". Object ID has to be of type "Repair Item" and the calling object must be either weapon or Armor.

The similar function OnRepair is apparently **broken**. It should get set to 1 when any repair is attempted at the object: "returns true if calling object is repaired at all".

Equipping an Item – equip and PCSkipEquip

Equip, "Object_ID"

```
"Actor_ID" -> Equip, "p_restore_health_q"
```

(Also see OnPCEquip function above)

Partly Broken. This could have been an immensely useful function. Unfortunately, most of the potential uses do not work: You can NOT autoequip anything on the Player. You can NOT force the Actors to equip weapons or armor with this (this is completely governed by their skills with armor or weapons). You can NOT make "non-removable" items, like cursed armor, etc. You CAN make Actors swallow potions, or so I heard.

Note: This Function was apparently **fixed** in Tribunal. You can now force Actors to equip armor, weapons and clothing. So you now CAN do all of the above ☺. Praise to Bethesda.

Sample Script: This script (Tribunal required) curses an item (a Chitin Club) so that it can't be unequipped anymore. Not even using quick-keys. Bugger! Now you have to fight with a chitin club to the end of your days, which will probably come soon ☺ The player can still use magic however.

```
Begin cursed_item  
  
short state  
short OnPCEquip
```

```

if ( OnPCEquip == 0 ) ; item is not equipped
    if ( state == 0 ); if club has never been equipped, don't do anything yet.
        return
    else
        Player -> Equip, "cursed_club" ; reequip the item!
        MessageBox "The item is cursed, it doesn't leave your hand" ;taunt the player
    endif
else
    if ( state == 0 ) ;first time equipped. The trap snaps shut
        set state to 1
    endif
endif
End

```

PCSkipEquip (is short variable)

Set this to 1 to skip equipping object. Good for popping up messages for breaking seals on books and such. For an extended example look at the SealedTreasuryReport script in the editor.

Sample Script: Here is a short script I made for a werewolf mod, it makes an item non-equipable under certain conditions:

```

Begin non_equipable

; keeps lycanthropic PC's from equipping werewolf hunter items for balancing reasons
; if the PC equips these before becoming a werewolf, he can wear them until he takes them off
; but then can't reequip them. So after the first transform he can't equip them again

short PCSkipEquip
short OnPCEquip

if ( PCWerewolf != 1); if player is not a ww, he can use the armor
    set PCSkipEquip to 0
    return
else
    set PCSkipEquip to 1
endif

if ( OnPCEquip == 1 )
    MessageBox "This item is enchanted with werewolf bane-spells. You can not wear it!"
    set OnPCEquip to 0
endif

End

```

Placing an item near the PC

PlaceAtPC, "Object_ID", number_enum, distance_enum, direction_enum

```

PlaceAtPC, "Secret Message", 0, 30, 1
PlaceAtPC, " ancestor_ghost", 1, 256, 1

```

▲ This function places a new reference of "Object_ID" near the player. The function lets you define a direction relative to the player where the object is going to appear and a distance (in units). If that location is not safe (in the air, in a wall, etc), the object will be placed at one of the other axis or at the player's exact location (feet). (Erratum: Thanks to Isildur and Esteban for pointing out that number_enum and distance_enum were switched around in previous versions)

direction is:

0 = front
1 = back
2 = left

Formatted: English (U.K.)

3 = right

Checking activation of an item and activating it

An item is usually activated when you press the spacebar to "use" it. Most items have a standard action that is performed when they are activated: Doors open, chests display their content, Actors initiate dialogue etc. The OnActivate function allows you to intercept the standard action and do something else or check a condition first:

OnActivate

```
If ( OnActivate == 1 )
```

This gets set to one for one frame when the object is activated. To perform the standard action of the object once OnActivate has been called, you have to use the activate command:

Activate

```
"Object_ID" -> activate
```

Notes: I have tried this function with doors and it did not work. It is tested and works for containers, actors, and activators.

Example script:

The following script demonstrates the use nicely. It is attached to a container object, a chest that does NOT advertise its trap like the normal in-game ones do:

```
Begin Trap_script
short done

if ( OnActivate == 1 )
.....if ( done == 1 ) ;do-once condition
.....Activate
.....return
.....else
.....Cast, "flame", Player ;damage to player
.....set done to 1
.....Activate ; Call standard action: the chest opens
.....endif
endif

End trap_script
```

Locking and Unlocking doors or chests

Lock, short_enum_locklevel

Unlock

GetLocked (returns Boolean/short)

(only Door and container objects)

```
My_Door -> Lock, 50

If ( GetLocked == 1 )
    Unlock
Endif
```

These functions are used to lock and unlock doors or containers. The GetLocked function returns 1 when the calling object is locked. Lock locks the object with the lock level specified (0-100). Lock 0 has an odd effect though. The door/container will be neither openable nor pickable. Unlock removes any lock, regardless of lock level.

Example script:

Here is a sample script by qwert, which makes a chest function as a security skill-training device by constantly relocking it:

```
Begin PC_Security_Skill_Trainer

float timer

if ( menumode == 1)
return
endif

set timer to timer + GetSecondsPassed
if ( timer > 10 )
set timer to 0
endif

if ( timer == 0 ) ;using a timer to relock after 10 seconds pass
"Storm_Chest_Trainer"->Lock 50
endif

End
```

Enabling and disabling objects

```
Enable
Disable
GetDisabled (returns Boolean/short)

"ObjectID" -> enable
```

The *Disable* function makes an object completely vanish from the game world, meaning it's neither rendered nor processed (attached scripts are still active, however). The *Enable* function makes a disabled object visible and processed again. *GetDisabled* (returns 1 if object is disabled) can be used to obtain the current status of an Object. These functions are very powerful and could e.g. be used to swap different models of statics (normal house replaced by house burnt to the ground, etc). It is e.g. used for the stronghold building process.

Example script:

An example from the game is the SlaveScript that makes freed slaves vanish once the player leaves the cell:

```
begin slaveScript

short slaveStatus
short doOnce
short NoLore

[other slaves status checks - check original script!]

if ( slaveStatus == 3 )
    if ( GetCurrentAIPackage == 3 )
        AIWander 512 0 0 0 0 0 0 0 0 0 0
    endif
    if ( GetItemCount Slave_Bracer_Left > 0 )
        Drop Slave_Bracer_Left 1
    endif
    if ( GetItemCount Slave_Bracer_Right > 0 )
        Drop Slave_Bracer_Right 1
    endif
    if ( CellChanged == 1 )
        Disable ;***** Make slaves vanish once freed and player leaves
    endif
endif

end slaveScript
```

Caution: disabling lights

There seems to be a game engine issue with disabled lights – actors and some other types of objects still seem to be illuminated, while the world around is not. I have not thoroughly tested if this can be avoided, but a suitable workaround is to physically remove the light to a remote location (e.g. a few meters below the floor) instead of disabling it. Another trick comes from Indigo: If you enable a light that is set to "negative" (which means its generating darkness instead of light) after you disable the normal light, the illumination problem goes away.

Don't save changes to an object

DontSaveObject

Call this function if changes to the object are not to be saved to the savegame. Not entirely sure about the use of this one, assumedly to avoid cluttering savegames with unnecessary information. I think it only stops additional state data from being saved with the object (e.g. current position and angle etc.). It does not delete the object entirely (See the Tribunal function "SetDelete" for this). In the original game, it's used in the SignRotate script and the following one:

Sample Script:

```
Begin diseaseAscended

DontSaveObject

;ascended sleeper has all the blight diseases for some reason...
if ( CellChanged == 0 )
    return
endif
AddSpell "ash woe blight"
AddSpell "black-heart blight"
AddSpell "chanthrax blight"
AddSpell "ash-chancre"

End
```

Moving and rotating objects

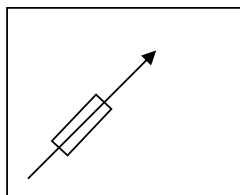
The following functions do not work on the player, NPCs or monsters. They do work on static objects, activators, containers, miscellaneous objects etc. **Note:** Actors, including the PC tend to fall through moving objects after a while. This can be avoided by quickly disabling and enabling a moving object every frame – doing this apparently updates the collision information. Actors can also be equipped with a levitate or slowfall ability to further decrease the chance of falling through objects (for more details see the tips and tricks section on rideable objects by MadMax)

Moving along an objects axis

```
Move    axis(x/y/z), units/sec_enum

Move    x, 100
```

Moves the object along the selected axis (x, y, or z) at the speed selected. This speed is in units per second (21.3 units per foot). This movement is based on the object's local rotation. Thus, a positive y movement will always move the object along its local forward vector:

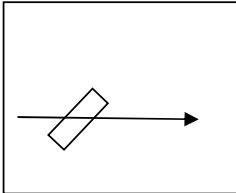


Moving along the world axis

MoveWorld axis(x/y/z), units/sec_enum

MoveWorld z, 100

Moves the object along the selected world axis (x, y, or z) at the speed selected. This speed is in units per second (21.3 units per foot). This movement is based on the world axis, thus a positive z movement will always move the object up, regardless of it's local rotation:



In world coordinates Z is always up / down , X is east / west and Y is north / south. This is an **example** after a script I once picked up on the forums that makes a platform slowly move out and back once the player stands on it:

```
Begin platform_script

Short PlatformMoving
Short ActivateMe
Float Timer

If ( GetStandingPC == 1 )
    Set ActivateMe to 1
Endif

If ( ActivateMe == 1 )
    If ( PlatformMoving == 0 )
        Set Timer to Timer + GetSecondsPassed
        If ( Timer <= 15 )
            "floating_platform_01"->MoveWorld X 10
        Else
            Set Timer to 0
            Set PlatformMoving to -1
        Endif
    Endif
    If ( PlatformMoving == -1 )
        Set Timer to Timer + GetSecondsPassed
        If ( Timer <= 15 )
            "floating_platform_01"->MoveWorld X -10
        Else
            Set Timer to 0
            Set PlatformMoving to 0
            Set ActivateMe to -1
        Endif
    Endif
Else
    "floating_platform_01"->SetAtStart
Endif

End platform_script
```

Rotating objects

Similar to the movements described above, you can also rotate objects, around either their local or the world axis and determine the current angle:

Rotate , axis, angle/sec_enum
RotateWorld, axis, angle/sec_enum
GetAngle , axis(x/y/z) (returns float)

```
Rotate, z, -30; rotate counterclockwise, 30° per second, around objects z axis
If ( GetAngle, z == 180 )
```

The GetAngle function returns the world angle, not the local angle. Axis can be x, y, or z. Notice that like the move functions the value you are giving Rotate or RotateWorld is a speed setting (not an angle), if you want to turn an object by 90° either use set angle (for an instantaneous change) or use Rotate together with GetAngle to check how far the object has already been turned.

Positioning an object in the world or in an interior cell

```
Position, float_enum_x, float_enum_y, float_enum_z, float_enum_zRot
(for outdoors)
```

```
PositionCell, float_enum_x, float_enum_y, float_enum_z, float_enum_zRot, "cellID"
(for interior / exterior cells)
```

```
Player -> position -23515, -15355, 3355, 90
"Actor_ID" -> PositionCell, -254, 475, -376, 360, "Balmora, Council Club"
```

The classic application for this function is the teleport ring, transporting the player to certain locations. However, it can also be used to warp NPCs or objects to a new location. Note that this function only accepts literal values as arguments. (This probably changes with Tribunal). **Note:** One thing to be aware of is that using PositionCell in dialog results isn't reliable, and may cause crashes. The way Bethesda does this correctly is to use StartScript to start a script that does the teleporting. (Forum Info/Emma)

Sample Script: A simple teleport ring could look like this:

```
Begin TeleportScript
;Attached e.g. to a ring

short status
short button
short OnPCEquip

if ( MenuMode == 1 )
    return
endif

if ( OnPCEquip == 1 )
    Set Status to 10
    Set OnPCEquip to 0
Endif

If ( status == 10 ); Display menu
    MessageBox "Teleport me to", "Balmora", "Vivec", "Cancel"
    Set Status to 20
Elseif ( status == 20 ); wait for response
    Set button to GetButtonPressed
    If ( button == -1 ) ; no answer yet
        Return
    Elseif ( button == 0 ); selected Balmora
        Player -> PositionCell -21278, -17613, 534, 0, "Balmora (-3, -3)"
    Elseif ( button == 1 ); Selected Vivec
        Player -> Position 29872, -82108, 578, 180
    Elseif ( button >= 2 ); selected cancel
        Set status to 0
    Endif
Endif

End
```

Note that both targets are outdoor cells and the different formats used. If you try to teleport to an unsafe place (clipping with an object or out in the void), you will instead be placed at the next safe location.

You can also relocate NPCs or objects with this function.

CellUpdate

CellUpdate

Broken! According to Bethesda: Updates the current objects cell position. This should be called when moving objects over large distances. The game keeps tracks of objects based on what cell they are in, and if an object moves a cell over from its starting position, it may not get processed correctly when running its script.

The part about not processing correctly is certainly right. Objects can disappear or "warp" if moved too far from the place they were created in. Unfortunately my attempts to use this function always resulted in a runtime error: "need to add function code for function CellUpdate".

Note: a way around this problem (requires Tribunal) is to disable and delete (`SetDelete`) the object on a regular basis (for rideable objects upon entering a new cell) and immediately placing a new version (`PlaceItem`) at the very same position using a global script (seen in MadMax boat script from the Fishing Academy Mod). See the Tips and Tricks section for an in-depth explanation by MadMax himself. This works like a charm, because this way the object never really leaves the cell it was created in.

Setting position or facing of an object for each axis

```
SetAngle, axis, float_enum_angle  
SetAngle, z, 30
```

```
SetPos, axis, float_enum_pos  
SetPos, z, 477
```

These functions (unlike the move and moveworld functions) work also with Actors, including the player. Axis is x, y, or z. The float value sets the position or angle of the calling object to that value. This always refers to the local coordinate system the object is currently in. **Note:** With Tribunal, this function accepts float variables, but only within the currently active cells. This is relevant for exteriors, you can not move objects an arbitrary distance, the target location must be within the active cells (current cell of player plus surrounding cells). (Forum info / reposted by Srikandi).

SampleScript: This script is made for floating crates in the Mournhold sewer (Tribunal). It demonstrates how *SetPos* and *SetAngle* can be used instead of *MoveWorld* and *Rotate* to produce fluid movements:

```
begin floatAboveStartHeight  
  
float timer  
float swingTime  
float startAngle  
float startHeight  
float currange  
float xvalue  
float zvalue  
float zoffset  
float tmpoffset  
float weightoffset  
float waterlevel  
  
short reset  
short initialized  
  
if ( initialized == 0 ); this section stores the starting height and facing of the object  
    set startAngle to GetAngle, X  
    set startHeight to GetPos, Z  
    set swingTime to 1
```

```

        set initialized to 1
endif
if ( MenuMode == 0 )
    set waterlevel to GetWaterLevel
    if ( waterlevel > startHeight )
        if ( timer == 0 )
            if ( reset == 0 )
                set timer to Random 100
                set timer to timer / 4
            endif
        endif

        set timer to ( timer + GetSecondsPassed )
        set currangle to GetAngle X
        ;These set the amount to move or rotate depending on framerate:
        set xvalue to 10 * GetSecondsPassed
        set zvalue to 5 * GetSecondsPassed
        ; the crate sways around its x axis:
        ;rotate up
        if ( timer < swingTime )
            set currangle to currangle + xvalue
            SetAngle X currangle
            set zoffset to zoffset + zvalue
        ;rotate down
        elseif ( timer < (swingTime * 3) )
            set currangle to currangle - xvalue
            SetAngle X currangle
            set zoffset to zoffset - zvalue
        ;up again
        elseif (timer < (swingTime * 4) )
            set currangle to currangle + xvalue
            SetAngle X currangle
            set zoffset to zoffset + zvalue
        ;reset timer to zero
        else
            set timer to 0
            set reset to 1
            set zoffset to 0
            SetAngle, x, startangle
        endif

        set tmpoffset to waterlevel
        set tmpoffset to tmpoffset + zoffset
        ; The crate bobs up and down
        SetPos Z tmpoffset
    Else ; Waterlevel is normal
        SetAngle, X, startAngle
        SetPos Z startHeight
    endif
endif
end

```

Resetting an object to its original position

SetAtStart

Object_ID -> SetAtStart

This resets the object to the original position it was given in the editor, before any movement or rotation occurred. For an example, see the moving platform script example under the topic "moving along the world axis". See the *Move* function for a sample script.

Animating Objects

There is a group of functions that allows you to play specific animations that are defined in a model (.nif file). You can find out about the animation group names by loading a model into the preview window and then skipping through the different animation groups or by looking at "base animation" windows in the "Character" menu. An excellent summary of actor animation groups can be found here:

<http://morrowind.preik.net/animationgroups.html>

but only the ones listed in the base animation window can be called by this function. Not all models have animation groups, but the different flags (under activators) are good examples to see what is meant. Examples for GroupName are: idle, idle2, idle3, walk, etc.

```
PlayGroup, GroupName, [Flags]
```

```
PlayGroup, walk, 1
```

Plays the animation group defined by GroupName. Optional flags can be used to start the group in different ways (see below).

```
LoopGroup, GroupName, Number_enum, [Flags]
```

Plays the animation group defined by GroupName. The animation will be looped the number of times specified, and then return to the Idle animation. Optional flags can be used to start the group in different ways (see below).

```
SkipAnim
```

Causes the current animation to not be played for this frame.

Flags:

0 = Normal

The current animation will finish its full cycle, and the new animation will start from its beginning.

1 = Immediate Start

The current animation will stop regardless of the frame it is on, and the new animation will start from its beginning.

2 = Immediate Loop

The current animation will stop regardless of the frame it is on, and the new animation will start at the beginning of its loop cycle.

Example Script:

This original script is attached to all the outside banners and makes them move differently depending on the weather:

```
begin OutsideBanner

;this script is for a banner object outside that
;animates in the wind.
;Idle is still, Idle2 is a little breeze, and Idle3 is a large breeze

short ran

if ( MenuMode == 0 )
    set ran to random 100
    if ( ran < 30 ) ;30% chance the flag does something new
        if (GetCurrentWeather >= 5) ;thunder, ash, or blight
            LoopGroup, Idle3, 5
        endif
        ;the last anim called in this script is the one it will play
        if ( ran <= 10 )
            PlayGroup, Idle
        elseif ( GetCurrentWeather < 5 )
            PlayGroup, Idle2
        endif
    endif
endif
endif

end OutsideBanner
```

Determining location

Determining the players Cell

GetPCCell, "Cell_ID" (returns Boolean/short)

```
if ( GetPCCell "Balmora" == 1 )
    Set dream to 1
endif
```

The GetPCCell function tests for the player's presence in the specified cell. It returns 1 if the player is in the specified cell, 0 if not. Partial matches are supported, e.g. GetPCCell, "Vivec" will return true for the cells "Vivec", "Vivec, foreign quarter waistworks" and "Vivec, temple", etc.

Sample Script:

This small Bethesda script checks for the PC leaving a certain area until removing a certain item from an NPC:

```
Begin DrothPost

if ( GetJournalIndex "MS_EstateSale" >= 70 )
    if ( GetPCCell "Mournhold, Geon Auline's House" == 0 )
        "Geon Auline"->RemoveItem "silver dagger_droth_unique" 1
        Journal MS_EstateSale 80
        StopScript DrothPost
    endif
endif

End DrothPost
```

Determining when the PC leaves a cell

CellChanged

```
If ( CellChanged == 1)
```

CellChanged returns 1 for one frame when player changes cells. If the script calling this function is local, this function will trigger when the player enters the cell where the script is active – leaving the cell will not trigger the function, because the script terminates before the cell change registers (Thanks to Klinn for this correction). There seems to be a slight bug: teleporting out of a cell reportedly does not trigger this function (unconfirmed).

Sample Script: In the SlaveScript, which governs freeing slaves in the game, CellChanged is the trigger to disable the slave – the slave has left for a better future:

```
Begin SlaveScript
[...]
```

```
if ( slaveStatus == 3 )
    if ( GetCurrentAIPackage == 3 )
        AIWander 512 0 0 0 0 0 0 0 0 0 0
    endif
    if ( GetItemCount Slave_Bracer_Left > 0 )
        Drop Slave_Bracer_Left 1
    endif
    if ( GetItemCount Slave_Bracer_Right > 0 )
        Drop Slave_Bracer_Right 1
    endif
    if ( CellChanged == 1 )
        Disable
    endif
endif

end slaveScript
```


another nice example is the Gateway Haunt's scrip. This specter always comes back just when you are not watching:

```
Begin ResurrectHaunt

;town_Sadrith quest
;gateway_haunt resurrects until journal town_Sadrith >= 35

if ( CellChanged == 1 )
    if ( gateway_haunt->GetHealth < 1 )
        gateway_haunt->Resurrect
    endif
endif

end ResurrectHaunt
```

Distance of one object to another

GetDistance, "ObjectID" (returns float)

"ObjectID1" -> GetDistance, "ObjectID2"

This function returns the distance (in units) of one object to another. In syntax one, that is the distance between the calling object (to which script is attached to) and the named object. This can be used to trigger attacks or other events, or simply to roughly determine the player's whereabouts for use in a script.

Here is a snippet from an original Morrowind script:

```
; From a script attached to an NPC Ashamanu:
; Ashamanu will give journal entry 60 when player is near

if ( GetDisabled != 1 )
    if ( GetDistance Player <= 256 )
        if ( GetDistance "guar_white_unique" <= 256 )
            if ( GetJournalIndex "MS_WhiteGuar" <= 50 )
                Journal "MS_WhiteGuar" 60
            endif
        endif
    endif
endif

endif
```

Limitations:

- Note that you should use this function only with unique ID's or in environments where you exactly know that there is only one instance of the ID – otherwise the Game engine will just grab the first instance of the ID it finds and report that distance – most likely not the distance to the object you want. Thus, a script that warns the player of the presence of a slaughterfish that is closer than 800 units would have to be attached to the ID of the slaughterfish, and check the distance to "player" (which is unique), not vice versa.
- If you determine distance to an object you are moving with Move or MoveWorld, GetDistance will still report the **distance to the original location** of the object (the one you set in the editor). Use GetPos and good ol' Pythagoras ($c^2 = a^2 + b^2$) to determine distances in these instances.

Determining an objects position and facing

GetPos, axis (x/y/z)

Object_ID -> GetPos, z

When you are moving objects with the functions described above, you might want to obtain information on its current whereabouts. In the following sample script I used this function to control the movement of a light source (a fire) to make a fire pit where the flames actually start slowly and die down in the evening on a daily schedule – the fire objects original Z position is 511:

```
Begin _HB_Scheduled fire

short control_fire

; The script is attached to an NPC that guards the fire.
;***** this controls the fires schedule:
if ( GetDistance, "HB_Furn_De_Firepit_camp" < 600 )
    If ( GameHour < 17 )
        if ( HB_Light_Fire_camp -> GetPos Z >= 400 )
            HB_Light_Fire_camp -> MoveWorld z, -0.1 ; Move fire down
        else
            HB_Light_Fire_camp -> disable
        endif
    elseif ( GameHour >= 17)
        HB_Light_Fire_camp -> enable
        if ( HB_Light_Fire_camp -> GetPos Z < 511 )
            HB_Light_Fire_camp -> MoveWorld z, 0.1 ; move fire up
        else
            HB_Light_Fire_camp -> enable
        endif
    endif
endif

end
```

Line of Sight:

GetLOS, ObjectID (returns Boolean/short)
Actor_ID -> GetLOS, Player

Undocumented:

GetLineOfSight (returns Boolean/short?)
(maybe this one works better? Not tested yet)

This function determines whether the calling object has line-of-sight to the referenced object. It does not seem to work for non-actor type objects, as far as I could determine. It does not take facing into account, so don't take the "sight" part too literally.

Sample Script:

```
Begin balynScript
float timer
short doOnce
[...]; references to journal settings
Set timer to ( timer + GetSecondsPassed )
if ( timer < 5 ); A timer to avoid testing to often (avoids performance problems)
    Return
endif
Set timer to 0
if ( doOnce == 0 )
    if ( GetDistance Player <= 1024 )
        if ( player->GetDistance "hlaalu_loaddoor_ 02_balyn" <= 256 )
            if ( GetLOS Player == 1 )
                ForceGreeting
                Journal DA_Mephala 55
                set doOnce to -1
            endif
        endif
    endif
endif
End
```

Determine whether an Actor is detected by another Actor

GetDetected, "Actor ID" (returns Boolean/short)

If (Actor_1->GetDetected, Player == 1)

Returns true if the calling actor can detect "Actor ID". This function will return 0 if the Actor is hidden in some form, e.g. is sneaking successfully, or has an invisibility or chameleon spell active. According to the helpfile this is a slow function, do not call it a lot (e.g. make a counter to only call it every 3 seconds).

Sample script: The player must approach an object undetected – if not he is "caught"

```
Begin jeanneScript

float timer
short nolore

if ( GetJournalIndex "EB_Bone" < 20 )
    Return
endif

if ( GetJournalIndex EB_Bone >= 40 )
    Return
endif

Set timer to ( timer + GetSecondsPassed )

if ( timer < 5 ) ;this makes sure GetDetected is only called every 5 seconds
    Return
endif

Set timer to 0

if ( GetDistance Player <= 1024 )
    if ( player->GetDistance "com_chest_02 " <=128 )
        if ( GetDetected Player == 1 )
            ForceGreeting ;The player has been caught and will be punished
            Journal EB_Bone 50
        endif
    endif
endif

End jeanneScript
```

Triggers for Actors standing on an object

GetStandingPC (short) returns 1 if PC is standing on it.
GetStandingActor (short) returns 1 if ANY actor (including PC) is standing on it.

```
If ( GetStandingPC == 1)
    [... trigger horrible trap]
endif
```

This is a great function to trigger events, especially for interior cells. It is also an excellent function to build traps. You can make an "activator" object using the nif files of any static object (including hallway, carpets etc., and trigger certain events once the player (or another actor) steps on that object.

My sample script is used to turn a fires in a hall on as soon as the player steps on a particular piece of floor:

```
Begin HBHallLighting

if ( GetStandingPC == 1 )
    set HB_hallfire to 1
endif

end
```

HB_hallfire is a global variable, used to turn on the fire. This is the script for the fire:

```
Begin HBHallfireon

if ( HB_hallfire == 1)

    if ( GetPos, z, < -736 )
        MoveWorld, z, 3 ; fire rises, until its reached full height
        if ( GetPos, z, > -780)
            enable
        endif
    endif
else
    disable
endif

end
```

Hurting an actor standing on an object

HurtStandingActor, float_HP/s

```
HurtStandingActor -3
HurtStandingActor 1

Float hurt_variable
HurtStandingActor, hurt_variable
```

This function affects the health of an actor (including the PC) that stands on the object. Positive values will reduce health, negative values heal (hitpoints per second). This function accepts float variables

Sample script: The effect is probably best known from the lava fields:

```
begin lava

if ( menumode == 1 )
    return
endif

if ( CellChanged == 0 )
    if ( GetSoundPlaying "lava layer" == 0 )
        PlayLoopSound3DVP "lava layer", 1.0, 1.0
    endif
endif
```

```

endif
endif

HurtStandingActor, 20.0           ;20 pts of damage a sec

end lava

```

Changing and testing Skills, Attributes, AI and other Stats

Get, Set, and Modify stats

```

GetStat           (returns float)
SetStat, var_float
ModStat, var_float

```

This is really a whole family of functions that can alter player and actor stats, AI settings and more. Replace *Stat* with any of the game stats, attributes, AI-settings, resistances, reputation, etc. (**list see Appendix**). positive values are added to current stat, negative values subtracted. GetStat returns a float value with the **current** value of *Stat* (Not the maximum or base "natural" value of that stat for the player, but what is currently used by the game, e.g. it could be boosted by magic or reduced by disease).

SetStat sets the stat's **base and current** value to the given value.

ModStat adds (positive values are added to current stat, negative values subtracted) the given value to both the **base and current** value of *Stat*. ModStat can not set an attribute beyond its natural limit (100) while SetStat can. Presumably the behavior is equivalent for other *Stat*'s

Note: This is not true of some of the more unusual stats like resistances, which can be negative (weakness) and aren't limited to 100 either.

Special use with ModStat only:

```

ModCurrentHealth, var_float
ModCurrentMagicka, var_float
ModCurrentFatigue, var_float

```

While ModHealth changes both the maximum and the current health of an actor for the same amount (e.g. even a healthy actor would be affected), ModCurrentHealth affects only the current health and can not set health above the original maximum health value for that actor (so doing ModCurrentHealth, 10000 to an actor with 70 Health and a current health of 35 would set Health to 70 – Doing ModHealth, 10000 would set him to 10035 health).

Undocumented:

```

GetHealthGetRatio           (returns float)

```

This function returns the health ratio of the actor as a float value from 0 to 1, e.g. 1 means 100% health, 0.9 means 90% health and 0 means, well, dead I guess. This replaces the erroneously listed function GetHealthRatio listed in the original helpfile.

If you want to know an actors maximum health (Remember, GetHealth returns your *current* health points) you can use this:

```

Float MaxHealth
Float CurrentHealth
Set CurrentHealth to "Actor ID"->GetHealth
Set MaxHealth to (CurrentHealth / "Actor ID" -> GetHealthGetRatio)

```

There are so many things you can do with this set of functions that it is not very useful to provide a sample script. Take a look at the Marksman Toggle script in the Tips and Tricks section for a good example. The script given under "Resurrecting a dead actor" below also uses ModHealth, as do many others.

These commands have a wealth of applications. They could be used for special items, curses, blessings, to gain information on the player's strengths and weaknesses, and to change AI settings (making an actor more aggressive after player has insulted him, making an NPC uncommunicative at night etc.).

```
Set floatvar to ( Player -> GetHealth )
Player -> SetWillpower, 20
Player -> ModHealth, floatvar
```

Full list of Get-, Set-, and Mod-“Stat” type functions:

Attributes:

```
Get/Mod/SetStrength
Get/Mod/SetIntelligence
Get/Mod/SetWillpower
Get/Mod/SetAgility
Get/Mod/SetSpeed
Get/Mod/SetEndurance
Get/Mod/SetPersonality
Get/Mod/SetLuck
```

Health, Magicka, Fatigue:

```
Get/Mod/SetHealth
ModCurrentHealth
Get/Mod/SetMagicka
ModCurrentMagicka
Get/Mod/SetFatigue
ModCurrentFatigue
```

```
GetHealthGetRatio
```

Returns ratio of current health / full health

Skills:

Changing weapon skills can be used to change what weapon the NPC uses by default. This doesn't appear to work correctly for armor – the NPC will only wear armor based on the skills set in the TES CS. If you know the item ID, and have either Tribunal or Bloodmoon, you can use the Equip function to force the NPC to use it. (Forum Info/Vorwoda_the_Black) See the tips and tricks section for an example. What's not obvious is the range of acceptable values for a skill, it's not just 0-100 as you might expect. In fact skills appear to be stored as a float so you can set some large numbers in there, but there are some checks: you can't set negative values, and decimal points are discarded when saving/loading (Thanks FreshFish).

```
Get/Mod/SetBlock
Get/Mod/SetArmorer
Get/Mod/SetMediumArmor
Get/Mod/SetHeavyArmor
Get/Mod/SetBluntWeapon
Get/Mod/SetLongBlade
Get/Mod/SetAxe
Get/Mod/SetSpear
Get/Mod/SetAthletics
Get/Mod/SetEnchant
Get/Mod/SetDestruction
Get/Mod/SetAlteration
Get/Mod/SetIllusion
Get/Mod/SetConjuration
Get/Mod/SetMysticism
Get/Mod/SetRestoration
Get/Mod/SetAlchemy
Get/Mod/SetUnarmored
Get/Mod/SetSecurity
Get/Mod/SetSneak
Get/Mod/SetAcrobatics
Get/Mod/SetLightArmor
Get/Mod/SetShortBlade
```

```
Get/Mod/SetMarksman
Get/Mod/SetMercantile
Get/Mod/SetSpeechcraft
Get/Mod/SetHandToHand
```

Magic:

Most of these seem to refer to certain boni you normally get only from spells. With these functions you can apparently make them permanent or alter them. Most of these will normally use values between -100 to 100 (%) but will accept any number, but others are flags (0 or 1). Thus, you could make a creature that removes e.g. your ResistBlight bonus – wouldn't that be a nice surprise for our Nerevarine?

```
Get/Mod/SetResistMagicka
Get/Mod/SetResistFire
Get/Mod/SetResistFrost
Get/Mod/SetResistShock
Get/Mod/SetResistDisease
Get/Mod/SetResistBlight
Get/Mod/SetResistCorprus
Get/Mod/SetResistPoison
Get/Mod/SetResistParalysis
Get/Mod/SetChameleon
Get/Mod/SetResistNormalWeapons
```

```
Get/Mod/SetWaterBreathing
```

Setting this to 1 enables water breathing

```
Get/Mod/SetWaterWalking
```

Setting this to 1 enables water walking

```
Get/Mod/SetSwimSpeed
Get/Mod/SetSuperJump
```

These correspond to the Swift Swim and Jump spell effects, so they normally range from 0 to 100, but work with negative or higher values as well.

```
Get/Mod/SetFlying
```

I found the following info on the UESP: This sets the player's flying mode. To get this cheat to work, enter the console command and then cast a Levitate Spell. The effect should now last until you disable the flying with the console (thanks Dave Humphrey).

```
Get/Mod/SetArmorBonus   correspondsto shield effect
Get/Mod/SetCastPenalty  corresponds to "Sound" effect? (<0 makes casting harder, >0, easier)
Get/Mod/SetSilence
Get/Mod/SetBlindness
Get/Mod/SetParalysis
Get/Mod/SetInvisibile  (sic! Not invisible!)
```

```
Get/Mod/SetAttackBonus  correspondsto fortify attack effect
Get/Mod/SetDefendBonus  corresponds to sanctuary effect
```

I am not sure what these last two are, they might target the same bonus that you get with some birthsigns that give you Fortify Attack.

Other:

```
Get/Mod/SetReputation
```

```
Get/Mod/SetDisposition
```

probably refers to base disposition (as set in the TES CS, unaltered by any modifiers)

```
Get/Mod/SetPCCrimeLevel (PC Only)
```

PCCrimeLevel governs the gold you have to pay to be cleaned of crimes, influences NPC disposition and how guards react to you. See also the PayFine function.

```
Get/Mod/SetLevel (only works with Set and Get)
```

Sets the Actors Level. To my knowledge, skills and stats are not automatically increased, nor does the leveling menu come up when you call this. I have not tested it myself, however.

Formatted: English (U.K.)

AI Settings:

Get/Mod/SetFight

Changing this changes it for ALL references of the Actor. <In my limited testing this does not seem to be true>. Some info from the helpfile:

An actors fight setting determines how prone the actor is to attacking the PC. When an actor's fight setting hits 100, they will attack the PC.

Player actions will increase (or decrease) an actor's fight setting. These are:

PC Action	Default Value	Game Setting Formula
PC Distance	20 - (Char Distance * 0.005)	iFightDistanceBase - (Char Distance * fFightDistMult)
Attack Actor	100	iFightAttack
Disposition	(50 - Disposition) * 1	(50 - Disposition) * fFightDispMult
Stealing	5 * Item Value	fAlarmStealing * Item Value
Pick Pocketing	25	iAlarmPickPocket
Trespassing	25	iAlarmTresspass
Taunting	From Persuasion Formula	
Intimidation	From Persuasion Formula	
Bribery	From Persuasion Formula	

The following table gives you the resulting general behavior:

100	Always Attacks
95	Will Attack as PC gets close (3000 units)
90	Will Attack as PC gets close (2000 units)
80	Will Attack as PC gets close or if he dislikes you (1000 units, 40 Disp)
70	Will Attack if close and strong dislike (1000 units, 35 disp)
60	Will Attack if he dislikes you and you get close (Disp below 30)
50	Will Attack if he hates you (Disp at 0)
40	Will attack if he dislikes you, and you get close. (500 Units, Disp 10)
30	Will Attack if hates you and you commit crime.
20	Will Attack if dislikes you and multiple crimes.
10	Will attack if he hates you and you do multiple crimes on him.
0	Will ONLY attack if attacked first.

Get/Mod/SetFlee

Changing this changes it for ALL references of the Actor.

Setting this to a higher value will make the actor more likely to flee, but this may not always be the result, as the Actor will also use other factors like how much damage they can give out, or other strategies they may use such as magic and ranged combat. The behavior is strongly influenced by a number of GameSettings that are listed below, and a number of mods (e.g. by wakim and maxpublic) have tweaked these values to allow for more realistic fleeing behavior.

Get/Mod/SetAlarm

Changing this changes it for ALL references of the Actor.

Some info from the helpfile: When a crime is committed, and it is detected by an NPC, they will shout something at the player, this also notifies other NPCs in the area.

When the NPCs hear this, they adjust their settings based on their alarm setting. The higher the alarm setting, the angrier they will get.

If an NPC has an alarm of 100, he will put gold on the PC's head if they hear of a crime.

If the NPC with alarm 100 is also of class “Guard”, they will have extra behavior: Intercept the PC, by running up and arresting the PC.
If the PC’s CrimeLevel is over 10000, they will attack on site, instead of initiating dialogue. Guards will also attack any creatures they can see that are attacking people (including the PC).

Get/Mod/SetHello

Changing this changes it for ALL references of the Actor. Some info from the helpfile: Hello equates to the distance at which the actor will stop, face the PC and say hello. The setting (which defaults to 30) is multiplied by the game setting, iGreetDistanceMultiplier, which defaults to 7. Thus, a setting of 30 yields a hello distance of 210 (just under 10 feet).

Resurrecting a dead actor

Resurrect

gateway_haunt->Resurrect

This function brings an actor back to life. His stats and inventory will be reset, basically he "respawns". There is a bug when you used this function on the player – it would stop the PC (and all actors) from casting magic. After saving and reloading this side effect goes away. The Puzzle Canal Script simply uses GetHealth <10 to determine when player is "nearly" dead and then "resurrects" him by giving him his health back – so the player actually never really dies.

Sample Script: some people are just tougher than others...

```
Begin dandrasScript

short deathbed
float dandrasHealth

if ( deathbed == -1 )
    return
endif

set dandrasHealth to GetHealth

if ( dandrasHealth <= 50 )
    if ( dandrasHealth < 1 )
        Resurrect
        ModHealth 100
    endif
    set deathbed to 1
endif

if ( deathbed == 1 )
    ForceGreeting
endif

End dandrasScript
```

Player Controls

Player sleeping

ShowRestMenu

Brings up rest menu, and allows the player to sleep. This is used e.g. for beds in cells where it is otherwise illegal to sleep.

Sample Script: This is the standard script for beds:

```
begin Bed_Standard

;used for standard beds the player can activate and sleep in
```

```

if ( MenuMode == 0)
    if ( OnActivate == 1 )
        ShowRestMenu
    endif
endif
end

```

GetPCSleep (returns Boolean/short)

Returns true (1) if pc is sleeping. **Note:** The sleep selector and counter you see while sleeping counts as a menu. So be aware of that, if you want to use this function, and the MenuMode function in the same script!

WakeUpPC

Makes the PC wake up before the selected sleeping time is over. Sometimes creates a monster if the player was sleeping outside. This always happens if they try to sleep for only one hour, with longer times it may or may not happen (Thanks to Manauser for this info).

Sample script: This is an edited excerpt from the lengthy "sleepers" script by Bethesda. It is responsible for giving you the dreams about Dagoth Ur that plague the player during the main quest. It shows how GetPCSleep and WakeUpPC can be used:

```

if ( GetPCSleep == 0 )
    return
endif

Set dream to 0

if ( GetPCCell "Balmora" == 1 )
    Set dream to 1
endif

if ( GetPCCell "Ald-ruhn" == 1 )
    Set dream to 2
endif
[...]
if ( dream == 0 )
    Set doOnce to 0
    ;this makes sure you have to leave the city and come back for another attack to occur
    return
endif

AddTopic "Disturbing Dreams"
;add this topic, doesn't matter if you do it over and over

;THE FIRST DREAM...

if ( GetJournalIndex Al_2_AntabolisInformant >= 10 )
    if ( GetJournalIndex Al_Dreams < 1 )
        WakeUpPC
        MessageBox "You had a disturbing dream. Bla bla bla", "Ok"
        Journal Al_Dreams 1
        return
    endif
endif
endif

```

Enabling and disabling player control

Disable control functions:

All of these functions disable some part of the user interface, thus restricting the players actions.

DisablePlayerControls

Player can only look around with mouse or use options menu, nothing else and menus disappear.

```
DisablePlayerFighting
DisablePlayerMagic
```

These two functions seem to be unreliable according to forum information: If the player holds a weapon or has a spell readied he can continue to use it and quick-keys for weapons and spell likewise still seem to work. I currently don't know of a reliable solution for this problem.

```
DisablePlayerJumping
DisablePlayerLooking
DisablePlayerViewSwitch
DisableVanityMode
```

Enable control functions

Once a disable function has been used, the corresponding enable function can be used to restore control.

```
EnableLevelUpMenu
EnablePlayerControls          (Enables the controls and menus.)
EnablePlayerJumping
EnablePlayerFighting
EnablePlayerLooking
EnablePlayerMagic
EnablePlayerViewSwitch
EnableRest
EnableVanityMode
```

Check control status

All of these functions return 1 if the corresponding "Disable" function has been called and is active, 0 if control is with the player.

```
GetPlayerControlsDisabled
GetPlayerFightingDisabled
GetPlayerJumpingDisabled
GetPlayerMagicDisabled
GetPlayerLookingDisabled
GetPlayerViewSwitch (Broken, function does not work. Instead use:
GetVanityModeDisabled
```

Force first or third person view

PCGet3rdPerson (returns Boolean/short)

returns 1 if in 3rd person mode

PCForce3rdPerson

queue the change to 3rd person mode (this may have to wait for the animation to finish)

PCForce1stPerson

same as above but 1st person mode

Functions for character generation menus

These undocumented functions are used for character creation. They enable all the menus that are used during the character creation process and enable certain basic features like the inventory, magic menu, stats window and the map:

Show character generation menus:

EnableBirthMenu

EnableClassMenu

EnableRaceMenu

EnableNameMenu

There is no disable command for these. They are disabled by selecting ok in the menu.

Enabling in-game menus:

EnableMagicMenu

EnableMapMenu

EnableInventoryMenu

EnableStatsMenu

Also no disable commands here unfortunately. These would have been useful.

The names of the functions should be self-explanatory. One use for these functions is as a cheat if you want to change your appearance or other things during a running game (although there might be problems associated with doing that). They can (and have been) used to create different ways of character generation. Be careful, sometimes these reset your level to 1.

Sample Script: This is one of many CharGen scripts that guide the player through character generation. This one is basically a safety feature for a player who just runs out the door, without triggering any or all of the little tutorials.

```
Begin CharGenDoorExit

;this is the door that exits the first part of the census building
;safety check for all menus on

short done

if (done == 1)
    return
endif

if ( OnActivate == 1 )
    enablestatsmenu
    enableinventorymenu
    enablemagicmenu
    enablemapmenu
    enableplayerfighting
    enableplayermagic
    set done to 1
    Activate
endif

End
```

Weather

Changing weather

```
ChangeWeather, "RegionID", short_Type_Enum
```

```
ChangeWeather, "West Gash", 4
```

This function changes the weather in the indicated region to the weather type specified by TypeEnum, and will change again to according to the region settings after the time set by the game (I assume that is set in the Morrowind.ini file in the Weather section. In mine the entry reads:

```
Hours Between Weather Changes=20
```

The weather TypeEnum values are:

0	Clear
1	Cloudy
2	Foggy
3	Overcast
4	Rain
5	Thunder
6	Ash
7	Blight

Changing weather settings for a region

```
ModRegion, "RegionID", clear_enum, cloudy_enum, foggy_enum, overcast_enum, rain_enum, thunder_enum, ash_enum, blight_enum
```

```
ModRegion, "West Gash", 10, 20, 10, 5, 5, 40, 10, 0
```

Changes the weather chances for the RegionID. Used to get rid of, or add weathers to an area permanently. The values must add up to 100 or you will get odd results.

Determining current weather

```
GetCurrentWeather (returns short)
```

```
If ( GetCurrentWeather == 1)  
[it's cloudy]
```

This returns the weather TypeEnum listed above.

Sample script: Bethesda used this to make the banners move in the wind according to weather type:

```
begin OutsideBanner  
  
;this script is for a banner object outside that  
;animates in the wind.  
;Idle is still, Idle2 is a little breeze, and Idle3 is a large breeze  
  
short ran  
  
if ( MenuMode == 0 )  
    set ran to random 100  
    if ( ran < 30 ) ;30% chance the flag does something new  
        ;this will check the weather in the future  
        if ( GetCurrentWeather >= 5 ) ;thunder, ash, or blight  
            LoopGroup, Idle3, 5  
        endif  
  
        ;the last anim called in this script is the one it will play  
        if ( ran <= 10 )  
            PlayGroup, Idle
```

```

elseif ( GetCurrentWeather < 5 )
    PlayGroup, Idle2

endif

endif

endif

```

Detecting wind speed

Undocumented:

GetWindSpeed (returns float)

I have only very briefly tested this function and it returns values, 0 indoors and floats outdoors (varying quickly, in overcast weather the values seemed to oscillate around 2). (Thanks to XPCagey for finding this)

Miscellaneous functions and variables

Making someone fall

Fall

Seems to give an NPC the extra nudge they may need even after you yank the floor out from under them. It also brings down flying creatures. Used by the Icarian Flight guy.

Determining if player has menus open

MenuMode

MenuMode returns one if the player has activated the menu (the inventory screen). It is common practice to put the following lines at the beginning of almost any script, to avoid that the script processes while the player is in the inventory. Dialogue, the sleep timer, the console and probably pretty much any menu will activate this too.

```

If ( MenuMode == 1 )
    Return
Endif

```

Using MenuTest to close inventory

Undocumented:

MenuTest

MenuTest doesn't return anything, however, when it is called, it closes certain types of inventory menus, including Player, NPC and containers. It doesn't work for dialogue, enchanting, alchemy, spell or armorer menus. (Forum Info / JOG)

```

if ( OnPCEquip == 1 )
    set OnPCEquip to 0
    coc Balmora
    MenuTest
endif

```

Fading the screen in and out

FadeIn time_float_enum
 FadeOut time_float_enum

FadeTo alpha_enum time_float_enum

```
FadeTo 50 2.0 ; (Fades screen to 50% in 2 seconds)
```

FadeIn and Fadeout fades the screen (not an object) to blackness in the time specified (in seconds). Time is > 0 and <= 10.0. FadeTo fades only to a certain percentage: 0 is full transparency. 100 is black.

Adding a location to the map

```
ShowMap "cell ID"
```

```
ShowMap "Gnisis"
```

This function will highlight the indicated cells. Cell ID can be full or partial, i.e. all cells that begin with the given string will be highlighted on the world map (e.g. ShowMap "Vivec" will highlight all the cantons).

Sample Script: reading this book will indicate all these places on the world map:

```
Begin bookPilgrimsPath
if ( GetJournalIndex TT_PilgrimsPath >= 100 )
    Return
endif

if ( OnActivate == 1 )
    Journal TT_PilgrimsPath 100
    ShowMap "Gnisis"
    ShowMap "Vivec"
    ShowMap "Ghostgate"
    ShowMap "Koal Cave Entrance"
    ShowMap "Fields of Kumm"
    Activate
endif

End
```

Detecting if player is indoors or outdoors

```
GetInterior (returns Boolean/short)
```

```
If ( GetInterior == 1 )
```

Undocumented function! (Thanks XP-Cagey and Killgore)

This function will return 1 if the current cell is an interior cell and 0 if it is an exterior cell.

The following is a global sample script by Killgore. If you want to try it out start it by typing "StartScript Outside_Check" in the console.

```
Begin Outside_Check
short doonce

if (MenuMode == 1)
    Return
EndIf

if (doOnce == 0) ;if you're in some brand new cell
;or it just started

if ( GetInterior == 1 )
    MessageBox "1: inside"
elseif ( GetInterior == 0 )
    MessageBox "0: outside"
else
    MessageBox "mystery else"
endif

set doOnce to 1
Return
endif

if (doOnce == 1)
```

```

if (CellChanged == 0) Return
else ;if the player changes cells that frame..
set doOnce to 2 ;it waits an extra frame
endif
Return
endif

if (doOnce == 2) ;then starts over and prints
set doOnce to 0
Return
endif

End Outside_Check

```

Breaking of script processing

Return

Return tells the game engine to finish processing the script for this frame. All code below this line will be ignored for this frame. In the next frame the script is executed again from the top.

```

If ( MenuMode == 1 )
    Return
Endif

```

Careful: Anything below a return function will not be processed, even if there are "true" if statements there! So use this with caution.

Assigning random values to variables

```

Random, value
Set my_variable to Random, 50

```

Introducing some unpredictability into the effects of a script is a nice option, and it can be done with the *Random* function. *Random* returns values between 0 and the set value -1. So in the example above, my_variable will be set to a value in the range from 0 to 49.

Note that the global short variable Random100 gets set each frame by the games *Main* script to a random value between 0 and 100, so you can make use of that one, too.

Playing videos

```

PlayBink "filename" flag_enum

```

Pauses game and plays video. Set Flag to true if player can escape movie. The video needs to be in Bink format and placed into the Datafiles/Videos directory. MW defaults to loading videos from the CD, so I am not sure if this will actually work. It's possible (not tested) that setting "TryArchiveFirst=-1" in the Morrowind.ini file could have an influence on this (-1 Use raw data, 0 Use Newer, 1 use Archive Only). Otherwise you may have to resort to a no-CD crack to play your custom made videos.

Controlling global scripts

```

ScriptRunning, "ScriptName" (returns Boolean/short)

StartScript, "ScriptName"

StopScript, "ScriptName"

```

These function are used to control global scripts. Global scripts have to be started with the StartScript function (either from another local or global script or from a dialogue result field) and a running script can be terminated with the StopScript function.

Using 'StopScript' on a global script resets any local variables used by that script. If you use 'StopScript' from inside the global script you're terminating, it doesn't actually terminate the

script immediately. Instead, the script continues executing to the 'End' statement, and then terminates. Tribunal "Start Scripts" in a plug-in start executing each time the game is loaded. (I assume the ones that come with the game do too, but I wasn't testing those). If a Tribunal Start Script is terminated with a 'StopScript', it will start up again the next time the game is loaded. (Thanks DinkumThinkum).

To my knowledge these functions do not work with local scripts attached to objects, you can however start "targeted scripts" by using ObjectID->StartScript (for more info see the Tips and Tricks section below).

The ScriptRunning function returns 1 if a script is running, 0 if it's not running:

```
if ( ScriptRunning, CharGen == 0 )
    StartScript CharGen
Endif
```

StopScript can also be used to construct do-once conditions for global scripts in a very clean way by self-terminating the script:

```
Begin do-once_script
[...]; do stuff here
StopScript do-once_script
End
```

New functions that come with TRIBUNAL

The expansion for Morrowind, Tribunal, also introduces a number of new functions and fixes and extends several of the old ones. Note that both you and the user of the mod must have Tribunal installed to make use of these, so make sure you mark your mod accordingly. This part of the document is based on an internal document that Bethesda was kind enough to send my way (many thanks to Mike Lipari). I only reformatted it and added a few comments.

Note: These functions are also included in the Bloodmoon expansion, so any mods made with just Tribunal can be resaved with Morrowind/Bloodmoon without problems.

Changes / fixes to Morrowind scripting :

- SetPos now accepts variables (float) as arguments.
- SetAngle now accepts variables (float) as arguments.
- Equip now works as intended and forces NPC to equip (put on) armor and clothing.
- AIActivate was apparently fixed

Index of new Tribunal Script Functions:

GetPC Sneaking	ForceJump
GetPC Running	ClearForceJump
GetPC Jumping	GetForceJump
GetScale	ForceMoveJump
SetScale	ClearForceMoveJump
ModScale	GetForceMoveJump
GetWaterLevel	GetCollidingPC
SetWaterLevel	GetCollidingActor
ModWaterLevel	HurtCollidingActor
EnableLevitation	AddToLevCreature
DisableLevitation	AddToLevItem
PlaceItem	RemoveFromLevCreature
GetWeaponType	RemoveFromLevItem
GetArmorType	GetSquareRoot
HasItemEquipped	ExplodeSpell
ForceRun	SetDelete
ClearForceRun	GetWeaponDrawn
GetForceRun	GetSpellReadied

Enable equipment sharing

```
Short companion  
Set companion == 1
```

Tribunal introduced the option to "share" equipment with NPC's or creatures. To enable this option, you must define a local short variable named "companion" and set it to 1. Setting it to 0 will disable sharing. This method is used both for mercenaries and for pack animals.

```
Float minimumprofit
```

Seems to be another variable set by the game, it is probably the difference between current value of all goods and gold minus starting value. If this gets negative, the mercenary can be scripted to quit.

Sample Script: here is the relevant part from Calvus' (the mercenary in Mournhold) script. This section handles changes in state when Calvus leaves a contract, either because the contract expires, or because the player has taken Calvus' stuff. The sharing is initiated from dialogue (setting companion to 1), not in the script itself

```
if ( GetJournalIndex Merc_Calvus_Quit < 1 ) ;if Calvus has already quit, don't do this  
    if ( Contract_Calvus == 1 ) ;if Calvus doesn't have a contract, don't do this  
        if ( minimumProfit < 0 ) ;Calvus is quitting because player took his stuff  
            Set Companion to 0; stop sharing  
            StopScript Contract_Calvus  
            Set Contract_Calvus to 0  
            ForceGreeting  
            return  
        else  
            if ( Contract_Calvus == 0 ) ;handles Calvus after a contract expires  
                AiWander 128 6 0 40 30 20 0 0 0 0 0 0  
                Set Companion to 0; stop sharing  
                if ( GetJournalIndex Merc_Calvus < 10 )  
                    Journal Merc_Calvus 10;first contract expired  
                else  
                    Journal Merc_Calvus 20; most recent contract expired  
                endif  
            endif  
        endif  
    endif  
endif
```

```
short StayOutside  
set StayOutside to 1
```

A useful variable for use with companions. When used in a script, it causes whoever it's assigned to to automatically remain (and wait) outside of any interior the player may enter (automatically rejoins upon return).(Forum info / Grumpy)

PC Action functions

```
GetPCSneaking (short)  
GetPCRunning (short)  
GetPCJumping (short)
```

These functions returns 1 if the PC is performing the appropriate action and 0 if he is not.

Sample script:

When this script is placed on an NPC, and the player has an item equipped called "scissors", MessageBox warnings will be given based on the player's current actions.

```

Begin momscript

short warn

if ( player->HasItemEquipped "scissors" )
    if ( warn != 1 )
        if ( GetPCRunning )
            MessageBox "Don't run with scissors!"
            set warn to 1
        endif
    endif
    if ( warn != 2 )
        if ( GetPCJumping )
            MessageBox "Don't jump with those scissors! You'll put your eye out!"
            set warn to 2
        endif
    endif
    if ( warn != 3 )
        if ( GetPCSneaking )
            MessageBox "You can't hide those scissors from me!"
            set warn to 3
        endif
    endif
else
    set warn to 0
endif

end

```

Scale Functions

```

GetScale (float)
SetScale newScale_float
ModScale scaleChange_float

```

```

If (doonce == 0 )
    SetScale 0.1
    Set doonce to 1
endif

```

These functions are used to determine or modify the scale of a reference. There seem to be no actual limits to the scale you can set (so you can set it beyond 0.5 and 2, contrary to what was written in the original description by Bethesda). The above sample script can thus be used to set scale beyond the limits of what the TES CS itself allows.

You shouldn't call "setscale" in every frame, at least not in exteriors or other FPS-critical situations. But keep in mind that the scale will be reset to the 0.5 - 2.0 frame when you reload. So don't use a done-flag to call it only once. Either call it regularly (about all 10 frames) or test for "getscale" and reset the scale when it doesn't fit. Another way is to set it in a Tribunal or Bloonmoon start script. This can be done once, and will make sure the scale is set each time you load a game. (Forum info / JOG).

Sample script:

A more complex sample script by Bethesda shows how to grow and shrink an object.

When this script is placed on a reference, activating that reference gives the user the ability to change that reference's scale.

```

Begin scalescript

short questionAsked
short button

float direction
float currscale
float tempScale

if ( MenuMode )

```

```

        return
endif
if ( OnActivate == 1 )
    if ( questionAsked == 0 )
        MessageBox, "Make this object..." "...Grow" "...Shrink"
        set questionAsked to 1
    endif
endif
if ( questionAsked == 1 )
    set button to GetButtonPressed
    if ( button == -1 )
    else
        if ( button == 0 )
            set direction to 1
        elseif ( button == 1 )
            set direction to -1
        endif
        set questionAsked to 0
        set button to 0
    endif
endif
if ( direction != 0 )
    set tempscale to .3 * GetSecondsPassed
    set tempscale to tempscale * direction
    ModScale tempscale
    set currscale to GetScale
    if ( direction == -1 )
        if ( currscale <= .5 )
            set direction to 0
        endif
    else
        if ( currscale >= 2 )
            set direction to 0
        endif
    endif
endif
endif
end scalescript

```

Water Level Functions

```

GetWaterLevel (float)
SetWaterLevel newWaterLevel_float
ModWaterLevel waterLevelChange_float

```

A great opportunity for cruel traps... These functions are used to determine and modify the Water Level of the current interior cell. When an actor suddenly finds itself underwater, it will wait until it is halfway out of breath and then begin to make its way to the surface in a straight line up. Floating corpses are moved with the water without regard for collision.

Sample scripts:

This script goes on a crank to make it raise or lower the water level in the room.

```

Begin crank

short changelevel
float direction
float waterlift
short crankturn
short currcrank
float newwaterlevel

if ( MenuMode )
    return
endif

if ( OnActivate == 1 )
    if ( changelevel == 0 )
        if ( direction == 1 )
            set direction to -1

```

```

        else
            set direction to 1
        endif
        set changelevel to 1
    endif
endif
if ( changelevel == 0 )
    return
endif

set crankturn to 360 * GetSecondsPassed
set crankturn to crankturn * direction
set currcrank to GetAngle X
set crankturn to currcrank + crankturn
SetAngle X crankturn

set waterlift to 120 * GetSecondsPassed
set waterlift to waterlift * direction
ModWaterLevel waterlift

set newwaterlevel to GetWaterLevel
if ( direction == 1 )
    if ( newwaterlevel >= 600 )
        SetWaterLevel 600
        set changelevel to 0
    endif
else
    if ( newwaterlevel <= 0 )
        SetWaterLevel 0
        set changelevel to 0
    endif
endif
end crank

```

This modified “Float” script is placed on any object with a centered pivot point to make it float on the water’s surface regardless of water level. It also will make the object stop bobbing if the PC is standing on it.

```

Begin NewFloat

float timer
float swingTime
float startAngle
float currangle
short reset

float xvalue
float zvalue
float zoffset
float tmpoffset
float weightoffset

set startAngle to GetStartingAngle, x

if ( MenuMode == 0 )

    if ( timer == 0 )
        if ( reset == 0 )
            set timer to Random 100
            set timer to timer / 4
        endif
    endif

    set swingTime to 1

    set timer to ( timer + GetSecondsPassed )
    set currangle to GetAngle X
    set xvalue to 10 * GetSecondsPassed
    set zvalue to 5 * GetSecondsPassed

    if ( GetStandingPC )
        set zoffset to -30
        SetAngle X 0
    else

```

```

;rotate up
if ( timer < swingTime )

    set currangle to currangle + xvalue
    SetAngle X currangle
    set zoffset to zoffset + zvalue

;rotate down
elseif ( timer < (swingTime * 3) )

    set currangle to currangle - xvalue
    SetAngle X currangle
    set zoffset to zoffset - zvalue

;up again
elseif (timer < (swingTime * 4) )

    set currangle to currangle + xvalue
    SetAngle X currangle
    set zoffset to zoffset + zvalue

;reset timer to zero
else
    set timer to 0
    set reset to 1
    set zoffset to 0
    SetAngle, x, startangle

endif
endif

set tmpoffset to GetWaterLevel
set tmpoffset to tmpoffset + zoffset

SetPos Z tmpoffset

endif

end NewFloat

```

Levitation Functions

EnableLevitation
DisableLevitation

These functions are used to allow and block the functioning of Levitation magic effects. When DisableLevitation is called, all existing Levitation effects are canceled. When the player tries to cast a spell with a Levitate effect while Levitation is disabled, a notify message is displayed with the text in the GameSetting sLevitateDisabled. Currently this text reads "Levitation magic does not work here."

Sample scripts:

This script is on an object in the room with levitation disabled.

```

Begin clampstone

short turnedoff
short gavemessage

if ( turnedoff == 0 )
    DisableLevitation
    if ( gavemessage == 0 )
        set gavemessage to 1
        MessageBox "A strange stone in the roof of this room prevents levitation here."
    Endif
else
    EnableLevitation
    if ( gavemessage == 1 )
        set gavemessage to 0
        MessageBox "The stone has been disabled. You can now levitate in this room."
    Endif
endif

```

```

if ( OnActivate == 1 )
    if ( turnedoff == 0 )
        set turnedoff to 1
    else
        set turnedoff to 0
    endif
endif
end

```

This script is on a door leaving the room.

```

Begin enable_lev_on_exit

if ( OnActivate == 1 )
    MessageBox "You leave the presence of the stone..."
    EnableLevitation
    Activate
endif

end

```

Object Creation

```

PlaceItem "object ID", float_var_X, float_var_Y, float_var_Z, float_var_Zrot
PlaceItemCell "object ID", "cellID", X, Y, Z, Zrot

```

These functions are used to create new references to objects. PlaceItem will create a reference to object "Object ID" at coordinate (X, Y, Z) with Z rotation Zrot in the current cell. The function accepts (local) float variables. PlaceItemCell does the same thing as PlaceItem except it allows you to specify a cell other than the current one in which the object should be created. With either function, if the target cell for the reference is an exterior cell and the given coordinate is outside of that cell, then the reference will be added to the cell containing the coordinate.

This is a nice addition that allows you to add things to the world without previously placing them in the editor.

JOG posted the following interesting info on PlaceItemCell:

Well, When I first tried PlaceitemCell, I thought it's a great function to place items instead of having a "storage-cell" where the objects lie around until the game needs them. I soon realized that you can't refer to those objects by script. (To disable them for example.) The script won't compile until at least one of the objects is in use, and then the disable command would refer to that object, so you still need PositionCell.

So the only use left for PlaceitemCell was placing items that already exist in the game. I thought you could make a daedric dagger for example, that appears at a certain point in a quest, without having to create a new object.

Now it seems, the use of PlaceitemCell, is restricted to placing standard-objects in the same cell as the player is, on a random position dependent on another object. (Like a NPC that drops coins while walking around)

DinkumThinkum adds:

PlaceItemCell would have been perfect for what I wanted to do, until I discovered that the placed NPC disappeared if I saved and reloaded before going to the cell they were placed in.

Sample Script:

Putting this script on an object causes it, on activation, to ask the user for an object type and then to create a reference to the specified object at 100 units above the activated reference with 45 degree Z rotation. If the key is selected, it is created at that same coordinate and rotation in the cell "key room" rather than the current cell.


```

Begin makethingsimple

short questionAsked
short button
float myX
float myY
float myZ

if ( MenuMode )
    return
endif

if ( OnActivate == 1 )
    if ( questionAsked == 0 )
        MessageBox, "Create new..." "...Pot" "...Key"
        set questionAsked to 1
        set myX to GetPos X
        set myY to GetPos Y
        set myZ to GetPos Z + 100
    endif
endif

if ( questionAsked != 0 )
    if ( questionAsked == 1 )
        set button to GetButtonPressed
        if ( button == -1 )
            else
                if ( button == 0 )
                    PlaceItem "Misc_pot_redware_01" myX myY myZ 45
                elseif ( button == 1 )
                    PlaceItemCell "misc key" "key room" myX myY myZ 45
                endif
                set questionAsked to 0
                set button to -1
            endif
        endif
    endif
endif

end

```

Worn / equipped Object Information

GetWeaponType (short)
GetArmorType armorPart_enum (short)
HasItemEquipped "item_ID" (short)

These functions are called on an actor to gather information regarding what the actor has equipped. GetWeaponType returns the weapon type (see Table 1.1) of the actor's current weapon. GetArmorType returns the armor weight (see Table 1.3) of the actor's currently equipped armor part. The armor parts are coded by the numbers listed below (see Table 1.2). HasItemEquipped returns 1 if the actor has the given item currently equipped and 0 if it does not.

Weapon types (Table 1.1):

Weapon Type Name	Type Number
Unarmed	-1
Short blade, 1 hand	0
Long blade, 1 hand	1
Long blade, 2 hand close	2
Blunt, 1 hand	3
Blunt, 2 hand close	4
Blunt, 2 hand wide	5
Spear, 2 hand wide	6
Axe, 1 hand	7
Axe, 2 close	8
Bow	9
Crossbow	10
Thrown weapon	11
Arrow<?>	12
Bolt<?>	13

Armor parts (Table 1.2):

Armor Part Name	Part Number
Helmet	0
Cuirass	1
Left Pauldron	2
Right Pauldron	3
Greaves	4
Boots	5
Left Gauntlet	6
Right Gauntlet	7
Shield	8
Left Bracer	9
Right Bracer	10

Armor types (Table 1.3):

Armor Type Name	Type Number
Unarmored	-1
Light Armor	0
Medium Armor	1
Heavy Armor	2

Sample Script:

When this script is placed on an object, Activating a reference to that object does “damage” to it based on the PC’s current weapon and strength. If the weapon is the specific weapon “Rock Splitter” the object is fully damaged in one hit. When the object is fully damaged, it explodes sending shards into the PC’s face unless he has a shield or a helmet equipped.

```
Begin breakme

float hitsleft
float hitpercent
short damage
short tempdamage
short weapon
short doOnce
short shieldType
short hasHammer
short hitRock

if ( doOnce == 0 )
    set hitsleft to 10000
    set doOnce to 1
endif

if ( OnActivate )
    set hasHammer to ( player->HasItemEquipped "RockSplitter" )
    if ( hasHammer == 1 )
        MessageBox "Rock Splitter unleashes its mighty force..."
        set hitsLeft to 0
    else
        MessageBox "You hit the rock with your current weapon..."
        set weapon to ( player->GetWeaponType )
        set damage to ( player->getstrength )
        set tempdamage to 5

        if ( weapon == -1 )
            set tempdamage to 1
        endif
        if ( weapon >= 9 )
            set tempdamage to 2
        endif
        if ( weapon == 4 )
            set tempdamage to 10
        endif
        if ( weapon == 8 )
            set tempdamage to 8
        endif

        set damage to damage * tempdamage

        set hitsleft to hitsleft - damage
    endif

    if ( hitsleft <= 0 )
        disable
        set shieldType to ( player->GetArmorType 8 )
        if ( shieldType == -1 )
            set shieldType to ( player->GetArmorType 0 )
            if ( shieldType == -1 )
                MessageBox "...and the rock shatters sending jagged shards into your eyes."
                Player->ModHealth -50
            else
                MessageBox "...and the rock shatters, deadly shards glancing off your helmet."
            Endif
        else
            MessageBox "...and the rock shatters, deadly shards glancing off your shield."
        Endif
    else
        set hitpercent to hitsleft / 100
        set hitpercent to 100 - hitpercent
        MessageBox "...and the rock is %.2f percent damaged but remains intact.", hitpercent
    endif
endif
```

NPC Movement Functions

```
ForceRun  
ClearForceRun  
GetForceRun (short)
```

```
ForceJump  
ClearForceJump  
GetForceJump (short)
```

```
ForceMoveJump  
ClearForceMoveJump  
GetForceMoveJump (short)
```

These functions all control the specified NPC's movement. The ForceRun function makes the NPC always run when they move, the ForceJump function makes the NPC constantly jump, and the ForceMoveJump makes the NPC always jump when they are moving. The Get versions of the functions return one if the specified NPC currently is forced into the given action and zero otherwise. The Clear functions are used to turn off the forced movement. An NPC can only be forced to do one movement at a time. The priority for forced movement is Sneak > Running > Jump > MoveJump.

Sample Script:

This script lets an object control the movement type of Athlete, an NPC set to Travel endlessly in a four-point square.

```
Begin AthleteControl  
  
short questionAsked  
short button  
short isrunning  
short isjumping  
  
if ( MenuMode )  
    return  
endif  
  
if ( OnActivate == 1 )  
    set isrunning to ( Athlete->GetForceRun )  
    set isjumping to ( Athlete->GetForceMoveJump )  
    if ( questionAsked == 0 )  
        if ( isrunning )  
            MessageBox, "Make Athlete stop running? " "Yes" "No"  
        else  
            MessageBox, "Make Athlete run? " "Yes" "No"  
        endif  
        set questionAsked to 1  
    endif  
endif  
  
if ( questionAsked == 1 )  
    set button to GetButtonPressed  
    if ( button == -1 )  
        else  
            if ( isrunning == 0 )  
                if ( button == 0 )  
                    Athlete->ClearForceMoveJump  
                    Athlete->ForceRun  
                endif  
            else  
                if ( button == 0 )  
                    Athlete->ClearForceRun  
                endif  
            endif  
            if ( isjumping )  
                MessageBox, "Make Athlete stop jumping? " "Yes" "No"  
            else  
                MessageBox, "Make Athlete jump? " "Yes" "No"  
            endif  
            set questionAsked to 2  
        endif  
    endif  
endif
```

```

        set button to -1
    endif
endif

if ( questionAsked == 2 )
    set button to GetButtonPressed
    if ( button == -1 )
        else
            if ( isjumping == 0 )
                if ( button == 0 )
                    Athlete->ClearForceRun
                    Athlete->ForceMoveJump
                endif
            else
                if ( button == 0 )
                    Athlete->ClearForceMoveJump
                endif
            endif
        endif
        set questionAsked to 0
        set button to -1
    endif
endif

end

```

Object Collision Functions

GetCollidingPC (short)

GetCollidingActor (short)

```
if ( GetCollidingPC == 1 )
```

HurtCollidingActor Damage

```
HurtCollidingActor 100
```

These functions go on an object to allow it to interact with Actors colliding with it. GetCollidingPC returns 1 if the PC is currently colliding with the object and 0 otherwise. GetCollidingActor works the same as the PC version of the function but will return 1 if any Actor (other than the PC) is colliding with the object. HurtCollidingActor lowers the colliding Actor's health Damage points every second.

Sample script:

When this script is placed on an object, any actor touching that object will take damage. A different message is given depending on whether the actor is the PC or someone else.

```

Begin hurtactor

if ( GetCollidingPC == 1 )
    MessageBox "You scream in pain as you touch the rock."
Elseif ( GetCollidingActor == 1 )
    MessageBox "Nearby someone screams in pain."
Endif

HurtCollidingActor 100

End

```

Leveled List Functions

```
AddToLevCreature      "levcreaname" "creature" level_enum
AddToLevItem "levitemname" "item" level
RemoveFromLevCreature "levcreaname" "creature" level_enum
RemoveFromLevItem "levitemname" "item" level_enum
```

These functions are used to manipulate Leveled Item and Leveled Creature lists at run-time. Leveled lists are comprised of object/level pairs where the level is the level the PC has to be to encounter the object. The AddTo functions will add the given object/level pair to the specified leveled list as long as the list does not already contain a matching pair. The RemoveFrom functions will remove all occurrences of the object/pair from the leveled list. Additionally, if a RemoveFrom function is given an object pair with a level of -1, all object pairs containing the specified object are removed.

NOTE: The RemoveFrom functions will not remove existing objects from the world. If a Leveled Creature reference has already calculated to be a certain creature, removing that creature from the Leveled Creature's list will not get rid of the existing creature in the world. However it will prevent that Leveled Creature reference from calculating to be that creature again.

Sample Script:

When this script is placed on an object, activating it will toggle the existence of rats in the world by removing them from a Leveled Creature and removing rat meat from a Leveled Item.

```
Begin norats

short norats

if ( OnActivate == 1 )
    if ( norats == 0 )
        set norats to 1
        RemoveFromLevCreature "rat_scamp_crab" "rat" -1
        RemoveFromLevCreature "rat_scamp_crab" "rat-fast" -1
        RemoveFromLevItem "lev_meat" "rat_meat" -1
        MessageBox "No more rats."
    Else
        set norats to 0
        AddToLevCreature "rat_scamp_crab" "rat" 1
        AddToLevCreature "rat_scamp_crab" "rat-fast" 1
        AddToLevItem "lev_meat" "rat_meat" 1
        MessageBox "The rats return."
    Endif
endif

end
```

Miscellaneous Functions

Square root

GetSquareRoot, number (float)

```
set var_1 to GetSquareRoot var_2
```

The GetSquareRoot function returns the square root of the given number. This can be useful for vector calculations.

Explosion

ExplodeSpell "spellName"

```
ExplodeSpell "proj_trap_spell"
```

The ExplodeSpell function makes a reference cast the given touch range spell at itself. If an area effect touch range spell is used, this can make the reference “explode”.

Deleting a reference completely

SetDelete flag (flag is literal value)

The SetDelete function can be used in combination with Disable to remove an object more completely. SetDelete 1 marks a reference for deletion and SetDelete 0 clears that flag. This can be useful in optimization. Certain objects have scripts on them which simulate picking them up by disabling the activated reference and adding a new object to your inventory. This leaves an ever-present, but disabled, reference at that location (which eats up memory and processor time since the script on the disabled reference is still run every frame). If the reference is marked for deletion then it is essentially gone. If the reference came from the master file, it is still there but knows it shouldn't be so has no art and no scripting. If was created in game, it will actually be deleted.

A remark by Soralis from the forums:

There are a couple of things that should be done to make this work, as I had some problems with using it crashing the game otherwise. If you use this on an object, give it a bit of time of inactivity before you delete it, and Disable it ahead of time. So you could put something like this in your script near the top, as an example:

```
if ( deleteobj = 1 ) ;Local variable, set when you want to delete
  if ( deletetimer == 0 )
    Disable
  endif
  if ( deletetimer < 10 )
    set deletetimer to ( deletetimer + 1 )
  endif
  if ( deletetimer == 10 )
    SetDelete, 1
  endif
  Return
endif
```

Also, you should always call SetDelete from a local script assigned to the object you want to remove. Never use Object->SetDelete 1 as this usually causes crashes when Morrowind is running. If an item is in your inventory, it shouldn't be deleted since this may cause you encumbrance to become wrong. If you really need to delete an item, and you know that the

PC is carrying it, you can try using the Drop command before disabling and deleting. One last thing to be aware of is that sometimes magic effects can also cause problems with the SetDelete function.(Forum Info/Dan_Wheeler)

Sample Script:

When this script is placed on an object, as soon as it is placed in the world (or encountered if it was placed in the editor) it will calculate where the player is and move towards that location at a steady rate (determined with GetSquareRoot). When that point is reached, or if the object is ever within range of the player, it explodes (with ExplodeSpell) and disables. Once it has detonated, it waits a few frames for the spell system to clear, then deletes itself (with SetDelete).

```
Begin trapProjScript

short range
short initialized
short distance
short detonate
short triggered

float targx
float targy
float targz
float shiftx
float shifty
float shiftz
float currshift
float currx
float curry
float currz
float totaldist
float rate

float killtimer

if ( triggered == 1 )
    if ( killtimer < 4 )
        set killtimer to ( killtimer + GetSecondsPassed )
    else
        SetDelete 1
    endif
    return
endif

if ( MenuMode == 1 )
    return
endif

if ( initialized == 0 )
    set initialized to 1
    set range to 150
    set rate to 300
    set targx to ( player->GetPos X )
    set targy to ( player->GetPos Y )
    set targz to ( player->GetPos Z )
    set shiftx to ( targx - GetPos X )
    set shifty to ( targy - GetPos Y )
    set shiftz to ( targz - GetPos Z )
    set totaldist to ( ( shiftx * shiftx ) + ( shifty * shifty ) + ( shiftz * shiftz ) )
    set totaldist to GetSquareRoot totaldist
    if ( totaldist != 0 )
        set shiftx to ( shiftx / totaldist )
        set shiftx to ( shiftx * rate )
        set shifty to ( shifty / totaldist )
        set shifty to ( shifty * rate )
        set shiftz to ( shiftz / totaldist )
        set shiftz to ( shiftz * rate )
    else
        set triggered to 1
        return
    endif
endif

endif
```



```

set distance to GetDistance "player"
if ( distance < range )
    set detonate to 1
else
    set currx to GetPos X
    set curry to GetPos Y
    set currz to GetPos Z

    set currshift to ( shiftx * GetSecondsPassed )
    set currx to ( currx + currshift )

    set currshift to ( shifty * GetSecondsPassed )
    set curry to ( curry + currshift )

    set currshift to ( shiftz * GetSecondsPassed )
    set currz to ( currz + currshift )

    if ( shiftx < 0 )
        if ( currx < targx )
            set detonate to 1
            set currx to targx
        else
            set detonate to 0
        endif
    else
        if ( currx > targx )
            set detonate to 1
            set currx to targx
        else
            set detonate to 0
        endif
    endif

    if ( shifty < 0 )
        if ( curry < targy )
            set detonate to 1
            set curry to targy
        else
            set detonate to 0
        endif
    else
        if ( curry > targy )
            set detonate to 1
            set curry to targy
        else
            set detonate to 0
        endif
    endif

    if ( shiftz < 0 )
        if ( currz < targz )
            set detonate to 1
            set currz to targz
        else
            set detonate to 0
        endif
    else
        if ( currz > targz )
            set detonate to 1
            set currz to targz
        else
            set detonate to 0
        endif
    endif

    SetPos X currx
    SetPos Y curry
    SetPos Z currz
endif

if ( detonate == 1 )
    ExplodeSpell "proj_trap_spell"
    set triggered to 1
    disable
endif

end

```

Combat Readiness Functions

GetWeaponDrawn (short)
GetSpellReadied (short)

```
if ( player -> GetWeaponDrawn )
```

These functions can be used to determine whether or not an actor has their weapon out or whether or not they have a spell readied for casting.

Sample Script: This global script gives notification messages based on the player's weapon and spell states.

```
Begin player_notifications

short weapstate
short spelstate

if ( player->GetWeaponDrawn )
    if ( weapstate != 1 )
        set weapstate to 1
        MessageBox "The player's weapon is drawn."
    Endif
else
    if ( weapstate != 0 )
        set weapstate to 0
        MessageBox "The player's weapon is sheathed."
    Endif
endif

if ( player->GetSpellReadied )
    if ( spelstate != 1 )
        set spelstate to 1
        MessageBox "The player's spell is readied."
    Endif
else
    if ( spelstate != 0 )
        set spelstate to 0
        MessageBox "The player's spell is put away."
    Endif
endif

end
```

New functions that come with BLOODMOON

The second expansion for Morrowind, Bloodmoon, introduces a few new functions. Note that both you and the user of the mod must have Bloodmoon installed to make use of these, so make sure you mark your mod accordingly.

Index of new Tribunal Script Functions:

GetPCTraveling	TurnMoonRed
GetPCInJail	GetWerewolfKills
PlaceAtMe	IsWerewolf
SetWerewolfAcrobatics	UndoWerewolf
TurnMoonWhite	BecomeWerewolf

GetPCTraveling and GetPCInJail

GetPCTraveling
GetPCInJail

Bloodmoon adds these two ‘variables’ that can be checked to see if the PC is either travelling (i.e. on a Silt Strider) or in Jail. This is used in the werewolf change script to stop the PC from changing if either of these states are the case.

```
if ( PCWerewolf != 1 ) ; DON' RUN IF PLAYER ISN'T WEREWOLF
    return
endif

if ( GetPCInJail == 1 )
    return
endif

if ( GetPCTraveling == 1 )
    return
endif
```

Place items near an object

Object->PlaceAtMe 'Item_ID' count distance direction

The PlaceAtMe function works the same as PlaceAtPC without it being centered on the PC. Bloodmoon uses this to place attackers in different places depending on the player's distance at the time. This allows the script to make it appear as though there is a large number of opponents that just keep coming, among other things.

```
;THIS POPS IN A HUNTER AT APPROPRIATE SPOT, INCREMENTS HUNTERCOUNT, AND RESETS TIMER
if ( popA == 1 )
    "active_BM_hunter1"->PlaceAtMe skaal_hunter 1 1 1
    set huntercount to ( huntercount + 1 )
    set timer to 0
elseif ( popB == 1 )
    "active_BM_hunter2"->PlaceAtMe skaal_hunter 1 1 1
    set huntercount to ( huntercount + 1 )
    set timer to 0
elseif ( popC == 1 )
    "active_BM_hunter3"->PlaceAtMe skaal_hunter 1 1 1
    set huntercount to ( huntercount + 1 )
    set timer to 0
endif
```

Set the werewolf attributes

Object->SetWerewolfAcrobatics

This function set the attributes of the object to those of a werewolf. This sets the targets skills and attributes to match the fWerewolfxxxx gameplay settings. In most cases, this means a high strength, agility, acrobatics etc, and 0 in most other things.

```
Player->AddSpell "werewolf vision"
Player->AddSpell "werewolf regeneration"
Player->SetWereWolfAcrobatics
```

Change the color of Secunda

TurnMoonWhite
TurnMoonRed

These two functions are very simple – they change the color of Secunda (The small, white moon) from white to red and back again. This doesn't have any real effect to gameplay, but it does make the sky look different. It is used during the Bloodmoon main quest – hence the expansion title.

```
if ( doOnce == 0 )
    TurnMoonRed
    set doOnce to 1
endif
```

Determine how many kills a werewolf has

GetWerewolfKills

This keeps count of how many NPC's killed by the werewolf. Each time an NPC is killed while the PC is a werewolf, one is added to this count. It is reset automatically when the PC changes back into human form.

```
if ( GetWerewolfKills > 0 )
    ; Do code to stop the PC from being affected by the hunger.
endif
```

Check to see if the creature is in werewolf form

Object->IsWerewolf

This 'variable' determines if the target is a werewolf or not. It can be used on the PC or other creatures.

```
if ( Player->IsWerewolf != 1 ) ;DON'T RUN IF PLAYER ISN'T WEREWOLF
    return
endif
```

Change to a werewolf

Object->BecomeWerewolf
Object->UndoWerewolf

These functions change the target object to a Werewolf or change them back to their original form. **IMPORTANT:** Using Becomewerewolf and Undowerewolf CAN break your game. Some quests and variables depend solely on on use of these, so if you use one to toy around.... you may be asking for it. (This message brought to you by your friendly dev WormGod)

```
if ( OnPCEquip == 1 )
    Player->BecomeWereWolf
    Set OnPCEquip to 0
Endif

Set timer to ( timer + GetSecondsPassed )

If ( timer > 10 )
    Player->UndoWereWolf
Endif
```

Tips and tricks

(I hope this section will greatly improve in future versions: send me your own scripting tricks and I will put them here if they are good.)

Script with style for safer scripting

This is bound to be a bit controversial as it is as much about personal style as it is about fact. And it's bound to sound snobbish ☺. Nevertheless, I think this might be of some use for the newcomer, so here are a few comments about my personal views on good style and safer scripting:

- Use annotations. In a really short script they might seem useless, but even there you might want to state which mod/quest/item they belong to or what their general purpose is, etc. For long scripts this becomes indispensable, for yourself, if you stop working for a few days, for others that might want to learn from your script. Explain your variables, put headers on the main part of your script, comment on important lines of code
- Use state variables with style. Basically, due to the "executed once a frame" nature of the scripts this is your principal way of structuring your script for sequential events. There are several things to this.
 - A) Limit yourself to the minimal amount you need for the script.
 - B) use elseifs to chain different states of one state variable together, not separate if-blocks – this should be the main structural element of your script (it's not always possible nor necessary, but if it is, it's going to save you a lot of trouble).
 - C) Check the elseifs are arranged from lowest to highest and in a logical order of events – will help you to keep everything organized, and thus avoids bugs. Jumping around wildly with state variables is the equivalent to careless use of GOTO in good old BASIC.
 - D) Use them extensively, do small steps. I can't tell you how many times bugged scripts started working simply because I moved some functions to a separate "state-block". Sometimes this doesn't seem to be logical at all – but if you can safely enter another step, do it.
- Decide for one style. TES Script is relatively forgiving regarding syntax. You can write functions small letters or capitalized as in this document, or all caps. You can use (if SomeFunction == 1) or if (somefunction). But whatever you choose, try to be consistent in your usage.
- Use verbose variable names. A name that reflects the function of this variable makes a script much more readable. If you use global variables give them a unique name, e.g. put your initials in front or whatever – just try to minimize the chance that another mod will come up with the same name for a global – because that would screw things up royally.
- Keep track of your Return function uses. The Return function is inherently dangerous – remember that it will stop anything in that script below that line from being executed. Use it, but use it sparingly. If you have the feeling you have to use it a lot in one script, you should probably introduce a state variable instead.

Cleaning up your mod

When you work on your mod, you will probably want to look up other things as reference or simply to cut and copy things for your own purposes. The problem is that the TESCS will remember you looked at the things if you ever hit an "OK" button, even if you did not change

anything you didn't want to change. Before you release your mod you should check it for such unwanted references and remove them.

In the File Menu select Data Files. Then select your mod like you normally would and hit the Details button. Look at the list of features here: this is a list of everything your mod changed or added. Look for things that you did not want to change, select them and hit [del] on your keyboard. This marks the feature as ignored. Loading and resaving the mod will remove these ignored features from the mod.

An alternative, and maybe easier to use is the utility TESAME (TES advanced mod editor), available from <http://www.sphosting.com/scarabus/tesame.html>.

For scripts, try to remove any unused global variables or whole scripts you may have made in the course of developing your script. It bloats your .esp filesize, it probably wastes memory, or at the very least it looks bad.

This is a list of the change indicators found in the details tab:

DIAL - A new or changed topic

INFO - A dialogue response, or journal entry

REFR - a reference of an object that was put into the game world, while MISC, CONT etc. describe the actual Object (even if there is no instance of it placed in the world)

SOUN - A Sound

NPC_ - A new type of NPC, or a changed NPC

CREA - A new creature or a change to a creature

LIGH - A new or changed Light

LTEX - An application of a landscape texture

PGRD - A change to the AI grid

CELL - obvious, signifies a changed cell either indoors or outdoors - in case your mod had nothing to do with that cell you should get rid of it. This affects all changes in that cell automatically, I think automatically

SCRPT - A Script - lots of people leave superfluous "test scripts" in their mods - that's bad style, I think.

MISC - A new or changed miscellaneous object - check names, delete if the change is unrelated to your plugin

ACTI - An activator - see above

CONT - A container - very critical: make sure you only change a new ID (copy of a container), not the original container - or they will all be changed.

STAT - A static object - see above

Careful with cleaning up dialogue: When you enter new dialogue into a topic, that also changes the topics above and below the one you inserted - that is because the responses in topics are a linked list, each contains the information about the next line, which allows quick processing of the topics to search for the correct response. Do not clean up topics that show up changed due to this - it will create an error telling you that the next line is different for a specific response in a topic. You can recreate the link by moving the topic in question up and down, to reregister the order of topics with the TES CS.

Limits of the Script Editor

Character Limit: There is a limit of the maximum number of characters per script. It is somewhere around 30000 characters (the true limit is most likely 32767, which is the max value for a 16-bit signed integer, which is how script length is stored in the .esp - thanks to

Horatio for this info). If this occurs you can no longer type in the editor window. To save characters try the following:

- Remove characters
- Use shorter variable names
- See if the script can be split and some part maybe handled in a global script or attached to a separate object as a separate script.

Line Limit: there also is reportedly a maximum line limit. This seems to vary and reports on the forum range between 900-and 1500 lines of code it's probably rather a limit of the compiled script than the actual lines of text, so empty lines and comments don't count. This is reported by an error message upon saving the script.

If-elseif limit: There is a limit on the maximum number of if-elseif conditions that can be used per script. I am not sure of the absolute number (I heard both 127 and 256). Also there is a maximum depth of nested if commands, it's reportedly 10 (thanks Riiak)

Making Actors switch between weapons

Since the equip function does not work, the only way to do this is to take items away from the Actor, or to modify his skills at runtime.

Here is an example I used to make a guard switch between bow and sword:

```
Begin HBCaravanGuardAI

; this script makes the AI guard more dangerous by making him switch from bow to sword when
the player closes in
short currentarrows
short storearrows
short doonce

set currentarrows to GetItemCount "arrow of wasting flame"

if ( doonce == 0 )
    set storearrows to currentarrows
endif

if ( GetDistance, Player < 120 )
    set currentarrows to GetItemCount "arrow of wasting flame"
    if ( currentarrows > 0 )
        RemoveItem "arrow of wasting flame", 1
        set doonce to 1
    endif
endif

elseif ( GetDistance, Player >= 120 )
    if ( currentarrows < storearrows )
        AddItem "arrow of wasting flame", 1
    else
        set doonce to 0
    endif
endif

End
```

The next example is one by Bethesda, which does the same thing, using the skill change method (admittedly more elegant than mine ☺):

```
begin marksmanToggle

short counter
short myMarksman

if ( MenuMode == 1 )
    return
```



```

endif

if ( counter < 20 )
    Set counter to counter + 1
    Return
endif

if ( myMarksman == 0 )
    set myMarksman to GetMarksman
endif

if ( GetMarksman > 0 )
    if ( GetDistance Player < 400 )
        SetMarksman 0
    endif
else
    if ( GetDistance Player > 600 )
        SetMarksman myMarksman
    endif
endif

;for level designers... forces AI to do what it ought to do
;when they are near, they use melee weapons
;when they are far, they use missile weapons
;checks every 20 frames for speed
;Note: does not affect spellcasting AI

End

```

Little helpers

A good function to use for the beginning scripter is the search text function in the main TESCS edit menu. You can use it to search scripts, e.g. for a specific function you would like to use and want to find sample scripts for.

You can also use any text-editor you prefer for editing your scripts and copy / paste the script into the editor using ctrl-c / ctrl-v.

Testing for the presence of another Mod

(by Ragnar_GD)

Some people want to know from a script, if some other, specific Plug-In is loaded. This is possible under certain circumstances.

It is possible if the Plugin you want to test for

- a) has a fairly unique global variable
- b) sets this global variable from a script

If you know of such a Global, implement this Global in your own script as well. Then check during run-time, if the value within the Global is set to any value, like, assuming the other Mod has a Global named "ForeignGlobal", and you installed this Global as well:

```

If (ForeignGlobal != 0)
    MessageBox "I detected the foreign Plug-In!"
endif

```

This method does not work, if the aforementioned Global has a pre-set value, as with this method, it will be overwritten. or the foreign plug-in has not yet changed the value to a number other than 0 by the time this check is performed

Strong advice: Given this, **never** implement a global variable in your PlugIn with a pre-set value, but instead let it start with its default (Zero), and fill it with a script later (i.e. an autostart-script), as for if someone implements a Global of this name as well, your Plug-In

might crash.

I consider the fact that you can overwrite the Globals of another PlugIn without warning a serious bug!

(Note by GBG: – one more reason why people should try to use unique names wherever possible in their mods – don't name your global "check" at least call it (your initials)_check, e.g. "YI_check" – that avoids lots of problems and compatibility issues)

Safely starting global scripts- avoiding the main script

Many modders like to add a line to the main script (the only script that is started by default when a new game is started, and always runs) to make sure a global script is started that is essential to their mod. For example:

```
StartScript "My_Script"
```

While this works, it might easily cause conflicts with mods that used the same approach – because only the changes of the last loaded plugin will be active in the game. There are some good alternatives to using the main script. Putting an (invisible) activator in the Seyda Neen census office with the following attached script will make sure the script is started during character creation.

```
Begin Script_launcher
If ( ScriptRunning, My_Script == 0 )
    StartScript, My_Script
    ; MessageBox "MyScript was activated" ; tell the player if you like
endif
End
```

To make sure a global script is started in an already running game you can use a similar method, and place activators in commonly visited cells. If you have one in Balmora, Vivec, Sadrith Mora, Dagon Fel, and Caldera and maybe the PC strongholds, I am sure it won't take long until the script is running. You can also use an object that is required for your mod anyway to start it. E.g. for Indestructibles excellent Bank mod, I made myself a version that attaches the above script to the banner of the bank and starts the "interest" script. That makes sure that that script is running before the PC ever sets foot inside the bank.

With TRIBUNAL, this issue is a problem no more – just add the script you want to start running to the list of "Start Scripts" by selecting Edit Start Scripts from the Gameplay menu. This script will now be automatically started when a game is loaded, just like the main script.

Saving CPU time

If you plan a mod with lots of scripts or long and involved scripts, you may want to give some thought to not wasting CPU power. There are a number of things you can do here:

If the script does not have to be executed every frame, put a little counter in there:

```
Begin My_super_long_script

Short framecounter
If ( framecounter < 10 )
    set framecounter to ( framecounter + 1 )
    Return
Endif
set framecounter to 0

[super long script goes here]
end
```

This little piece of code, that should always be at the very top of your script, will allow your script (or rather the main, CPU power eating part of it) to be executed only every 10th frame. You could do the same thing with a timer, and execute the script only every 3 seconds or once every minute.

Execute script only if the player is suitably near. If you have scripted a fancy magic bouncing ball, or basically anything that is a visible effect, there is no reason to run the script if the player cannot see it. So put something like this on top of your script:

```
If ( GetDistance, player < 5000 )
    Return
Endif
```

Shortcut scripts that are no longer needed. If you have local scripts that you may not need from a certain point onwards, e.g. because of Actor death or because the object was disabled, reduce their CPU need by putting something like the following at the top of the script:

```
If ( GetDisabled == 1 )
    Return
Endif

If ( GetHealth <= 0 )
    Return
Endif
```

Terminate global scripts. Remember, global scripts are running all the time until you stop them again with StopScript. You can do do-once global scripts by just putting a StopScript command at their end

```
Begin do_once_global_script

[your code here]

StopScript "do_once_global_script"

End
```

Try to use local scripts instead of global scripts. With local scripts you are sure that they only run when you are in the vicinity. Think hard before making a script global if it can be done with a local script instead.

Be careful with while-loops, GetDetected, GetLOS and other "slow" functions. Use methods as described above (e.g. a counter or a timer) to make sure they are not called too often.

Stop script while in menu mode. Always, unless you have specific reason not to, put the following at the top, to avoid the mouse lagging in the menu and other unwanted effects.

```
If ( MenuMode == 1)
    Return
Endif
```

Use sound to detect events

This to me was a very smart idea (thanks to BalorNG), so it bears mentioning again here, although it's also described in the sound section above. You can use the GetSoundPlaying function to determine certain events in the game that would otherwise not be accessible. Just take a look at the sounds in the Gameplay/sounds menu, and it may give you some ideas: determine if someone is falling, determine whether a certain monster is near, determine a hit with a weapon etc.

Here is some more info on this (thanks Horatio):

GetSoundPlaying is a very powerful command that can be used to detect when the PC (and I'm assuming other actors) is doing a certain action like casting a spell or swinging a weapon. I used it in my spellcasting mod to determine when the PC is casting a spell and what school of magic the spell is in. the format is as follows:

```
if ( player->GetSoundPlaying, "Sound ID" == 1 )
;do something cool here
endif
```

look in the sounds menu in the TESCS to find which Sound ID corresponds to a specific action. For instance "illusion cast" corresponds to the player casting an illusion based spell. You'll probably have to experiment a little. Note: for some reason the Sound ID "drink" causes an error, so no checking if the PC is drinking a potion.

Large battles:

(by Horatio)

The easiest way to do a large battle between two groups of NPCs is to use the AI commands. Let's use an example of a bunch of imperial legionnaires, with whom the PC is aligned, versus a dark brotherhood gang. First set the 'fight' rating (in the AI tab) of the DB NPC's to 100 so that they'll attack the PC on sight. Then, you'll need to set the AI of the legionnaires to:

AIFollow, player, 0,0,0,0,0

you can do this either with scripts attached to the legion NPCs or with an external script. The default behavior of AIFollow is to attack whatever is attacking the person they are following. So when the DB guys attack the PC, all the legionnaires will freak out and starting attacking them back. Presto instant giant melee.

I use a variation on this in the GIANTS mod to convince the guards to attack monsters that are actually NPCs (vampires, shades, giants, gorgos, etc).

Targeted scripts: running "global" scripts tied to an object

```
"Object_ID" -> StartScript "Script_ ID"
```

It is possible to use the StartScript function to run global scripts that are tied to an object or actor. These scripts resemble both local scripts (in that the functions called always default to the object or actor the script targets) and global scripts (in that they are always running). This is from a post by FreshFish, who found this really amazing technique:

"This is what the helpfile has to say on StartScript:

This function starts a script running. This is a Global script. It is not attached to any object, so functions like moving, rotating, checking distances and such have no bearing in a global script.

This is nonsense, ever tried starting a script from dialogue? Or from an NPC's default script? Well it works just fine, any AiTravel or PositionCell functions in the script are applied to the object they were started on. I'm going to call these 'targeted' scripts. Targeted scripts run all the time just like globals so you can get your NPC to do stuff when you are in other cells. And if you start another script from a targeted script then that script inherits the same target so you can fork 'em off or chain 'em up or whatever.

So it looks like the limitation of only one small script per object is no more.

I should point out that a targeted script still differs from a 'local' script in that variables defined in the targeted script are not considered local to the object from the point of view of dialogue and other scripts.

If a script is stopped then restarted any variables are preserved, including doOnce type variables so if you are planning on using your script again you may need to reinitialize some stuff yourself.

I don't know what happens if you try to start a targeted script on more than one target simultaneously but I doubt it would be pretty."

>can you start one script on an object, then after it stops, start a second,

>different script on the same object?

You can start a second different script on the same object any time you like, no need to wait for the first to finish. Whether this causes a conflict or not is down to what you do in your scripts, can't walk in two directions at once, obviously.

>what about objects that already have a "local" script attached?

>are there any conflicts?

No problems here either, I have an NPC with a local script which puts on a robe when it rains and also uses a 'targeted' script to make her follow another NPC sometimes.

And some more info from a post by Riiak on the Morrowind Mods forum:

- a) Variables located in targeted scripts are NOT considered local and are basically unusable in dialogue. This is because the script is not a local script it is global.
- b) A targeted script can be started from generic dialogue on a generic NPC. Thus you could write a dialogue for say "Crusaders" and start the script from there regardless of character ID. (This can allow for some very interesting situations). In the same vein is a suggestion by Cortex: You can use generic voice bits (like the sounds Actors say when you approach, when they are initiating combat, or when they are hit to start these scripts, since they, like all dialogue have a result field. This opens up a lot of possibilities!
- c) Targeted scripts can generically access an NPC's inventory (with the same limitations as regularly accessing the NPC's inventory, must know item ID, etc...)
- d) (As far as I can tell) Can not be used for Tribunal's companion sharing, but can be used to set up a simple follower script.
- e) (As far as I can tell) Can have multiple copies of same script targeting the same NPC. (could be helpful, but most likely will only cause problems)

I would recommend that the script be self stopping as that would be the easiest way to have the script clean itself up and prevent multiple copies running.

When using targeted scripts from dialogue, the script apparently always attaches to the NPC calling the dialogue response, not a referenced object: Object_ID -> StartScript "script_name" will not work, the script still attaches to the NPC saying the dialogue (Forum info / Argent).

A guide to making rideable objects

(By MadMax_001)

MadMax shares his insights on how to make rideable objects (e.g. boats) here, especially which problems will arise and what he did to solve them. There is a lot of interesting info here that goes beyond just this specific application, though. You can look at his scripts in the "Fishing Academy" and the "Magic Carpet" Mods (but don't use them in your mod without his consent).

Selecting objects

Practically all objects (statics/activators) can be used. However, selecting the right type of objects is paramount. It will make your scripting a lot easier later on. Now, what type of objects are suitable? Preference is given to small and minimum height (thickness). The other important factor is the center point which is also the point where your character will be standing on. This will also save you a lot of programming work later on. If the object center point is not what you want, you can fix this by importing the object to 3DS and move the axis to the point to where you plan your character to stand. I am not going to explain in details how to do it in 3DS, there should be quite a number of tutorials out there which teaches you how to do it.

Creating/Deleting objects

I am sure a lot of modders know that you can only move an object through the exterior cells within certain distances. This is because the game will only update and process objects/codes within a certain distance. When your character gets out of that parameter, the object will actually be frozen or to the player, the object has warped/disappeared into thin air. But, if you trace back to the original cell, the object will re-appear.

Now, to move object throughout the entire exterior cells, the trick to use here is to create a new object. Everytime when you move to a new cell, the function "CellChanged" will become TRUE for one frame. This is the best time to replace the existing object with a new one. There is one BIG problem that I discovered here. NEVER create an object from an object script. It doesn't seems to work. So, what you need to do is to use a global script to create one instead. At the same time, do not forget to Delete the old object or you will have all these objects spreading in different cells which will cause you major problems later on. Always maintain one object at one time. Below is a simple example you can use:

```
;-----  
; Object script  
;-----  
if ( player->CellChanged == 1 )  
Startscript, "Create_obj_script" ; this is the global script  
set obj_count to ( obj_count - 1 ) ; global parameter to count how many object exist  
Disable  
SetDelete, 1  
endif
```

```
;-----  
; Global script  
;-----  
PlaceAtPC "objectname", 1, 0, 0 ; or you can use PlaceItem  
set obj_count to ( obj_count + 1 )  
Stopscript "create_obj_script"
```

Ideally, the above script should work like a charm but in reality it is going to cause you major problems. Deleting an object immediately after changing cell may sometimes cause CTD. This is especially true when you have a heavy area loading. To fix this problem, delay the deletion. Introduce a time delay (I personally find the 1.5seconds to be OK so far). Here's how the object script will looks like now.

```
if ( player->CellChanged == 1 )  
Startscript, "Create_obj_script" ; this is the global script  
Disable  
set timer_flag to 1  
endif  
  
if ( timer_flag == 1 )
```

```

set timer to ( timer + GetSecondsPassed )
if ( timer > 1.5 )
set obj_count to ( obj_count - 1 )
SetDelete, 1
else
return ; stop all other code processing
endif
endif

```

That is not all. If you plan to move your object at very high speed. There is a possibility that the object may encounter another cell change during the 1.5 seconds delay. You must make sure that the object gets deleted before it gets out of the processing parameter or it will come back and haunt you later. Now the script looks like this.

```

if ( player->CellChanged == 1 )
if ( timer_flag == 1 )
SetDelete, 1
return
endif
Startscript, "Create_obj_script"
Disable
set timer_flag to 1
endif

if ( timer_flag == 1 )
set timer to ( timer + GetSecondsPassed )
if ( timer > 1.5 )
SetDelete, 1
else
return
endif
endif
endif

```

Falling off from objects

This is actually very simple. Remove the gravitational effect on the character. This can be achieved by either introducing a levitation or floating ability to the character. The former being the perfect solution but it will disable your ability to make use of detecting RUN and SNEAK buttons. The latter allows you to detect movements but at the expense of possible falloff. You must also be wondering why your character falls off in the first place. The fact is that everytime when you create an object, the object collision parameter are not updated. It will be updated though when you change cell. You will find out that you can actually clip through the object with your character. The good news is that there is a way to work around this. By Disable and then Enable the object again will automatically update this. To make sure that the object is always "SOLID", perform the Disable and Enable at least once in every frame in the object script.

Collision detection

This is biggest headache of all. Objects will not collide with objects. Only character/NPC/creatures can collide with objects. In this term, objects means statics/activators and landmass. The only way to detect collision is when your character hit the object. That is why I mentioned earlier that if you select a big object to ride, you will see clipping until the moment your character hit something. If you can live with that, that's fine otherwise it looks pretty awkward. The simplest method to detect collision is to get your player coordinates and measure against the object coordinates (the one you are riding on). Although, this is not fully proven, using GetSquareRoot function in the object script can sometimes cause CTD. There are 2 planes that you need to take care of. For eg, the flying carpet uses detection for both vertical (z axis) and horizontal (x, y axis) planes. You can refer to my script on how this is done. If somebody can come up with a better method, please do share it.

Savegame issue

If you have not realised already, when you save your game while in motion, the object coordinates updated in the your savegame are the ones during the cell changed. In order to position the object correctly, it is always best to keep a global parameters of the object coordinates. When you first load the game, do a detection on the object existing coordinates vs its global coordinates. If there is a big discrepancy, set the object to the global coordinates. In this way, when you first load the game, the object will be in precisely the same position when you save it.

Mannequins

These are popular with many people as they are a nice way to show off your collected armor – you find them in many house mods – this shows how it is done (many thanks to Stephen Kent aka Riiak Shi Nal for sharing the script). *This example is a "next generation" one that uses Tribunal functions for checking weapon/armor to prevent PC from moving mannequin while weapons (Riiak has not yet figured out how to get mannequins to wield weapons) and/or armor are present on mannequin. Also split into two separate scripts to support both male and female versions of the mannequin. These changes do not prevent PC from picking up the mannequin while there are still misc items on it, those items will be lost.*

I have added some extra comments in addition to Riiak's. The Mannequin is in reality a normal NPC with 0 health or corpse (health set to 0 in TES CS). For this version you simply activate it to give it items, and it will equip armor you give it.

```
Begin rsn_mannequin_f_script

short button
short questionState
short nEquipType
short nStillEquipped
float fDeleteTimer

SkipAnim ;GBG: This is essential, it makes the Mannequin, which is an NPC, stand still

if ( menumode == 1 )
return
endif

if ( GetDisabled == 1 )
; if mannequin has been disabled then we need to wait some time then delete this reference
Set fDeleteTimer to ( fDeleteTimer + GetSecondsPassed )
if ( fDeleteTimer > 5 )
SetDelete 1
endif
return
endif

if ( OnActivate == 0 )
if ( questionState == 0 )
return
endif
endif

if ( questionState == 0 )
MessageBox, "Armor Mannequin", "Move Mannequin", "Add/Remove Armor"
set questionState to 1
endif

if ( questionState == 1 )
set button to GetButtonPressed

if ( button == 0 )
set questionState to 10
elseif ( button == 1 )
set questionState to 0
Activate
endif
endif
```



```

if ( questionState == 10 )
; This section is split into two groups of nested ifs because of nested if limits of
; the scripting language.
Set nStillEquipped to 0
; Here we check to see if a weapon is equipped (put in mainly for if anyone can figure
; out how to get a mannequin to show a weapon)
Set nEquipType to ( GetWeaponType )
if ( nEquipType == -1 )
; Here we check to see if any armor is equipped (NOTE: there are 10 different possible
; pieces of armor so we need to check each individually)
Set nEquipType to ( GetArmorType 0 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 1 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 2 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 3 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 4 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 5 )
if ( nEquipType != -1 )
Set nStillEquipped to 1 ;GBG: Set to 1 if there is still some armor equipped
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
endif

if ( nStillEquipped != 1 ) ;We only want to process if we haven't found any equipment.
Set nEquipType to ( GetArmorType 6 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 7 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 8 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 9 )
if ( nEquipType == -1 )
Set nEquipType to ( GetArmorType 10 )
If ( nEquipType == -1 )
;Only show this question if the mannequin doesn't have weapons or armor equipped.
MessageBox "Did you remove all items from the mannequin?", "Yes", "No"
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
else
Set nStillEquipped to 1
endif
endif
endif

; Now we need to go to the next stage of processing (either wait for user choice or
immediately activate)
set questionState to 20
endif

```

```

if ( questionState == 20 )
if ( nStillEquipped != 1 )
set button to GetButtonPressed
else
; Mannequin still has weapons or armor equipped so we want activate it and warn
; the user instead of allowing pick-up.
    MessageBox "You haven't removed your equipment."
    Set button to 1 ;says that there are still items on the Mannequin
endif

if ( button == 0 )
    set questionState to 0
; Disable the current mannequin
; and create a new one we wouldn't have to worry about losing items)
; GBG: if Mannequins are transported a lot
; a SetDelete function might be a good idea
    Disable
; This is the item that contains the script
; to generate a new mannequin when dropped.
; GBG: for a "female" mannequin this would be a different item
player->addItem, "_rsn_man_f_holder", 1
playSound "Item Misc Up"
elseif ( button == 1 )
; There are still items on the mannequin (either from checks or user response)
set questionState to 0
Activate
endif
endif
end

```

The following scriüpt goes on the item that's added to our inventory when we move the mannequin. When you drop it, a new mannequin is created at your feet, hence the necessary removal of armor. You will lose it all otherwise:

```

; Script split into two scripts to handle the two different mannequin genders.

Begin rsn_man_f_holder_script

short OnPCDrop
float fDeleteTimer

if ( GetDisabled == 1 )
; if holder has been disabled then we need to wait some time then delete this reference
Set fDeleteTimer to ( fDeleteTimer + GetSecondsPassed )
if ( fDeleteTimer > 5 )
SetDelete 1
endif
return
endif

if ( OnPCDrop == 1 )
Disable
; This is the NPC with 0 health that is really just a standing corpse.
PlaceAtPC, "_rsn_mannequin_female", 1, 0, 0
Set OnPCDrop to 0
endif
end

```

Is she looking at me?

Here is a beautiful script by Horatio that allows to detect whether an actor looks towards the player:

```

Begin PCLookAtMe

float fPCX
float fPCY
float fPCAngle
float fdx
float fdy
float fRatio

```

```

short sPCLookAtMe

set sPCLookAtMe to 1

;you could probably also add a GetLOS check in here
; however i could never get GetLOS to work properly...although i didn't try very hard

;is PC really far away
if ( GetDistance, Player > 8000 )
    set sPCLookAtMe to 0
else
;yay trigonometry
;this basically does a rough calculation of the PCs direction relative
;to the actor, it only uses 45 degree chunks, though

    set fPCX to ( player->GetPos, X )
    set fPCY to ( player->GetPos, Y )
    set fPCAngle to ( player->GetAngle, Z )

    set fdx to GetPos, X
    set fdy to GetPos, Y

    set fdx to ( fdx - fPCX )
    set fdy to ( fdy - fPCY )

    set fRatio to ( fdx / fdy )

    if ( fdx > 0 )
        if ( fdy > 0 )
            if ( fRatio > 1 )
                if ( fPCAngle < -45 )
                    set sPCLookAtMe to 0
                endif
            else
                if ( fPCAngle < -90 )
                    set sPCLookAtMe to 0
                endif

                if ( fPCAngle > 135 )
                    set sPCLookAtMe to 0
                endif
            endif
        else
            if ( fRatio < -1 )
                if ( fPCAngle < 0 )
                    if ( fPCAngle > -135 )
                        set sPCLookAtMe to 0
                    endif
                endif
            else
                if ( fPCAngle < 45 )
                    if ( fPCAngle > -90 )
                        set sPCLookAtMe to 0
                    endif
                endif
            endif
        endif
    else
        if ( fdy > 0 )
            if ( fRatio < -1 )
                if ( fPCAngle > 45 )
                    set sPCLookAtMe to 0
                endif
            else
                if ( fPCAngle > 90 )
                    set sPCLookAtMe to 0
                endif

                if ( fPCAngle < -135 )
                    set sPCLookAtMe to 0
                endif
            endif
        else
            if ( fRatio > 1 )

```

```

        if ( fPCAngle > 0 )
            if ( fPCAngle < 135 )
                set sPCLookAtMe to 0
            endif
        endif
    else
        if ( fPCAngle > -35 )
            if ( fPCAngle < 90 )
                set sPCLookAtMe to 0
            endif
        endif
    endif
endif

endif

endif

endif

endif

endif

if ( sPCLookAtMe == 0 )
;do something while the PC is not looking
endif

End

```

Cinematic sequence

The following is a very smart approach to do a cinematic sequence by gianluca (Morrowind Summit forums). It removes player control, places the player on an invisible "CollisionWall" object and then moves him (and thus the camera) around. You can't have cinematic sequences involving the PC, but it's still great.

```

If menumode==1
return
endif

if doOnce==0
"Collision wall2"->disable
"Collision wall3"->disable
"Collision wall4"->disable
set doOnce to 1
endif

if doOnce==1
"Collision wall1"->moveworld X 800
messagebox "moving"
if ( "Player"->getPos Z < 570 )
set doOnce to 2
set playxx to "Player"->getPos X
set playyy to "Player"->getPos Y
set playzz to "Player"->getPos Z
"Collision wall2"->enable
"Player"->position -114679 -4119 590 90
endif
endif

if doOnce==2
"Collision wall2"->moveworld X 800
messagebox "moving"
if ( "Player"->getPos Z < 570 )
set playxx to "Player"->getPos X
set playyy to "Player"->getPos Y
set playzz to "Player"->getPos Z
"Collision wall3"->enable
"Player"->position -112634 -4119 590 90
set doOnce to 3
endif
endif

if doOnce==3
"Collision wall3"->moveworld X 200
"Collision wall3"->moveworld Y -800
if ( "Player"->getPos Z < 570 )

```

```

set doOnce to 4
set playxx to "Player"->getPos X
set playyy to "Player"->getPos Y
set playzz to "Player"->getPos Z
"Collision Wall4"->enable
"Player"->position -112126 -6150 590 90
endif
endif

if doOnce==4
"Collision wall4"->moveworld X 600
"Collision wall4"->moveworld Y -450
if ( "Player"->getPos Z < 570 )
set doOnce to 5
set playxx to "Player"->getPos X
set playyy to "Player"->getPos Y
set playzz to "Player"->getPos Z
endif
endif

if doOnce==5
stopscript ELDQ_visualforbattle
endif
end ELDQ_visualforbattle

```

Troubleshooting

General hints

- A good way to debug is to insert MessageBox commands at key points in your script.
- If you produce error messages and can't find the cause, try to remove suspect lines of codes one by one (using ; to mark them as comments) to pinpoint the error causing line.
- Pay attention to the error codes that the editor and the game give you, they usually let you pinpoint the source of the problem to some degree

The Console

Using the Console to check variables:

In the game you can use the console to check on the state of variables. Bring up the console window (standard key is ~ or whatever key is left of the "1" key if you are using a non-US keyboard layout) and type "sv" – this lists all the global variables with their values. Now find an object that has a script running on it. Bring up the console again, left click on the object – the console window title will change. Type "sv" again – now the local variables for the script running on this object will be listed.

Using the Console to quickly test scripts:

The console can help you test your scripts. For objects you don't need to place them with the editor and then go there. Just write down the ID for your object, load any savegame and then, in the console type

```
PlaceatPC "My_Object" 1,1,1
```

This will drop the object directly at your feet.

```
Player -> AddItem "My_Object", 1
```

This will drop the item into your inventory

To go to a specific location use

```
coc "cell_name"
```

for interior cells or

```
coe -1,-7
```

for exterior cells (write down the cell coordinates in the editor) coc works for exterior cells too but since most exterior cells have non-unique names you may not land quite where you

want. It can still be useful though, for example `coc balmora` will take you somewhere in Balmora.

`tcl`

Toggles collision – float through walls, visit difficult to reach places easily.

`Tgm`

Toggle God mode – test without worrying about some monster killing you.

Also, the invaluable console commands to run the game in "debug" mode allow one to view hard numbers about spell effects, chances to hit, etc (thanks to Wakim). Some of these are:

- 1) press the "~" key to call up the console.
- 2) type "tcs" into the console and hit return.
- 3) click (right or left click, can't recall) anywhere on the screen outside of the console box to continue game time while leaving the box displayed and active.

That's it. There are other variants on the "tcs" command, such as (but not limited to) "tks" and "tms" which all stand for "toggle xxxxxx statistics" where xxxxx is: c = combat, m = magic, k = kill, or what have you.

Error messages, malfunctions and common causes

In the game, when script executes:

In game error messages:

"EXPRESSION in script..."

Followed by

"RightEval ..."

This error indicates that Variables are not declared. Most often this happens with game variables that are almost used like functions, e.g. `PCEquip`.

EXPRESSION in general appears mostly when variables have not been declared in the script.

"LeftEval": this error seems to come up when you have accidentally declared a Function as a variable. The following lines e.g. would produce this error:

```
short ScriptRunning
if (ScriptRunning "MyScript" == 1 )
```

Both of the above can also be caused by not having spaces at the proper places. Always leave a space between parantheses and function calls, variables etc.

If (`OnActivate == 1`), not if (`OnActivate==1`). This only causes errors very rarely, but it sometimes does, trust me.

Infix to Postfix error

Usually indicates bad syntax. Can be caused by bad set commands using the "fix" arrow:

```
set somevar to actorID->GetHealth
should be changed to:
set somevar to ( actorID->GetHealth )
```

an alternative is again a forgotten variable declaration of variable type functions, e.g. of `OnPCEquip`, etc. (Thanks Horatio and Ragnar_GD)

AITravel command does not work

A common cause is coordinates erroneously set too far away (e.g. by omitting a "-" or having a digit too much in there). I once had a very nasty one of having typed two "-" which looked like one just slightly longer than usual "-" in the editor.

Doubled NPCs

Doubling of NPCs and other objects seems to happen occasionally when a new version of a plugin overwrites an old one, and the savegame info is also kept, leading to a clone of the NPC, even if you only changed his script. Try to load the plugin in a cell away from the NPC. Seems to be avoided by keeping the mod filename identical, apparently savegames save filename information with NPCs in savegames (thanks Raptormeat). For much more information on doubling and what to do to avoid it go to Shadowsongs excellent webpage on this issue: <http://www.angelfire.com/clone2/shadowsong/index.html>

Crash to desktop when executing the script

There are unfortunately a great number of possible causes. Many are connected with not having "do once" conditions (e.g. calling certain functions every frame). Other known problems: Removing objects with running scripts from within their own script. Using Equip on anything but potions (fixed with Tribunal). Casting targeted spells from an activator (fixed with Tribunal). Using AIActivate on teleport doors that lead outside of the active cell. Trying to use PlaceItem with the same Object ID the script is running on. Using SetDelete on a non-disabled object.

Crash to Desktop (CTD) upon loading the plugin

One reported reason is overlong calculations: there appears to be an issue with very long additions (e.g. adding up more than 20 variables in one line of code) that causes a mod to CTD upon loading. If this happens, split the calculations to several lines.

In the editor

The editor usually indicates the line in which the problem was encountered, so often error reported here are relatively easy to fix.

"Mismatched If/else/endif starting on line..."

one or several if statements are not closed. Use tabs to facilitate finding missing endifs. This also happens when wrong names are used for variables functions or other typing errors.

"Function reference object "Foobject" not found"

This error indicates that the object in question e.g.

`Foobject -> GetDistance Player`

Does not exist. When you write a script and then create a new object to be thus referenced you will also get this error. Close the editor window and reopen it to register the object with the compiler.

"Could not find variable or function "Foobject""

Another syntax error, indicating you have used an undefined variable, function or a non-existent object.

Appendix

Undocumented functions

These functions were not documented in the original helpfile. I have explained most of them in this guide, but for a few the uses are still unknown to me. I thought it might be useful to have as a list for an easy overview. The list was put together by someone who Hex-edited the TES-CS .exe, so it's likely to be complete.
(thanks to Soralis for the list and XPCagey and others who helped to put it together):

PayFineThief
EnableStatReviewMenu
GetFactionReaction
ShowMap
EnableBirthMenu
EnableClassMenu
EnableRaceMenu
EnableNameMenu
RemoveEffects
EnableMagicMenu
EnableMapMenu
EnableInventoryMenu
EnableStatsMenu
GetInterior
GetLineOfSight (alias for GetLOS?)
GetWindSpeed

GetCurrentTime
OutputObjCounts
OutputRefCounts
ResetActors
OutputRefInfo
Help
ToggleScripts
ShowAnim
PurgeTextures
BetaComment
ToggleWater
MOTO
MoveOneToOne
MenuTest
ToggleScriptOutput
ToggleLights

As well as some others by XP-Cagey, functions which are **spelled differently than documented**:

getHealthRatio -> getHealthGetRatio
getInvisible -> getInvisibile
setInvisible -> setInvisibile
modInvisible -> modInvisibile
getSecundusPhase -> getSecundaPhase
(actual syntax on the right)

And those which don't work:

getPlayerViewSwitch -> (can't be used)
OnRepair -> not in string tables
UsedOnMe -> not in string tables
name -> not in string tables

Game units:

1 game unit = 0.56 inches
50 = 28 inches
500 = 23.3 feet
5000 = 233.3 feet
8192 = 385 feet = 1 game cell

1 game unit = 1.42 cm
100 game units = 142 cm = 1.42 meters
1000 game units = 14.2 meters
8192 game units = 116.33 meters = 1 exterior cell

The island of Morrowind itself is 5.00 km north to south and 4.65 km east to west.

(thanks to Iudas for this information)

Magic Effect List

For the function GetEffect, you need the ID-string itself (e.g. GetEffect sEffectWaterBreathing). For the RemoveEffects function, the effect number is used (e.g. RemoveEffects, 0)

```
0 => sEffectWaterBreathing
1 => sEffectSwiftSwim
2 => sEffectWaterWalking
3 => sEffectShield
4 => sEffectFireShield
5 => sEffectLightningShield
6 => sEffectFrostShield
7 => sEffectBurden
8 => sEffectFeather
9 => sEffectJump
10 => sEffectLevitate
11 => sEffectSlowFall
12 => sEffectLock
13 => sEffectOpen
14 => sEffectFireDamage
15 => sEffectShockDamage
16 => sEffectFrostDamage
17 => sEffectDrainAttribute
18 => sEffectDrainHealth
19 => sEffectDrainSpellpoints
20 => sEffectDrainFatigue
21 => sEffectDrainSkill
22 => sEffectDamageAttribute
23 => sEffectDamageHealth
24 => sEffectDamageMagicka
25 => sEffectDamageFatigue
26 => sEffectDamageSkill
27 => sEffectPoison
28 => sEffectWeaknessToFire
29 => sEffectWeaknessToFrost
30 => sEffectWeaknessToShock
31 => sEffectWeaknessToMagicka
32 => sEffectWeaknessToCommonDisease
33 => sEffectWeaknessToBlightDisease
34 => sEffectWeaknessToCorprusDisease
35 => sEffectWeaknessToPoison
36 => sEffectWeaknessToNormalWeapons
37 => sEffectDisintegrateWeapon
38 => sEffectDisintegrateArmor
39 => sEffectInvisibility
40 => sEffectChameleon
41 => sEffectLight
42 => sEffectSanctuary
43 => sEffectNightEye
44 => sEffectCharm
45 => sEffectParalyze
46 => sEffectSilence
47 => sEffectBlind
48 => sEffectSound
49 => sEffectCalmHumanoid
50 => sEffectCalmCreature
51 => sEffectFrenzyHumanoid
52 => sEffectFrenzyCreature
53 => sEffectDemoralizeHumanoid
54 => sEffectDemoralizeCreature
55 => sEffectRallyHumanoid
56 => sEffectRallyCreature
57 => sEffectDispel
58 => sEffectSoultrap
59 => sEffectTelekinesis
60 => sEffectMark
61 => sEffectRecall
62 => sEffectDivineIntervention
63 => sEffectAlmsiviIntervention
64 => sEffectDetectAnimal
65 => sEffectDetectEnchantment
66 => sEffectDetectKey
67 => sEffectSpellAbsorption
68 => sEffectReflect
69 => sEffectCureCommonDisease
70 => sEffectCureBlightDisease
71 => sEffectCureCorprusDisease
72 => sEffectCurePoison
73 => sEffectCureParalyzation
74 => sEffectRestoreAttribute
75 => sEffectRestoreHealth
76 => sEffectRestoreSpellPoints
77 => sEffectRestoreFatigue
78 => sEffectRestoreSkill
79 => sEffectFortifyAttribute
80 => sEffectFortifyHealth
81 => sEffectFortifySpellpoints
82 => sEffectFortifyFatigue
83 => sEffectFortifySkill
84 => sEffectFortifyMagickaMultiplier
85 => sEffectAbsorbAttribute
86 => sEffectAbsorbHealth
87 => sEffectAbsorbSpellPoints
88 => sEffectAbsorbFatigue
89 => sEffectAbsorbSkill
90 => sEffectResistFire
91 => sEffectResistFrost
92 => sEffectResistShock
93 => sEffectResistMagicka
94 => sEffectResistCommonDisease
95 => sEffectResistBlightDisease
96 => sEffectResistCorprusDisease
97 => sEffectResistPoison
98 => sEffectResistNormalWeapons
99 => sEffectResistParalysis
100 => sEffectRemoveCurse
101 => sEffectTurnUndead
102 => sEffectSummonScamp
103 => sEffectSummonClannfear
104 => sEffectSummonDaedroth
105 => sEffectSummonDremora
106 => sEffectSummonAncestralGhost
107 => sEffectSummonSkeletalMinion
108 => sEffectSummonLeastBonewalker
109 => sEffectSummonGreaterBonewalker
110 => sEffectSummonBonelord
111 => sEffectSummonWingedTwilight
112 => sEffectSummonHunger
113 => sEffectSummonGoldensaint
114 => sEffectSummonFlameAtronach
115 => sEffectSummonFrostAtronach
116 => sEffectSummonStormAtronach
117 => sEffectFortifyAttackBonus
118 => sEffectCommandCreatures
119 => sEffectCommandHumanoids
120 => sEffectBoundDagger
121 => sEffectBoundLongsword
122 => sEffectBoundMace
123 => sEffectBoundBattleAxe
124 => sEffectBoundSpear
125 => sEffectBoundLongbow
126 => sEffectExtraSpell
127 => sEffectBoundCuirass
128 => sEffectBoundHelm
129 => sEffectBoundBoots
130 => sEffectBoundShield
131 => sEffectBoundGloves
132 => sEffectCorpus
133 => sEffectVampirism
134 => sEffectSummonCenturionSphere
135 => sEffectSunDamage
136 => sEffectStuntedMagicka
```

List of console commands

Console (in game only commands). Some of these CAN be used in scripts, by the way!

CenterOnCell (coc), CellID

Places the PC in the named cell. Very useful for testing mods.

CenterOnExterior (coe), X, Y

Places the PC in the exterior cell grid.

CreateMaps "Filename.esp"

Creates map image file for Xbox.

According to the UESP: Creates map image file depending on the Create Maps Enable value in the Morrowind.INI file. If it is 1 (XBox), the file FILENAME.ESP.MAP is created in the DataFiles path with the map data (unknown format). If the value is 2 (Exterior Cell Maps) and you have created a directory Maps in the main Morrowind game directory, this command will create a 256x256 high color bitmap of each exterior cell in the game. This command takes a long while even on fast computers as each cell in the game is loaded.

FillJournal

add all entries to journal, takes a long time

FillMap

show all the towns on the full map.

FixMe

Jump 128 units away from where I am now. Good to get "unstuck"

GetFactionReaction factionID factionID

The faction ids are not optional, works in Console window only

Help

Shows shorthand for most commands

MoveOneToOne (moto)

Unknown. Seems to affect the animation speed, but the true purpose remains unclear

Show

ShowVars (sv)

Lists global variables and variables in global scripts, or local variables if you click on an object with a local script first.

StopCellTest (sct)

TestCells (tc)

TestInteriorCells (tic)

TestModels (t3d)

ToggleAI (ta)

ToggleBorders (tb)

Shows borders of exterior cells

ToggleCombatStats (tcs)

Allows you to monitor combat statistics in realtime. Enable this, the rightclick outside the console window to continue playing with the console window open.

ToggleCollision (tcl)

Turns collision on and off. Lets you float through walls. Can make actors drop through floors, too.

ToggleCollisionBoxes (tcb)

ToggleCollisionGrid (tcg)

ToggleDebugText (tdt)

ToggleDialogueStats (tds)

ToggleFogOfWar (tfow)

Lets you see all of the local map

ToggleFullHelp (tfh)

shows you ownership and script on mouseover while in console mode.

name

ToggleGodMode (tgm)

Makes you unhurttable

ToggleGrid (tg)

ToggleKillStats (tks)

ToggleLoadFade

ToggleMagicStats (tms)

ToggleMenus TM

ToggleScripts

ToggleStats (tst)

ToggleSky (ts)

ToggleTextureString (tts)

ToggleWorld (tw)

ToggleWireframe (twf)

Shows grid instead of full render

TPG

Toggle AI path grid display.

SG

Show selected actor's group members

ST

Show selected actor's target group members.

ShowScenegraph (ssg)

Variable-type functions:

The following is a list of all those variable/function type hybrids that you need to declare as variables if you use them in a script.

Local variables that get set by the game:

```
Short OnPCEquip
Short OnPCAdd
Short OnPCRepair
Short OnPCSoulGemUse
Short OnPCHitMe
Float minimumProfit (Tribunal)
```

Local variables that you can set as a flag:

```
Short Companion (Tribunal)
Short PCSkipEquip
```

Special Globals

Some globals hold special significance that you can take advantage of for your own scripting. Since they are globals you do not need to declare them.

Short NPCVoiceDistance (750)	Used as a distance when Following NPCs call after you to wait for them (e.g. see DandsaScript)
Float GameHour	Holds the current hour of the day (0-23)
Short Day	Holds the current day of the month (1-30)
Short Month	Holds the current Month of the year (0-11)
Short Year (427)	Holds the current year
Float TimeScale (30)	Sets the ratio of real-time/game-time
Short Random100	Is randomly set between 0-100 (set by main script)
Short PCRace	Contains the players Race (1=Agonian, 2=Breton, 3=Dark Elf, 4=High Elf, 5=Imperial, 6= Khajiit, 7=Nord, 8=Orc, 9=Redguard, 10=Woodelf)
Short PCVampire	Vampire status: 0=Normal, 1=Vampire, -1= cured
Short VampClan	If the PC becomes a vampire, this indicates his clan. 1=Aundae, 2=Berne, 3=Quarra
Short DaysPassed	Tribunal/Bloodmoon: Contains the number of days since the game started (Forum info / JOG)
Short PCWerewolf	Werewolf status: 0=Normal, 1=Werewolf, -1=Cured
Float WerewolfClawMult (25.00)	Increased during the Werewolf quests to make your claw attack more powerful.

Game Settings

The following long table is a list of game settings. Listed here are all settings that have a numerical value. Not listed are string entries – which are used to set many standard message texts, menu-texts, spell effect names, etc. But since they are fairly descriptive, they should be easy to figure out. Not so the numerical settings. Thanks to four forum members (maxpublic, Ldones, Wakim and Iudas), the meaning of many of the settings is now known and compiled into the list below. The list may still be a bit rough. I have not edited it thoroughly, but I am sure it will be interesting information for many modders.

In many cases where you see Base and Mult game settings, the formula they're involved in is a standard linear equation in the form $y = mx + b$, where m is the mult, b is the base, and x is some attribute, skill, or other value.

Note that the entry names begin with f or i (for floats and integers). The string entries begin with s (string). To my knowledge they can not be changed in-game, only in the TES-CS.

Nr.	Name	Value	Description and Comments
"465"	"fRepairMult"	1.0000	Sets the general effectiveness of the repair skill of the character, via the armorer's hammer used
"466"	"fRepairAmountMult"	3.0000	Tells the game how many points of health are returned to the item when repaired Determines cost for repairing items (Whether calculated from Max Item Health or Item Cost, I'm not sure)
"467"	"fSpellValueMult"	10.0000	
"468"	"fSpellMakingValueMult"	7.0000	
"469"	"fEnchantmentValueMult"	1000.0000	is the setting for the price you pay at an enchanter to enchant an item. Linear.
"470"	"fTravelMult"	4000.0000	Sets the cost of Silt Strider and boat travel (I think) Multiplies cost of Travel – Unsure why the number is so high, but raising it raises the cost of Fast Travel
"471"	"fTravelTimeMult"	16000.0000	Tells the game how much time elapses during this sort of travel
"472"	"fMagesGuildTravel"	10.0000	Sets the cost of Guild Guide travel
"947"	"fWortChanceValue"	15.0000	Iudas: Used to calculate whether a plant has any ingredients inside. Wakim: is compared to your alchemy skill to determine which of the effects of an ingredient you can see.
"949"	"fMinWalkSpeed"	100.0000	This is the minimum walking speed of the PC, regardless of stats, skills or encumbrance
"950"	"fMaxWalkSpeed"	200.0000	This is the maximum walking speed of the PC, regardless of stats, skills, or encumbrance The actual walking speed of NPC's (and the PC) is set by checking various factors (Speed, Athletics, etc.) and assigning a value between fMinWalkSpeed and fMaxWalkSpeed based on that – The two settings dictate the spectrum of Walk Speeds
"951"	"fMinWalkSpeedCreature"	5.0000	The same as for the PC, but if you badly encumber a creature it'll move veeerrry slowly. I've done this by accident.
"952"	"fMaxWalkSpeedCreature"	300.0000	Same as above, they get faster, so they cover the speed spectrum more rapidly
"953"	"fEncumberedMoveEffect"	0.3000	This sets how encumbrance affects walking and running speed, within the min/max limits set by other values.
"954"	"fBaseRunMultiplier"	1.7500	Exactly as it says. Changing the value will increase/decrease base running speed. Dictates how much faster Running is than the current Walk Speed
"955"	"fAthleticsRunBonus"	1.0000	Sets how Athletics affects running speed.
"956"	"fJumpAcrobaticsBase"	128.0000	Sets the base jumping distance for the PC.
"957"	"fJumpAcroMultiplier"	4.0000	Sets the multiplier for Acrobatics, which is why you can leap over tall buildings when your Acro is high enough.
"958"	"fJumpEncumbranceBase"	0.5000	Effects how greatly jumping ability is effected by Encumbrance, but I'm unsure how
"959"	"fJumpEncumbranceMultiplier"	1.0000	Effects how greatly jumping ability is effected by Encumbrance, but I'm unsure how
"960"	"fJumpRunMultiplier"	1.0000	UNSURE – Presumably effects Jump Distance while running (it doesn't seem to effect height, but I could be wrong)
"961"	"fJumpMoveBase"	0.5000	
"962"	"fJumpMoveMult"	0.5000	
"963"	"fSwimWalkBase"	0.5000	Multiplies your walking speed to achieve the swimming speed at a 'walk' Base swim speed while 'walking'
"964"	"fSwimRunBase"	0.5000	Multiplies your running speed to achieve the swimming speed at a 'run'.
"965"	"fSwimWalkAthleticsMult"	0.0200	Tells the game how Athletics affects 'walking' swimming speed. These low values keep you from flying through the water like you do on land when your Athletics is high.
"966"	"fSwimRunAthleticsMult"	0.1000	Same as above.
"967"	"fSwimHeightScale"	0.9000	Determines how close to the surface you have to be before the breathe indicator goes away
"968"	"fHoldBreathTime"	20.0000	The number of seconds your PC can hold her breath. base time that a character can remain underwater before incurring suffocation damage
"969"	"fHoldBreathEndMult"	0.5000	How Endurance affects the time you can hold your breath. I believe this is a flat-out multiplier to End, added as seconds to HoldBreathTime.<Doesn't seem to work.>
"970"	"fSuffocationDamage"	3.0000	The amount of health damage you take each second you suffocate
"971"	"fMinFlySpeed"	5.0000	Exactly as it says – minimum flying speed.
"972"	"fMaxFlySpeed"	300.0000	See above

"973"	"fStromWindSpeed"	0.7000	UNSURE - Determines altered walk speed during an ash or blight storm, but I'm unsure how – Might be a separate value from that entirely – might determine speed of storm particles /sprites(Dust, etc.) Interesting (possibly related) note, while treading water in an ash storm, I noticed I was moving slightly.
"974"	"fStromWalkMult"	0.2500	Determines altered walk speed during an ash or blight storm, but I'm unsure how uses the getwindspeed to lower the PC movement speed during storms...
"975"	"fFallDamageDistanceMin"	400.0000	The minimum distance you have to fall before you take damage. (Presumably in units) In game units each unit - .0.56 inches
"976"	"fFallDistanceBase"	0.0000	This will increase/decrease the distance needed to fall before you take damage.
"977"	"fFallDistanceMult"	0.0700	Higher you are the more damage you take when you hit
"978"	"fFallAcroBase"	0.2500	Acrobatics skill increases the distance you can fall before you take damage.
"979"	"fFallAcroMult"	0.0100	Has to do w/ how the Acrobatics skill effects Fall Distance and Damage, but unsure how
"980"	"iMaxActivateDist"	192	Maximum distance for the player to be able to 'Activate' an object - approx 9 feet
"981"	"iMaxInfoDist"	192	Maximum distance for an Info message (object/NPC/creature name, etc.) to pop up in the Player's view
"982"	"fVanityDelay"	30.0000	Seconds until VanityMode begins – the camera starts circling the player if there is no input via mouse or keyboard.
"983"	"fMaxHeadTrackDistance"	400.0000	IIRC, this is the maximum distance an NPC or creature can be away from another NPC or creature and still trigger the 'head follow' routine you sometimes see. Put your PC in Balmora, let it go to Vanity View and you'll see your PC watch passing NPCs and 'follow' their movements for a certain amount of time.
"984"	"fInteriorHeadTrackMult"	0.5000	UNSURE – something to do w/ the modifier for this in Interiors – Do they track at half-distance in Interiors?
"985"	"iHelmWeight"	5	These values are used to set what weights are used to determine whether a piece of armor is light, medium, or heavy. Altering a value alters these categories for *all* armor of that type *everywhere* in the game. Rather nice, actually; I used it in redux to set weight categories for all of my armor types across the board.
"986"	"iPauldronWeight"	10	
"987"	"iCuirassWeight"	30	
"988"	"iGauntletWeight"	5	
"989"	"iGreavesWeight"	15	
"990"	"iBootsWeight"	20	
"991"	"iShieldWeight"	15	
"992"	"fLightMaxMod"	0.6000	These values are used in conjunction with armor weights to set the weight classes (Light, Medium, Heavy).
"993"	"fMedMaxMod"	0.9000	
"994"	"fUnarmoredBase1"	0.1000	These two values dictate the range of AR for characters going Unarmored, based on the Unarmored skill – As they are, the settings produce a Maximum Unarmored AR of 65 (at 100 Unarmored skill), which you can see is some for of multiplication between the two settings – Reversing the values produces the same effect , so it seems the values are interchangeable (unless I missed something – could be wrong) - Changing one to 1.000 and the other to 0.0650 results in a max Unarmored AR of 650 – The game multiplies the numbers and then multiplies the resulting number by 1000 to obtain the actual in-game Max Unarmored AR – Doesn't seem to effect Min AR independently, only max - Progression of AR (from low skill to high skill) appears to be hard-coded. It has also been found that the Unarmored skill doesn't work at all UNLESS atleast one item of armor is worn. (Forum Info / The other Felix)
"995"	"fUnarmoredBase2"	0.0650	
"996"	"iBaseArmorSkill"	30	The Skill Level where in-game armors reach their base (i.e. 'In-Editor') AR value – Example: Glass Armor has a Base AR of 50 – At Light Armor skill level 30 (as indicated above), it will read as having AR 50 in-game – Before Skill Level 30, armors have diminished AR's from their Base Value, and after Skill Level 30, armors have higher AR's than their base until Skill Level reaches 100 – I haven't figured out the game's scheme for determining the mult value yet
"997"	"fBlockStillBonus"	1.2500	UNSURE – Presumably the amount that standing still increases the chance to block
"998"	"fDamageStrengthBase"	0.5000	Your STR adds to the damage you do with weapons. This determines how much damage is added.
"999"	"fDamageStrengthMult"	0.1000	Effects amount that Strength effects damage dealt in combat (Unsure of how this value relates to in-game effect)
"1000"	"fSwingBlockBase"	1.0000	
"1001"	"fSwingBlockMult"	1.0000	
"1002"	"fFatigueBase"	1.2500	How much fatigue you lose while walking. However, this appears to let you jump very high without getting hurt (Bug? – Forum Info / DinkumThinkum). 1002 – 1022 All effect 'Fatigue' in-game, obviously – For separate actions, although not all of them actually have an effect in-game (I've never been able to get spells to reduce fatigue) – They seem pretty self-explanatory, but I haven't tested them thoroughly
"1003"	"fFatigueMult"	0.5000	Used to determine successful chance of casting if you are fatigued
"1004"	"fFatigueReturnBase"	2.5000	How much fatigue you regain per second. This is why you don't actually fatigue while walking.
"1005"	"fFatigueReturnMult"	0.0200	How much fatigue returns per second while walking
"1006"	"fEndFatigueMult"	0.0400	
"1007"	"fFatigueAttackBase"	2.0000	How much fatigue you lose with every melee attack you make
"1008"	"fFatigueAttackMult"	0.0000	
"1009"	"fWeaponFatigueMult"	0.2500	
"1010"	"fFatigueBlockBase"	4.0000	How much fatigue you lose blocking with a shield.
"1011"	"fFatigueBlockMult"	0.0000	This will increase the amount of fatigue lost when blocking with a shield.
"1012"	"fWeaponFatigueBlockMult"	1.0000	

"1013"	"fFatigueRunBase"	5.0000	How much fatigue you lose running.
"1014"	"fFatigueRunMult"	2.0000	This one appears to work with encumbrance,the more encumbered the more fatigue you lose/second
"1015"	"fFatigueJumpBase"	5.0000	How much fatigue you lose jumping.
"1016"	"fFatigueJumpMult"	0.0000	Modifier for fatigue loss
"1017"	"fFatigueSwimWalkBase"	2.5000	How much fatigue you lose swimming at a 'walk'
"1018"	"fFatigueSwimRunBase"	7.0000	How much fatigue you lose swimming at a 'run'.
"1019"	"fFatigueSwimWalkMult"	0.0000	Modifier for fatigue loss
"1020"	"fFatigueSwimRunMult"	0.0000	Modifier for fatigue loss
"1021"	"fFatigueSneakBase"	1.5000	The base level of fatigue loss while sneaking
"1022"	"fFatigueSneakMult"	1.5000	Multiplier to that base level
"1023"	"fMinHandToHandMult"	0.1000	
"1024"	"fMaxHandToHandMult"	0.5000	
"1025"	"fHandtoHandHealthPer"	0.1000	
"1026"	"fCombatInvisoMult"	0.2000	Reduce the chance to hit the PC when he is chameleoned or invisible
"1027"	"fCombatKODamageMult"	1.5000	
"1028"	"fCombatCriticalStrikeMult"	4.0000	This one appears to only work if you hit someone unawares while sneaking. I never got it to do anything else. 4x damage from a successful sneak attack works when chameleoned or invisible
"1029"	"iBlockMinChance"	10	Minimum chance of blocking with a shield
"1030"	"iBlockMaxChance"	50	Maximum chance of blocking with a shield
"1031"	"fLevelUpHealthEndMult"	0.1000	Multiplies current END to get hit points added at level-up.
"1032"	"fSoulGemMult"	3.0000	A soul gem's monetary value is multiplied by this value to determine the soul capacity of a soul gem. Creatures with a soul value less than or equal to that capacity can "fit" in the gem.
"1033"	"fEffectCostMult"	0.5000	The setting for all magicka costs for all spell effects. Changing this will change what all spells and enchantments cost. Everything. Linear change. Doubling this makes all spells cost twice as much magicka, all enchanted items cost twice as many charges.
"1034"	"fSpellPriceMult"	2.0000	
"1035"	"fFatigueSpellBase"	0.0000	
"1036"	"fFatigueSpellMult"	0.0000	
"1037"	"fFatigueSpellCostMult"	0.0000	
"1038"	"fPotionStrengthMult"	0.5000	
"1039"	"fPotionT1MagMult"	1.5000	
"1040"	"fPotionT1DurMult"	0.5000	
"1041"	"fPotionMinUsefulDuration"	20.0000	
"1042"	"fPotionT4BaseStrengthMult"	20.0000	
"1043"	"fPotionT4EquipStrengthMult"	12.0000	
"1044"	"fIngredientMult"	1.0000	Min # of an ingredient required to make a potion
"1045"	"fMagicItemCostMult"	1.0000	UNUSED
"1046"	"fMagicItemPriceMult"	1.0000	
"1047"	"fMagicItemOnceMult"	1.0000	
"1048"	"fMagicItemUsedMult"	1.0000	
"1049"	"fMagicItemStrikeMult"	1.0000	
"1050"	"fMagicItemConstantMult"	1.0000	
"1051"	"fEnchantmentMult"	0.1000	is the setting for how much enchantment an item can hold based upon the value set in each individual item's property file. Linear, if an item in TESCS shows an enchantment value of 1200 (i.e. an exquisite ring) multiply it by fEnchantmentMult to get the actual enchantment you'll see in the make an enchanted item window.
"1052"	"fEnchantmentChanceMult"	3.0000	These affect the PC's chance of making an enchantment
"1053"	"fPCBaseMagickaMult"	1.0000	This sets the spell point multiplier for the PC with respect to INT (e.g., 1 x INT with this setting)
"1054"	"fNPCBaseMagickaMult"	2.0000	This does the same thing for NPCs.
"1055"	"fAutoSpellChance"	80.0000	
"1056"	"fAutoPCSpellChance"	50.0000	
"1057"	"iAutoSpellTimesCanCast"	3	
"1058"	"iAutoSpellAttSkillMin"	70	
"1059"	"iAutoSpellAlterationMax"	5	
"1060"	"iAutoSpellConjurationMax"	2	
"1061"	"iAutoSpellDestructionMax"	5	
"1062"	"iAutoSpellIllusionMax"	5	
"1063"	"iAutoSpellMysticismMax"	5	
"1064"	"iAutoSpellRestorationMax"	5	
"1065"	"iAutoPCSpellMax"	100	
"1066"	"iAutoRepFacMod"	2	A positive modification to relations you get with people who belong to the same faction
"1067"	"iAutoRepLevMod"	0	You can apparently add rep points with each level-up. I've never tried it.
"1068"	"iMagicItemChargeOnce"	1	1068-1071 effect the amount of charges auto-calculated on magic items based on their function – This value is the number of uses that the game will account for when calculating the max charges of a magic item (works universally, across the
"1069"	"iMagicItemChargeConst"	10	
"1070"	"iMagicItemChargeUse"	5	

"1071"	"iMagicItemChargeStrike"	10	board with all –ingame items – 1068 is the setting for the number of charges an automatically calculated cast once effect enchanted item will have. Formula is BaseSpellEffectCost x iMagicItemChargeOnce. Linear. This way an item with a cast once effect will have exactly the number of charges needed to cast the effect upon it 1069 for const effect items 1070 is the setting for the multiplier for charges for automatically calculated cast when used effect enchanted items. See above for explanation. 1071 for "Cast on Strike" items (charges are calculated to account for X 'uses' with this value as is)
"1072"	"iMonthsToRespawn"	4	The time to respawn things like the Fighters Guild/Mages Guild chests, etc. How many months before a picked plant respawns ingredients. Chests in guilds respawn contents the same as any other chests.
"1073"	"fCorpseClearDelay"	72.0000	How many hours it takes before a non-persistent corpse disappears.
"1074"	"fCorpseRespawnDelay"	72.0000	How many hours it takes before a respawnable creature actually respawns (note that this doesn't seem to work properly).
"1075"	"fBarterGoldResetDelay"	24.0000	How many hours it takes before a trader resets it barter gold to its default value.
"1076"	"fEncumbranceStrMult"	5.0000	A straight multiplier to STR to see how much a PC/NPC/creature can carry.
"1077"	"fPickLockMult"	-1.0000	Dictates amount that Lock pick difficulty raises according to Lock Level – Lower the value here, the harder it gets – Positive values make locks get easier with higher lock Levels
"1078"	"fTrapCostMult"	0.0000	Dictates difficulty of traps based on the spell cost of the spell assigned as trap – Lower the value here, the harder it gets to disarm (again, based on spell cost of assigned 'trap') The values is multiplied by the spell cost of a trap and then added to your chance of disarming it. Since it's set to zero, the trap spell's cost is not incorporated into the chance. So basically it's also unused.
"1079"	"fMessageTimePerChar"	0.1000	
"1080"	"fMagicItemRechargePerSecond"	0.0500	This is the setting for the amount of charges restored to a charged magic item per second of game play. Linear. 0.05 x 20 seconds = 1 charge restored.
"1081"	"i1stPersonSneakDelta"	10	
"1082"	"iBarterSuccessDisposition"	1	If you barter with a merchant successfully, your disposition with that merchant increases by one and falls by 1 if you fail a barter attempt.
"1083"	"iBarterFailDisposition"	-1	
"1084"	"iLevelupTotal"	10	How many skill points you need before you level up.
"1085"	"iLevelupMajorMult"	1	How much each major skill is worth in points. E.g., if you set this to 2 then each point earned in a major skill counts as 2 skill points for leveling up.
"1086"	"iLevelupMinorMult"	1	Same as above, but for minor skills.
"1087"	"iLevelupMajorMultAttribute"	1	I *think* - not sure if I remember this correctly - but I think this works like the above, but for skills governed by your two primary attributes. So if your primary attributes are STR and AGI, and you set 1087 to 2, then any major skill governed by one of these attributes which goes up by a point counts as 2 points for purposes of leveling.
"1088"	"iLevelupMinorMultAttribute"	1	
"1089"	"iLevelupMiscMultAttribute"	1	
"1090"	"iLevelupSpecialization"	1	
"1091"	"iLevelUp01Mult"	2	The game keeps track of how many skill points you've gained since the last level up. If you gained 8 skill points in skills governed by AGI, then when you get to distribute attribute points whatever number is in place for iLevelUp08Mult will be used for AGI. So if this value is 4, you'll see a x 4 next to AGI when you level up (you'll get 4 points in AGI if you pick this during the leveling process).
"1092"	"iLevelUp02Mult"	2	
"1093"	"iLevelUp03Mult"	2	
"1094"	"iLevelUp04Mult"	2	
"1095"	"iLevelUp05Mult"	3	
"1096"	"iLevelUp06Mult"	3	
"1097"	"iLevelUp07Mult"	3	
"1098"	"iLevelUp08Mult"	4	
"1099"	"iLevelUp09Mult"	4	
"1100"	"iLevelUp10Mult"	5	
"1101"	"iSoulAmountForConstantEffect"	400	This is the setting for the minimum soul value to toggle the constant effect button in the enchantment creation window.
"1102"	"fConstantEffectMult"	15.0000	UNUSED
"1103"	"fEnchantmentConstantDurationMult"	100.0000	This setting is the multiplier for constant effect cast cost as compared to a 0 duration spell. so restore health 2-2 for 0 secs, which costs 0.50 to cast as a spell, costs 0.5 x 100 = 50 as a constant effect.
"1104"	"fEnchantmentConstantChanceMult"	0.5000	
"1105"	"fWeaponDamageMult"	0.1000	weapon damage during combat. depreciation as it were.
"1106"	"fSeriousWoundMult"	0.0000	UNUSED
"1107"	"fKnockDownMult"	0.5000	This sets the chance for a knock-down factored on how much damage you do in a single blow.
"1108"	"iKnockDownOddsBase"	50	Sets the base odds for a knockdown when the condition for it is met
"1109"	"iKnockDownOddsMult"	50	
"1110"	"fCombatArmorMinMult"	0.2500	
"1111"	"fHandToHandReach"	1.0000	Sets the reach of HTH weapons. Values of less than 1.0 have no meaning.
"1112"	"fVoiceIdleOdds"	10.0000	Controls likelihood of an NPC 'speaking' a voice clip when Idle (Unsure of specifics)
"1113"	"iVoiceAttackOdds"	10	Controls likelihood of an NPC 'speaking' a voice clip when attacking (Unsure of specifics)
"1114"	"iVoiceHitOdds"	30	Controls likelihood of an NPC 'speaking' a voice clip when being hit (Unsure of specifics)

"1115"	"fProjectileMinSpeed"	400.0000	Sets the minimum speed of projectile weapons
"1116"	"fProjectileMaxSpeed"	3000.0000	Dictates maximum speed of projectiles from bows and crossbows
"1117"	"fThrownWeaponMinSpeed"	300.0000	Sets the minimum speed of thrown weapons
"1118"	"fThrownWeaponMaxSpeed"	1000.0000	Dictates Max speed of thrown weapons
"1119"	"fTargetSpellMaxSpeed"	1000.0000	Sets the speed of spells. Double this and your spells will *zip* across the screen! – Min speed is apparently hard-coded
"1120"	"fProjectileThrownStoreChance"	25.0000	The odds of getting arrows back when you loot a corpse. Thrown weapons also
"1121"	"iPickMinChance"	5	UNSURE – Don't know if this is with pick pocketing or lock picking
"1122"	"iPickMaxChance"	75	UNSURE – Don't know if this is with pick pocketing or lock picking
"1123"	"fDispRaceMod"	5.0000	You have better relations with your own race than with others.
"1124"	"fDispPersonalityMult"	0.5000	These determine how personality affect NPC disposition
"1125"	"fDispPersonalityBase"	50.0000	
"1126"	"fDispFactionMod"	3.0000	These determine how your rank in a faction will alter your relations with people that belong to that faction. This is why when you reach high ranks in a faction everyone in that faction suddenly becomes very friendly.
"1127"	"fDispFactionRankBase"	1.0000	
"1128"	"fDispFactionRankMult"	0.5000	
"1129"	"fDispCrimeMod"	0.0000	This is multiplied by the player's crime level (bounty) to determine how that information affects an NPC's disposition towards the player.
"1130"	"fDispDiseaseMod"	-10.0000	How much disposition is lowered when you're suffering from a disease.
"1131"	"iDispAttackMod"	-50	Not completely sure – NPC disposition modifier if PC attacks said NPC
"1132"	"fDispWeaponDrawn"	-5.0000	How much disposition is lowered when you have a weapon drawn.
"1133"	"fDispBargainSuccessMod"	1.0000	I don't remember if these work the same as the previous barter disposition values, or if these are multipliers. These effect the long term disposition of the merchant
"1134"	"fDispBargainFailMod"	-1.0000	
"1135"	"fDispPickPocketMod"	-25.0000	NPC disposition modifier for catching the PC attempting to pickpocket them
"1136"	"iDaysinPrisonMod"	100	determines prison time based on your crime level.
"1137"	"fDispAttacking"	-10.0000	Unsure – I believe it's an NPC Disposition modifier if the PC is attacking something other than the NPC it effects non-combatants disposition.
"1138"	"fDispStealing"	-0.5000	Unsure - I believe it's an NPC Disposition modifier if the PC is stealing from someone other than the NPC it effects
"1139"	"iDispTresspass"	-20	NPC Disposition modifier for catching the PC 'trespassing' – Not sure what that exactly means in-game
"1140"	"iDispKilling"	-50	Unsure NPC Disposition modifier for witnessing the PC kill an innocent NPC (I think...)
"1141"	"iTrainingMod"	10	determines training costs. The higher the value, the more training costs. – unsure of method of calculation
"1142"	"iAlchemyMod"	2	
"1143"	"fBargainOfferBase"	50.0000	This is multiplied by the item's value to determine what the merchant will offer when selling. Base value is also modified by PC level. Base amount that merchants will buy items from you for, in percentage points – Believe it goes both ways, but I'm unsure how it would work the 'other way'
"1144"	"fBargainOfferMulti"	-4.0000	Effects how much the merchant lowers his offers during a bargaining session
"1145"	"fDispositionMod"	1.0000	
"1146"	"fPersonalityMod"	5.0000	
"1147"	"fLuckMod"	10.0000	IIRC, this is multiplied by your Luck as a percentage to get a base increase to all skills.
"1148"	"fReputationMod"	1.0000	
"1149"	"fLevelMod"	5.0000	
"1150"	"fBribe10Mod"	35.0000	Dictates amount that NPC Disposition will raise on a successful 10 Gold Bribe – Don't believe it's in straight disposition points – Could be percentages - (Other factors like race, sex, opposing faction etc. reduce this amount significantly)
"1151"	"fBribe100Mod"	75.0000	Dictates amount that NPC Disposition will raise on a successful 100 Gold Bribe. See above.
"1152"	"fBribe1000Mod"	150.0000	Dictates amount that NPC Disposition will raise on a successful 1000 Gold Bribe. See above.
"1153"	"fPerDieRollMult"	0.3000	
"1154"	"fPerTempMult"	1.0000	is used in just about every disposition modifying calculation.
"1155"	"iPerMinChance"	5	
"1156"	"iPerMinChange"	10	
"1157"	"fSpecialSkillBonus"	0.8000	These all determine how fast you gain skill points in each skill. The lower the value, the faster you'll gain skill points. The values are multiplied by whatever rate you set for each individual skill.
"1158"	"fMajorSkillBonus"	0.7500	
"1159"	"fMinorSkillBonus"	1.0000	
"1160"	"fMiscSkillBonus"	1.2500	
"1161"	"iAlarmKilling"	90	
"1162"	"iAlarmAttack"	50	
"1163"	"iAlarmStealing"	1	
"1164"	"iAlarmPickPocket"	20	
"1165"	"iAlarmTresspass"	5	
"1166"	"fAlarmRadius"	2000.0000	When an NPC raises the alarm, this is the base radius for response by other affiliated NPCs.
"1167"	"iCrimeKilling"	1000	These set the gold value for crimes. I believe that fCrimeStealing is multiplied by the price of the item stolen.
"1168"	"iCrimeAttack"	40	
"1169"	"fCrimeStealing"	1.0000	
"1170"	"iCrimePickPocket"	25	
"1171"	"iCrimeTresspass"	5	
"1172"	"iCrimeThreshold"	1000	When NPC's start to react negatively to the PC

"1173"	"iCrimeThresholdMultiplier"	10	
"1174"	"fCrimeGoldDiscountMult"	0.5000	Thieves guild discount when you have a price on your head.
"1175"	"fCrimeGoldTurnInMult"	0.9000	Discount on the fine if you turn yourself in.
"1176"	"iFightAttack"	100	
"1177"	"iFightAttacking"	50	
"1178"	"iFightDistanceBase"	20	
"1179"	"fFightDistanceMultiplier"	0.0050	
"1180"	"iFightAlarmMult"	1	
"1181"	"fFightDispMult"	0.2000	
"1182"	"fFightStealing"	50.0000	
"1183"	"iFightPickpocket"	25	
"1184"	"iFightTrespass"	25	
"1185"	"iFightKilling"	50	
"1186"	"iFlee"	0	UNUSED
"1187"	"iGreetDistanceMultiplier"	6	Used for those annoying voice greetings NPCs use when you get too close Specifically (if the contrustion set help is to be believed) this is multiplied by their hello rating to get the distance before they talk.
"1188"	"iGreetDuration"	4	
"1189"	"fGreetDistanceReset"	512.0000	How far away from an NPC you have to get before they check for a voice greeting again.
"1190"	"fIdleChanceMultiplier"	0.7500	Probability multiplier that an NPC will mumble something while standing idly near the PC
"1191"	"fSneakUseDist"	500.0000	Helps determine if you can sneak
"1192"	"fSneakUseDelay"	1.0000	Helps determine how long before the Sneak Icon come on
"1193"	"fSneakDistanceBase"	0.5000	see above
"1194"	"fSneakDistanceMultiplier"	0.0020	see above
"1195"	"fSneakSpeedMultiplier"	0.7500	Multiplied by base walking speed to see how fast you move while sneaking.
"1196"	"fSneakViewMult"	1.5000	Makes it more difficult to sneak when in view of an NPC.
"1197"	"fSneakNoViewMult"	0.5000	Makes it easier to sneak when you aren't in view.
"1198"	"fSneakSkillMult"	1.0000	
"1199"	"fSneakBootMult"	-1.0000	Multiplied by the boot value (weight?) to determine the reduction to Sneak skill.
"1200"	"fCombatDistance"	128.0000	Combined with weapon reach, determines the effective distance that hits can be obtained
"1201"	"fCombatAngleXY"	60.0000	
"1202"	"fCombatAngleZ"	60.0000	
"1203"	"fCombatForceSideAngle"	30.0000	
"1204"	"fCombatTorsoSideAngle"	45.0000	
"1205"	"fCombatTorsoStartPercent"	0.3000	
"1206"	"fCombatTorsoStopPercent"	0.8000	
"1207"	"fCombatBlockLeftAngle"	-90.0000	Shields are worn on the left and partially block attacks from 90 degrees left to 30 degrees right of the PC's facing.
"1208"	"fCombatBlockRightAngle"	30.0000	
"1209"	"fCombatDelayCreature"	0.1000	
"1210"	"fCombatDelayNPC"	0.1000	
"1212"	"fAIMeleeWeaponMult"	2.0000	Used in the determination of how far away an NPC will flee if they flee combat and the PC has a melee weapon in hand
"1213"	"fAIRangeMeleeWeaponMult"	5.0000	as above but the PC has a crossbow or Bow in hand
"1214"	"fAIMagicSpellMult"	3.0000	
"1215"	"fAIRangeMagicSpellMult"	5.0000	As above but the PC has a spell readied
"1216"	"fAIMeleeArmorMult"	1.0000	
"1217"	"fAIMeleeSummWeaponMult"	1.0000	
"1218"	"fAIFleeHealthMult"	7.0000	Alters the opponents flee rating when health declines
"1219"	"fAIFleeFleeMult"	0.3000	Used to alter base flee ratings.
"1220"	"fPickPocketMod"	0.3000	
"1221"	"fSleepRandMod"	0.2500	Affects the chance of a mob waking the PC up while asleep in the wilderness.
"1222"	"fSleepRestMod"	0.3000	Unused (Thanks to Damar Stiehl for these two)
"1223"	"iNumberCreatures"	1	
"1224"	"fAudioDefaultMinDistance"	5.0000	
"1225"	"fAudioDefaultMaxDistance"	40.0000	
"1226"	"fAudioVoiceDefaultMinDistance"	10.0000	
"1227"	"fAudioVoiceDefaultMaxDistance"	60.0000	
"1228"	"fAudioMinDistanceMult"	20.0000	
"1229"	"fAudioMaxDistanceMult"	50.0000	
"1230"	"fNPCHealthBarTime"	3.0000	Controls delay before the Opponents health bar disappears
"1231"	"fNPCHealthBarFade"	0.5000	Controls how many seconds the bar "fades" for (rather than abruptly vanishing)
"1232"	"fDifficultyMult"	5.0000	
"1399"	"fMagicDetectRefreshRate"	0.0167	
"1400"	"fMagicStartIconBlink"	3.0000	The number of seconds a spell icon will fade before the spell runs out, on the lower right-hand corner of the screen.
"1401"	"fMagicCreatureCastDelay"	1.5000	
"1431"	"fDiseaseXferChance"	2.5000	The chance of catching a disease if hit by a creature, or looting a diseased creature's corpse.
"1432"	"fElementalShieldMult"	0.1000	
"1435"	"fMagicSunBlockedMult"	0.5000	Vampire weakness
	"fWereWolfRunMult"	1.3000	Werewolf run speed multiplier.

	"fWereWolfSilverWeaponDamageMult"	2.0000	The damage multiplier for silver weapon damage against all werewolves.
	"iWereWolfBounty"	1000	These are the skills and attributes for Werewolf form.
	"fWereWolfStrength"	150.0000	
	"fWereWolfAgility"	150.0000	
	"fWereWolfEndurance"	150.0000	
	"fWereWolfSpeed"	90.0000	
	"fWereWolfHandtoHand"	100.0000	
	"fWereWolfUnarmored"	100.0000	
	"fWereWolfAthletics"	50.0000	
	"fWereWolfAcrobatics"	80.0000	
	"fWereWolfIntelligence"	0.0000	
	"fWereWolfWillPower"	0.0000	
	"fWereWolfPersonality"	0.0000	
	"fWereWolfLuck"	25.0000	
	"fWereWolfBlock"	0.0000	
	"fWereWolfArmorer"	0.0000	
	"fWereWolfMediumArmor"	0.0000	
	"fWereWolfHeavyArmor"	0.0000	
	"fWereWolfBluntWeapon"	0.0000	
	"fWereWolfLongBlade"	0.0000	
	"fWereWolfAxe"	0.0000	
	"fWereWolfSpear"	0.0000	
	"fWereWolfDestruction"	0.0000	
	"fWereWolfAlteration"	0.0000	
	"fWereWolfIllusion"	0.0000	
	"fWereWolfConjuration"	0.0000	
	"fWereWolfMysticism"	0.0000	
	"fWereWolfRestoration"	0.0000	
	"fWereWolfEnchant"	0.0000	
	"fWereWolfAlchemy"	0.0000	
	"fWereWolfSecurity"	0.0000	
	"fWereWolfSneak"	95.0000	
	"fWereWolfLightArmor"	0.0000	
	"fWereWolfShortBlade"	0.0000	
	"fWereWolfMarksman"	0.0000	
	"fWereWolfSpeechcraft"	0.0000	
	"iWereWolfLevelToAttack"	20	Determines the attack range of a Werewolf.
	"iWereWolfFightMod"	100	
	"iWereWolfFleeMod"	100	
	"fWereWolfHealth"	2.0000	
	"fWereWolfFatigue"	400.0000	
	"fWereWolfMagica"	100.0000	
	"fCombatDistaceWereWolfMod"	0.3000	
	"fFleeDistance"	3000.0000	Determines how far away someone will flee.

Index

A

Activate, 64
AddItem, 59
Addition, 26
AddSoulGem, 52
AddSpell, 52
AddToLevCreature, 102
AddToLevItem, 102
AddTopic, 39
AiActivate, 32
AIEscort, 34
AIEscortCell, 34
AiFollow, 34
AiFollowCell, 34
AiTravel, 29
AiWander, 32
AND, 28

B

BecomeWerewolf, 109
Boolean operators, 28

C

Cast, 52
CellChanged, 72
CellUpdate, 68
CenterOnCell, 130
CenterOnExterior, 130
ChangeWeather, 85
character creation, 84
Choice, 41
Cleaning, 110
ClearForceJump, 100
ClearForceMoveJump, 100
ClearForceRun, 100
ClearForceSneak, 35
ClearInfoActor, 41
coc, 130
coe, 130
Commands, 13
companion, 91
Console, 125
console commands, 130
CreateMaps, 130
ctrl-c, 12
ctrl-v, 12
ctrl-x, 12

D

Day, 59
Disable, 65
DisableLevitation, 95
DisablePlayerControls, 82
DisablePlayerFighting, 83
DisablePlayerJumping, 83
DisablePlayerLooking, 83
DisablePlayerMagic, 83
DisablePlayerViewSwitch, 83

DisableTeleporting, 50
DisableVanityMode, 83
Division, 26
DontSaveObject, 66
Drop, 60
DropSoulgem, 52

E

elseif, 27
Enable, 65
EnableBirthMenu, 84
EnableClassMenu, 84
EnableInventoryMenu, 84
EnableLevelUpMenu, 83
EnableLevitation, 95
EnableMagicMenu, 84
EnableMapMenu, 84
EnableNameMenu, 84
EnablePlayerControls, 83
EnablePlayerFighting, 83
EnablePlayerJumping, 83
EnablePlayerLooking, 83
EnablePlayerMagic, 83
EnablePlayerViewSwitch, 83
EnableRaceMenu, 84
EnableRest, 83
EnableStatsMenu, 84
EnableTeleporting, 50
EnableVanityMode, 83
endif, 27
EndWhile, 27
Equip, 62
Error messages, 126
ExplodeSpell, 103
EXPRESSION, 126

F

Face, 31
FadeIn, 86
FadeOut, 86
FadeTo, 86
FillJournal, 130
FillMap, 130
FixMe, 130
Float, 24
ForceGreeting, 40
ForceJump, 100
ForceMoveJump, 100
ForceRun, 100
ForceSneak, 35
Friend Hit (dialogue), 43
Functions, 13

G

game settings, 132
GameHour, 58
Get/Mod/SetAcrobatics, 78
Get/Mod/SetAgility, 78
Get/Mod/SetAlarm, 80
Get/Mod/SetAlchemy, 78

Get/Mod/SetAlteration, 78
Get/Mod/SetArmorBonus, 79
Get/Mod/SetArmorer, 78
Get/Mod/SetAthletics, 78
Get/Mod/SetAttackBonus, 79
Get/Mod/SetAxe, 78
Get/Mod/SetBlindness, 79
Get/Mod/SetBlock, 78
Get/Mod/SetBluntWeapon, 78
Get/Mod/SetCastPenalty, 79
Get/Mod/SetChameleon, 79
Get/Mod/SetConjuration, 78
Get/Mod/SetDefendBonus, 79
Get/Mod/SetDestruction, 78
Get/Mod/SetDisposition, 79
Get/Mod/SetEnchant, 78
Get/Mod/SetEndurance, 78
Get/Mod/SetFatigue, 78
Get/Mod/SetFight, 80
Get/Mod/SetFlee, 80
Get/Mod/SetFlying, 79
Get/Mod/SetHandToHand, 79
Get/Mod/SetHealth, 78
Get/Mod/SetHeavyArmor, 78
Get/Mod/SetHello, 81
Get/Mod/SetIllusion, 78
Get/Mod/SetIntelligence, 78
Get/Mod/SetInvisible, 79
Get/Mod/SetLevel, 79
Get/Mod/SetLightArmor, 78
Get/Mod/SetLongBlade, 78
Get/Mod/SetLuck, 78
Get/Mod/SetMagicka, 78
Get/Mod/SetMarksman, 79
Get/Mod/SetMediumArmor, 78
Get/Mod/SetMercantile, 79
Get/Mod/SetMysticism, 78
Get/Mod/SetParalysis, 79
Get/Mod/SetPCCrimeLevel, 79
Get/Mod/SetPersonality, 78
Get/Mod/SetReputation, 79
Get/Mod/SetResistBlight, 79
Get/Mod/SetResistCorprus, 79
Get/Mod/SetResistDisease, 79
Get/Mod/SetResistFire, 79
Get/Mod/SetResistFrost, 79
Get/Mod/SetResistMagicka, 79
Get/Mod/SetResistNormalWeapons, 79
Get/Mod/SetResistParalysis, 79
Get/Mod/SetResistPoison, 79
Get/Mod/SetResistShock, 79
Get/Mod/SetRestoration, 78
Get/Mod/SetSecurity, 78
Get/Mod/SetShortBlade, 78
Get/Mod/SetSilence, 79
Get/Mod/SetSneak, 78
Get/Mod/SetSpear, 78
Get/Mod/SetSpeechcraft, 79
Get/Mod/SetSpeed, 78
Get/Mod/SetStrength, 78
Get/Mod/SetSuperJump, 79
Get/Mod/SetSwimSpeed, 79
Get/Mod/SetUnarmored, 78
Get/Mod/SetWaterBreathing, 79

Get/Mod/SetWaterWalking, 79
Get/Mod/SetWillpower, 78
GetAIPackageDone, 30
GetAngle, 67
GetArmorType, 98
GetAttacked, 47
GetBlightDisease, 55
GetButtonPressed, 38
GetCollidingActor, 101
GetCollidingPC, 101
GetCommonDisease, 55
GetCurrentAIPackage, 34
GetCurrentWeather, 85
GetDeadCount, 48
GetDetected, 75
GetDisabled, 65
GetDistance, 73
GetEffect, 55
GetFactionReaction, 130
GetForceJump, 100
GetForceMoveJump, 100
GetForceRun, 100
GetForceSneak, 35
GetHealthGetRatio, 77, 78
GetInterior, 87
GetItemCount, 60
GetJournalIndex, 41
GetLineOfSight, 74
GetLocked, 64
GetLOS, 74
GetMasserPhase, 59
GetPCCell, 71
GetPCCrimeLevel, 50
GetPCFacRep, 43
GetPCInJail, 108
GetPCJumping, 91
GetPCRank, 43
GetPCRunning, 91
GetPCSleep, 82
GetPCSneaking, 91
GetPCTraveling, 108
GetPlayerControlsDisabled, 83
GetPlayerFightingDisabled, 83
GetPlayerJumpingDisabled, 83
GetPlayerLookingDisabled, 83
GetPlayerMagicDisabled, 83
GetPlayerViewSwitch, 83
GetPos, 73
GetRace, 43
GetScale, 92
GetSecondsPassed, 58
GetSecundaPhase, 59
GetSpell, 54
GetSpellEffects, 54
GetSpellReadied, 106
GetSquareRoot, 103
GetStandingActor, 76
GetStandingPC, 76
GetStat, 77
GetTarget, 47
GetVanityModeDisabled, 83
GetWaterLevel, 93
GetWeaponDrawn, 106
GetWeaponType, 98
GetWerewolfKills, 109
GetWindSpeed, 86
 Global scripts, 22
 global variables, 24

Goodbye, 41
GotoJail, 49

H

HasItemEquipped, 98
HasSoulgem, 51
Help, 130
HitAttemptOnMe, 48
HitOnMe, 48
HurtCollidingActor, 101
HurtStandingActor, 76

I

if, 27
INFIX to POSTFIX, 126
IsWerewolf, 109

J

Journal, 41

L

LeftEval, 126
local variables, 24
Lock, 64
Long, 24
LoopGroup, 71
LowerRank, 44

M

Mathematical calculations, 26
maximum health, 77
MenuMode, 86
MessageBox, 37
minimumprofit, 91
ModCurrentFatigue, 77, 78
ModCurrentHealth, 77, 78
ModCurrentMagicka, 77, 78
ModFactionReaction, 45
ModHealth, 78
ModPCFacRep, 45
ModRegion, 85
ModScale, 92
ModStat, 77
ModWaterLevel, 93
moto, 130
Move, 66
MoveOneToOne, 130
MoveWorld, 66
Multiplication, 26

O

Objects, 13
OnActivate, 64
OnDeath, 48
OnKnockout, 48
OnMurder, 48
OnPCAdd, 61
OnPCDrop, 61
OnPCEquip, 61
OnPCHitMe, 46
OnPCRepair, 62

OnPCSoulGemUse, 61
OnRepair, 62
OR, 28

P

PayFine, 49, 50
PC Clothing Modifier (dialogue), 42
PC Sex (dialogue), 42
PCClearExpelled, 45
PCExpell, 45
PCExpelled, 43
PCForce1stPerson, 84
PCForce3rdPerson, 84
PCGet3rdPerson, 84
PCJoinFaction, 44
PCLowerRank, 44
PCRaiseRank, 44
PCSkipEquip, 63
PlaceAtMe, 108
PlaceAtPC, 63
PlaceItem, 96
PlaceItemCell, 96
PlayBink, 88
Player Controls, 81
PlayGroup, 70
PlaySound, 56
PlaySound3D, 56
PlaySound3DVP, 56
PlaySoundVP, 56
Position, 67
PositionCell, 68

R

RaiseRank, 44
Random, 88
Rank Requirement (dialogue), 42
RemoveEffects, 55
RemoveFromLevCreature, 102
RemoveFromLevItem, 102
RemoveItem, 59
RemoveSoulgem, 51
RemoveSpell, 52
RepairedOnMe, 62
Resurrect, 81
Return, 88
RightEval, 126
Rotate, 67
RotateWorld, 67

S

SameFaction, 43
Say, 56
SayDone, 56
scripting window, 11
ScriptRunning, 88
SetAngle, 69
SetAtStart, 70
SetDelete, 103
SetFactionReaction, 45
SetJournalIndex, 41
SetPCFacRep, 45
SetPos, 69
SetScale, 92
SetStat, 77
SetWaterLevel, 93

SetWerewolfAcrobatics, 108
SetWillpower, 78
Short, 24
ShowMap, 87
ShowRestMenu, 81
ShowVars, 130
SkipAnim, 71
StartCombat, 46
StartScript, 22, 88
StayOutside, 91
StopCellTest, 130
StopCombat, 46
StopScript, 22, 88
StopSound, 57
StreamMusic, 56
string variables, 24
Subtraction, 26
sv, 130
Syntax, 23

T

Talked to PC (dialogue), 42
TestCells, 130

TestInteriorCells, 130
TestModels, 130
Text defines, 39
ToggleAI, 130
ToggleBorders, 130
ToggleCombatStats, 130
Tribunal Script Functions, 90, 107
Troubleshooting, 125
TurnMoonRed, 108
TurnMoonWhite, 108
tutorial, 11

U

UndoWerewolf, 109
Unlock, 64
UsedOnMe, 61

W

WakeUpPC, 82
While, 27