Morrowind Scripting for Dummies

(8th Edition)

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

Table des matières

Avant-propos à la 8e édition

Eh oui, une nouvelle mise à jour de MSFD ! pas mal de changements cette fois-ci. J'ai continué ma reclassification des fonctions. Les fonctions de Tribunal et Bloodmoon sont maintenant incluses dans le reste des fonctions (tout en étant clairement marquées) ; même choses pour les fonctions de type Get/Set/ModStat – celles qui ont un rapport avec la magie se trouvent dans la section Magie, celles qui sont en rapport avec le combat dans la section Combat, etc…. j'ai reformaté toutes les entrées de fonctions pour les rendre plus visibles. Il y a aussi bon nombre de corrections et d'ajouts : consultez la liste des *Idle* dans *AIWander*, regardez les infos détaillées sur les fonctios telles que *OnDeath*, *OnActivate*, ou découvrez que certaines fonctions, comme *PositionCell,* utilisent une unité différente pour les paramètres d'angles (des minutes au lieu de degrés). Il y a beaucoup de nouveaux détails sur les fonctions *RemoveItem*, *OnPCEquip* et *SkipEquip*. J'ai rajouté autant que je pouvais des scripts de démonstrations. J'ai rassemblé une liste de tips sur le dialogue sur les forums officiels. Pour faciliter vos tests, j'ai aussi ajouté une liste des commandes consoles ; vous pourrez peut-être y trouver des choses utiles pour vos scripts. Une belle collection de trucs et astuces a aussi été incorporée : détecter le chargement d'une sauvegarde, des éditeurs de scripts, un script qui permet de calculer rapidement cosinus et sinus, et bien d'autres choses encore.
Je voudrais mentionner quelque chose qui est devenu de plus en plus évident au cours des dernières discussions : apparemment les différentes versions du jeu (Morrowind, les extensions, versions européenne/américaine), ont un certain nombre de points qui les distinguent les unes des autres en ce qui concerne les scripts. Cela pourrait expliquer les informations contradictoires. Certaines fonctions semblent avoir perdu leur fondement dans les dernières versions, d'autres ont été corrigées, etc. En l'absence d'un journal complet des changements effectués par Bethesda, il est malheureusement impossible de donner une réponse définitive concernant ces fonctions.
Une fois encore, je voudrais insister sur le fait que ce guide n'existerait pas sans l'aide et le soutien de la communauté des moddeurs. Je voudrais remercier tout spécialement Nigedo, DinkumThinkum, ManaUser, ThePal, Erstam, JDGBOLT, Klinn et MentalElf, pour leur aide précieuse pour la détection et la correction des incohérences et des erreurs dans la dernière édition, ou pour avoir bien voulu partager leurs informations et scripts avec moi. Merci aussi à Emma, pour toutes les précisions qu'elle a apporté sur le sujet du dialogue des forums.

GhanBuriGhan

Notes du traducteur

Moi aussi j'ai quelques petites choses à dire ; je n'ai pas de raison de m'en priver !

Voilà donc la première version de Morrowind Scripting for Dummies entièrement traduite en français. Il aura quand même fallu attendre pas mal de temps… heureusement pour nous qu'Oblivion n'est pas encore achevé !

Tout d'abord, j'aimerais préciser que je ne suis absolument pas un scripteur accompli : j'ai appris sur le tas en suivant le tutorial et je n'ai pas eu l'occasion de tester l'ensemble des fonctions. Il est donc possible que le document contienne des erreurs. D'ailleurs, comme le dit GhanBuriGhan dans son avant-propos, les versions de jeu peuvent être différentes ; il vaut donc mieux prendre quelques précautions avant de foncer tête baissée. La communauté française est également très active et je suis sûr que vous trouverez toujours quelqu'un pour vous dépanner en cas de pépin.

Dans tous les cas, je considère que le retour des lecteurs est essentiel alors n'hésitez pas à me signaler les éventuels problèmes pour que je puisse effectuer les corrections nécessaires, que ce soit au niveau des scripts ou au niveau de la traduction.

Enfin je tiens à remercier tous ceux qui, de près ou de loin, m'ont donné un coup de main. Citons dans le désordre : Emma Indoril, Stilgar, flox, aqualonne pour leur participation dans la traduction ; wiwi et toute la communauté de wiwiland pour leur travail remarquable autour de Morrowind.

En espérant que ce document servira au plus grand nombre,

Zozio,
nidupinson@wanadoo.fr
sur le forum de wiwiland (www.wiwiland.com/morrowind/forum/)

Introduction

Utilisation du guide

Si vous êtes nouveau dans le monde des scripts, vous devriez probablement commencer par lire cette introduction et évidemment faire le tutorial. Il s'agit de ma meilleure tentative pour expliquer les scripts, ce qu'ils font et comment en programmer un, en des termes simples.

Deuxièmement, c'est un manuel, une référence. La plus grande partie de ce guide est une documentation des fonctions disponibles. Celle-ci n'est pas écrite pour les débutants, vous êtes supposés déjà connaître les bases sur les scripts. J'ai cependant essayé de mettre à disposition plus d'information, d'explications et d'exemples que le fichier d'aide original, afin qu'il soit plus facile d'utiliser ces fonctions correctement dans vos scripts. Utilisez l'index à la fin du document pour trouver des renseignements sur une fonction spécifique, ou la table des matières pour trouver les fonctions relatives à un domaine d'intérêt. Si vous obtenez des erreurs, la section *Troubleshooting* peut vous aider. L'Appendice contient des listes qui peuvent aussi être utiles en tant que référence.

Troisièmement, en tant que scripteur 'pro', vous pourrez trouver la section Trucs et Astuces intéressante ; elle couvre à la fois les conseils de bases et les techniques de scripting avancé.

Enfin, un petit conseil : ne suivez pas ce qui est écrit à la lettre. Les renseignements contenus dans ce guide représentent le meilleur des connaissances et de la sagesse des forums ; cela ne veut pas dire qu'il n'y a pas d'erreurs ou d'oublis etc…Par exemple, si j'écris qu'une fonction ne prend pas de variables en arguments, c'est *probablement* le cas – mais si c'est important pour votre mod, essayez quand même. Cela a pu changé avec un patch ou peut-être que personne ne l'a remarqué avant, tout simplement. Alors faites un test, et si vous trouvez quelque chose de nouveau, faites-le moi savoir, je l'ajouterais dans une nouvelle mise à jour.

Qu'est-ce qu'un script?

Les scripts sont à la base des morceaux de code écrit dans un langage de script spécial (Je l'appellerai TES Script à partir d'ici). Ces petits « programmes » s'exécuteront pendant le jeu et peuvent réaliser certaines choses dans le jeu, beaucoup de choses en fait : déclencher des évènements, contrôler le temps et le lieu, faire apparaître, bouger, disparaître des objets et des créatures, donner des messages au joueur, changer les stats, et même changer le temps qu'il fait – Les possibilités sont énormes.
TES Script est un langage de scripts unique, il n'est pas utilisé en dehors du TES Construction Set. En tant que langage de scripts, il possède certaines limitations comparé au langage de programmation « réel » tel que le C++ :

1. La portée du TES Script est limitée – n'espérez pas programmer des choses qui ne sont pas dans le jeu d'une manière ou d'une autre – ce qui ne revient pas à dire qu'on ne peut pas accomplir des choses nouvelles et inhabituelles avec le scripting ! Mais on ne peut pas utiliser le TES Script pour, disons, programmer un traitement de texte.
2. Le TES Script n'est pas non plus un SDK (Software Development Kit) qui vous laisse modifier le code source du jeu. C'est pourquoi vous ne pouvez pas utiliser le TES Script, pour, par exemple, ajouter de nouveaux effets climatiques. Ceux-ci sont codés dans l'exécutable et il vous faudrait changer l'exécutable du jeu pour cela.
3. Le langage est interprété et non pas compilé – le code a besoin d'un programme séparé (dans ce cas, Morrowind) pour s'exécuter – à l'inverse d'un code compilé qui peut s'exécuter par lui-même, comme un .exe.

Que peuvent faire les scripts ?

Les scripts pour Morrowind permettent au jeu de réagir dynamiquement à ce que fait le joueur dans le monde virtuel. On peut se servir des scripts pour gérer des quêtes complexes. On peut se servir des scripts pour créer des objets personnalisés, réalisant des actions au-delà des simples enchantements. On peut se servir des scripts pour créer des pièges. On peut se servir des scripts pour modifier le comportement des PNJ ou des créatures. Vous vous souvenez de la création de votre personnage ? c'est à la base des scripts. Vous voyez Fargoth fureter près de sa cachette à Seyda Nihin? C'est un script qui le dirige. Vous avez libéré des esclaves ? c'est aussi un script. Une réponse courte à cette question est donc : beaucoup de choses.

Ce qu'ils ne peuvent faire:

Le TES Script est limité dans ses capacités – toutes les fonctions ne sont pas accessibles et parfois les utilisations possibles ne sont pas du tout ce que vous voudriez qu'elles soient. En fait, certaines fonctions sont buggées ou clairement mal définies. Dans beaucoup de cas, des scripteurs intelligents peuvent trouver des astuces de programmation pour dépasser les limitations apparentes mais n'espérez aucun miracle. De nombreuses choses sont codées dans l'exécutable et ne peuvent pas ou seulement indirectement être influencées par les scripts.

Scripting tutorial

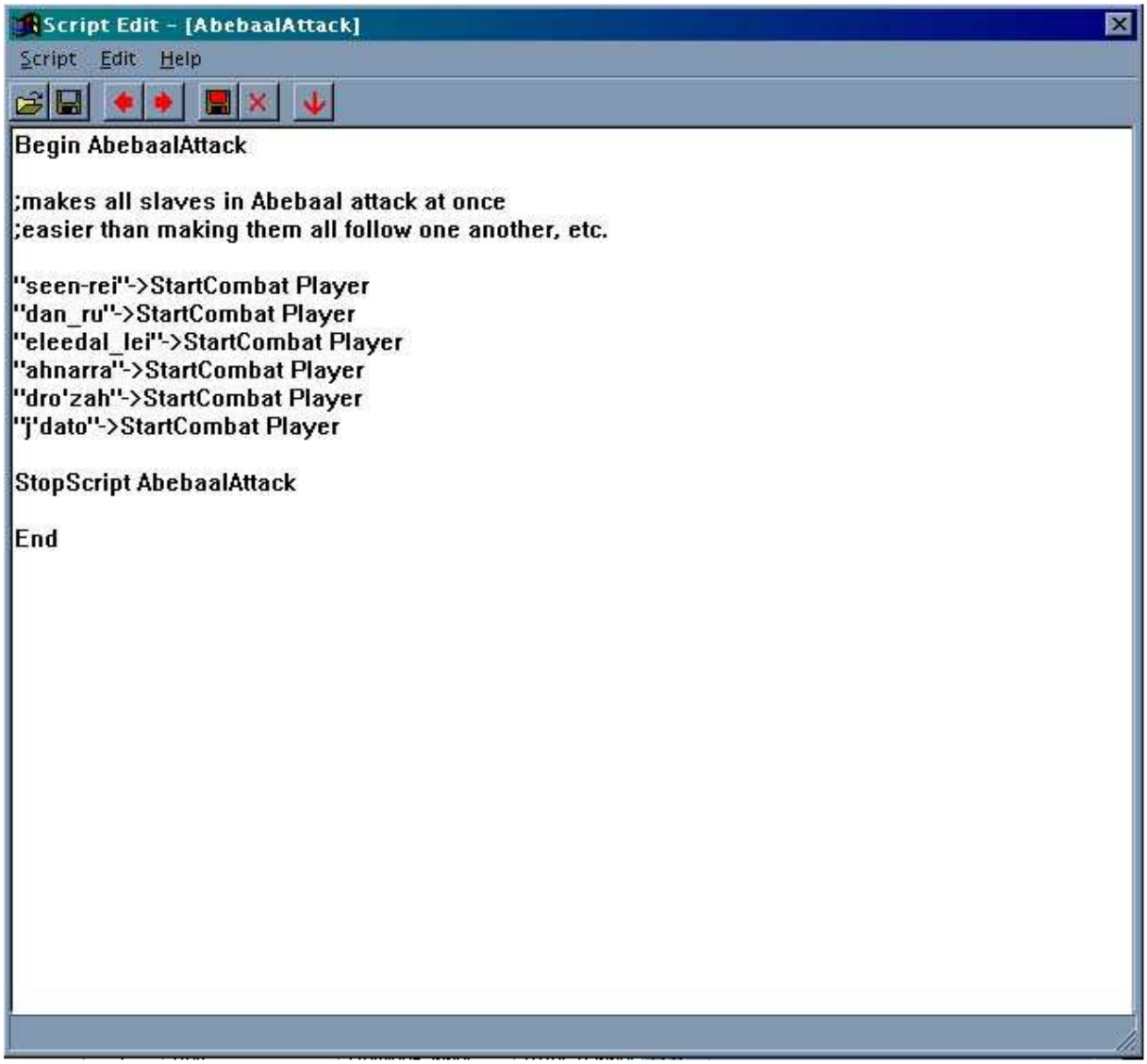
Si vous êtes complètement nouveau dans les scripts ou la programmation en général, le TES script peut vous sembler un petit effrayant – j'ai donc écrit un tutorial conséquent qui va vous aider à réaliser votre premier script. J'expliquerai aussi les principaux éléments du langage au fur et à mesure. Il y aura d'autres explications en route mais les instructions les plus importantes seront écrites en **gras**.

C'est parti!

Commençons par ouvrir l'éditeur de script : **Démarrez le TES Construction Set, ouvrez le fichier Morrowind.esm et sélectionnez ensuite Edit Scripts à partir du menu *Gameplay* pour ouvrir la fenêtre de script.**

La fenêtre de script

Vous entrez dans l'éditeur ou bien **en sélectionnant *Gameplay – Edit Scripts* à partir du menu**, ou bien en cliquant sur le bouton d'édition de script (le crayon) dans la barre des tâches ou bien à partir d'un objet (*Object*) ou de la boîte de dialogue d'un PNJ, en cliquant sur le bouton […] à côté du champ script. La fenêtre d'édition est assez basique :



Examinons les boutons de la barre des tâches, de gauche à droite : *Ouvrir (Open)* vous permet de sélectionner un script à éditer. *Sauver (Save)* vérifie les éventuelles erreurs du script ouvert, le compile ou affiche les messages d'erreurs – notez, cependant, que le plugin et donc le script n'est pas réellement sauvé sur le disque à ce moment précis. Durant la programmation de longs scripts, utilisez souvent la commande de sauvegarde de la fenêtre principale du TESCS après avoir sauver le script, juste au cas où le TESCS planterait.
Notez que si vous éditez un script et cliquez soudainement sur "*save plugin*" pour sauvegarder au milieu de votre travail, la mise à jour du script ne sera PAS sauvegardée. Vous devez le sauver manuellement d'abord. Donc, si vous vous contentez de fermer la fenêtre de script, cela ne signifie pas que le script a été sauvegardé. Vous devez le faire vous-même (Merci à Kir pour cette indication).
Les flèches Avant et Arrière permettent de passer respectivement au script suivant ou précédent (dans l'ordre alphabétique). Si vous donnez à vos noms de scripts des étiquettes communes, il sera plus facile de passer de l'un à l'autre : par exemple, pour un projet, nommez chaque script de la façon suivante AA_Nom_du_script, ils seront automatiquement placés en début de la liste, tous ensemble. *Tout compiler (Compile all)* recompile tous les scripts (l'utilité ? je ne sais pas vraiment). Enfin le bouton *Effacer (Delete)* supprime un script et le dernier bouton « flèche descendante » ferme la fenêtre d'édition.
Le menu d'aide (help) donne accès aux pages des fonctions et des commandes du fichier d'aide (utilité réduite, d'où la création de ce manuel !).
Vous pouvez copier et coller en utilisant les habituels ctrl-c (copier), ctrl-x (couper) et ctrl-v (coller) de Windows.

Ce que l'on veut

Avant de commencer à écrire notre script, nous devons décider ce que nous voulons faire. Pour ce tutorial, j'ai décidé de faire un **coffre à énigme : le coffre posera une énigme et seule la bonne réponse ouvrira le coffre. Si la réponse est erronée, un piège s'activera, blessant le joueur, et le coffre ne s'ouvrira pas.** C'est une tâche assez complexe, mais on va y aller pas à pas.

Ecrire un script

Ok, une fois la fenêtre d'édition ouverte, **cliquez sur la partie principale de la fenêtre.** C'est ici que vous écrirez votre script.

Nommer un script: Begin et End

Tout d'abord, nous devons donner un nom au script – chaque script doit commencer par la déclaration de son nom. **Alors tapez :**

```
Begin mon_premier_script
```

dans la fenêtre d'édition. Notez la présence des underscores (soulignés) : le nom doit être en un mot. Notez aussi que la casse du texte importe peu, *Begin* peut très bien s'écrire sans majuscule : *begin.* Le nom est le lien qui permet au TESCS de connaître le script. Essayez de sauver le script maintenant : vous obtiendrez le message d'erreur suivant «*you need to end your script with end scriptname* », c'est-à-dire « vous devez terminer votre script par end nom_du_script ».
Pour que le script soit valide, vous devez donc lui indiquer sa terminaison : **ensuite, écrivez**

```
End
```

sur une ligne en-dessous de la précédente. Comme vous le voyez, on peut se passer du nom du script ; de plus, *end* fonctionne aussi. Si vous sauvez maintenant, vous verrez le nom du script apparaître dans la barre de titre de la fenêtre, ce qui signifie que le script est valide. C'est le script le plus court possible, et bien sûr, il ne fait rien du tout.

Détecter une action du joueur

Ensuite, nous avons besoin d'un moyen de déterminer si le joueur essaie d'ouvrir le coffre. Il faut distinguer les objets, les fonctions, et les commandes –
Les **Objets** sont toutes les choses du monde : objets visibles, créatures, PNJ, ou encore les sons.
Les **Fonctions** sont tous les « mots » du TES script qui nous permettent ou de manipuler ces objets ou collecter des informations à leur propos.
Les **Commandes** sont tous les « mots » qui structurent le langage, mais qui ne manipulent pas les objets du jeu – par exemple, le mot "Begin" que nous avons utilisé pour indiquer au script son nom.
Pour dire au jeu sur quel objet on applique une fonction donnée, on utilise la « flèche » : -> (un tiret suivi d'un supérieur). Vous spécifiez le nom de l'objet pour la fonction à gauche (c'est l'objet appelant) et la fonction à appliquer à droite :

```
Identifiant_Objet -> fonction, [paramètres]
```

Une fonction peut ne pas avoir de paramètres. Ceux-ci peuvent être d'autres identifiants d'objets, des nombres et dans certains cas des variables.

Pour notre coffre, nous avons besoin de la fonction `OnActivate` : cette fonction indique si le joueur a « activé » l'objet ou non dans le monde. Elle renvoie 1 (c'est-à-dire « vrai » en terme de programmation) si l'objet a été activé : le joueur a ciblé l'objet et a appuyé sur la touche « utiliser » (soit espace par défaut). Donc nous devons vérifier si *OnActivate* renvoie « vrai » à tout moment dans le jeu. **Editez donc votre script ainsi :**

```
Begin mon_premier_script

If ( OnActivate == 1 )
       ; ici nous écrirons ce qui se passera quand on ouvrira le coffre
endif

End
```

Encore quelques petites explications : la commande "*if*" (*«si»*) est ici pour vérifier la condition – dès que l'expression entre parenthèses est évaluée à vrai, les lignes de codes suivantes seront exécutées jusqu'à la commande "endif". Le "==" vérifie qu'une expression (ici la fonction "*OnActivate*") à gauche est égale à l'expression de droite (l'entier 1). Si vous oubliez la commande *endif* (*fin de* si) après une commande *if*, l'éditeur signalera une erreur. Le point-virgule ";" désigne un commentaire – quoi que vous écriviez derrière le point-virgule, ce sera ignorée pendant l'exécution du script. Si vous écrivez des scripts importants, vous apprécierez cette possibilité.

Ecrire du texte et obtenir des réponses du joueur

Maintenant nous voulons que notre coffre énonce son énigme au joueur. Pour cela, nous utiliserons la fonction *MessageBox* qui permet d'afficher du texte à l'écran mais aussi des choix que le joueur peut sélectionner. Malheureusement Morrowind ne permet pas au joueur de taper la réponse à notre énigme ; il faudra donc donner plusieurs possibilités de réponses. Voici la ligne correspondante :

```
MessageBox "Sans voix cela crie, sans ailes cela vole, sans dent cela mord, sans bouche cela
murmure. Qu'est-ce que c'est ?", "Chauve-souris", "Vieille femme", "Vent", "Spectre"
```

Le premier texte correspond au texte affiché dans la boîte, les autres textes, séparés par des virgules indiquent au jeu d'en faire des boutons.
Mais comment s'assure-t-on que l'énigme n'est énoncée qu'une seule fois, et pas à chaque fois ? Nous voici à un point important : l'utilisation des conditions **do once** (ne faire qu'une seule fois) et des variables d'état. Les débutants en scripting rencontrent un grand nombre de problèmes car ils ne se rendent pas bien compte de la façon dont s'exécutent les scripts et de la façon dont ils doivent être structurés. Alors jetons un œil à cela.

Comment les scripts locaux sont exécutés

Chaque script attaché à un objet ou à un PNJ (script local) est exécuté *à chaque frame (image) que le jeu affiche à l'écran* tant que la cellule contenant l'objet est active (pour les intérieurs, seule la cellule dans laquelle se trouve le joueur est active, pour les extérieurs la cellule où se trouve le joueur et ses voisines sont actives). Donc le *script complet* (et pas une seule ligne) est exécuté 10 à 60 fois par seconde, ou plus selon la vitesse de votre ordinateur ! Il vaut mieux s'imaginer que chaque script local est encapsulé dans une boucle **tant que** ("*while-loop*") :

```
while (l'objet est dans une cellule active)

[Votre script]

endwhile
```

C'est pourquoi le script suivant affiche continuellement des messages (s'il est attaché à un objet ou un PNJ situé dans la même cellule que le joueur). Essayez si vous voulez :

```
Begin Message_script

MessageBox "Des milliers de messages inutiles"

End Message_script
```

Cet exemple est relativement inoffensif mais imaginez ce qui se passerait si vous écriviez une ligne qui ajoute un objet dans l'inventaire du joueur ou qui place un monstre à côté de lui !
Pour cette raison, les constructions "Do Once" sont essentielles et vous les utiliserez probablement beaucoup dans vos scripts. Continuons notre tutorial : nous devons déclarer une variable et l'utiliser pour être sûr que le message ne s'affichera bien qu'une fois. **Modifiez votre script ainsi :**

```
Begin mon_premier_script

Short variable_de_controle

If ( OnActivate == 1 )
       If ( controlvar == 0)
              MessageBox "Sans voix cela crie, sans ailes cela vole, sans dents cela mord,
sans bouche, cela murmure. Qu'est-ce que c'est ?", "Chauve-souris", "Vieille femme", "Vent",
"Spectre"
              Set variable_de_controle to 1
       endif
endif

End
```

(Remarquez que la commande MessageBox doit être sur une ligne dans l'éditeur !)

"*Short variable_de_controle*" déclare une nouvelle variable nommée "*variable_de_controle*", de type short. Pour le moment, il vous suffit de savoir qu'elle contient des valeurs entières (positives ou négatives). Une variable est un emplacement qui peut prendre différentes valeurs. Nous connaissons déjà la commande *if*, la commande *set* est nouvelle mais assez simple – elle positionne notre variable qui avait auparavant la valeur 0 (toutes les variables sont initialisées à 0 lors de leur déclaration) à 1. Ceci, lié à la commande *if (variable_de_controle == 0)* produit une condition do once – à la prochaine frame, le script sera exécuté après que la variable ait été placée à 1 et la boîte de dialogue ne sera pas affichée une deuxième fois.
Maintenant notre script peut être exécuté alors testons-le :

- **Sauvez le script et fermer la fenêtre de l'éditeur.**
- **Allez, dans le TESCS, *Object window*, sélectionnez l'onglet container et ouvrez "chest_small_01".**
- **Changez l'ID du coffre en "tutorial_chest"**
- **Dans la liste déroulante script, choisissez mon_premier_script**
- **Sauvez l'objet en tant que nouvel objet (*new object*), sauver le mod, et quitter le construction set. N'oubliez pas de cocher l'esp dans Fichiers Données ! Démarrez Morrowind et chargez une sauvegarde.**
- **Appelez la console (habituellement ² sur les claviers français, à gauche de "1" sur le clavier principal) et tapez dans celle-ci :**
  PlaceAtPC tutorial_chest 1,1,1
  et tapez entrée.

Reculez d'un pas (je parlais de votre personnage, bien sûr !); vous devriez voir un petit coffre sur le sol devant vous. Cliquez sur celui-ci pour voir apparaître notre message :

Cliquer sur les boutons ne fera que fermer la boîte de dialogue pour le moment et cliquer sur le coffre une seconde fois ne déclenchera rien – ce qui signifie que la condition do-once fonctionne bien.
Ok, quittez Morrowind et revenez dans l'éditeur pour recharger votre plugin.
Nous devons maintenant déterminer quelle est la réponse sélectionnée par le joueur et écrire le script approprié pour les réponses, bonnes ou mauvaises. La fonction pour tester la réponse choisie est "`GetButtonPressed`". Cette fonction renvoie un nombre correspondant au bouton de la boîte de dialogue cliqué. Elle renverra "0" pour le premier bouton ("chauve-souris" dans notre exemple) et 1, 2, 3 respectivement pour les boutons suivants, selon l'ordre dans lequel vous les avez définis dans la fonction *MessageBox*. Tant qu'aucune réponse n'a été donnée, la fonction renverra -1 ; nous devons en tenir compte aussi.

La fonction "`Activate`" ouvrira le coffre ; en fait *Activate* déclenche simplement l'action standard qui s'exécute habituellement lorsque vous « utilisez » l'objet sélectionné – les portes s'ouvriront, les PNJs entameront le dialogue etc.
La mise à jour suivante montre aussi comment on peut utiliser les variables de contrôle pour forcer MW à enchaîner les fonctions dans un ordre précis, bien que le script entier s'exécute à chaque frame : incrémenter simplement la variable de contrôle et tester dans une série de if – elseif. C'est un moyen très sûr pour les scripts de MW – pas toujours nécessaire, mais sûre.

Editez votre script :

```
Begin mon_premier_script

Short variable_de_controle
Short bouton

If ( OnActivate == 1 )
	If ( variable_de_controle == 0)
		MessageBox "Sans voix cela crie, sans ailes cela vole, sans dents cela mord,
sans bouche, cela murmure. Qu'est-ce que c'est ?", "Chauve-souris", "Vieille femme", "Vent",
"Spectre"
		Set variable_de_controle to 1
	elseif variable_de_controle > 1
		Activate
	endif
endif

if (variable_de_controle == 1)
	set bouton to GetButtonPressed
	if ( bouton == -1 )
		return
	elseif ( bouton == 2)
		MessageBox "Réponse juste"
		Activate
		set variable_de_controle to 2
	else
		MessageBox "Réponse erronée"
		set variable_de_controle to -1
	endif
endif

End
```

Regardez la partie qui commence par "*if (variable_de_controle == 1)*". Nous avons positionné variable_de_controle à 1 dès que le coffre est activé. Maintenant nous devons tester sur quel bouton on clique. On fait cela en affectant à la nouvelle variable "*bouton*" la valeur retournée par *GetButtonPressed*. Comme le script tourne toujours, même lorsque le jeu semble attendre une réponse, nous devons d'abord tester qu'aucun bouton n'a été pressé – return indique au jeu d'arrêter l'exécution du script pour la frame courante.
La réponse correcte est le vent, ce qui correspond au bouton numéro deux – si on appuie sur le bouton numéro deux, nous dirons au joueur qu'il a donné la bonne réponse et "*Activate*" ouvrira l'inventaire du coffre, comme c'est le cas normalement.
Toutes les autres valeurs de bouton signifient que le joueur a choisi une mauvaise réponse, donc nous pouvons ici utiliser la commande "*else*". Dans ce cas, nous dirons au joueur qu'il s'est trompé et le coffre ne sera pas activé.
Maintenant regardez le petit ajout au début du script :

```
If ( OnActivate == 1 )
	If ( variable_de_controle == 0)
		MessageBox "Sans voix cela crie, sans ailes cela vole, sans dents cela mord,
sans bouche, cela murmure. Qu'est-ce que c'est ?", "Chauve-souris", "Vieille femme", "Vent",
"Spectre"
		Set variable_de_controle to 1
	elseif  variable_de_controle > 1
		activate
```

```
        endif
endif
```

Cela signifie que, à chaque fois que le coffre sera activé dans le futur, il ne s'ouvrira que si variable_de_controle est supérieur 1. Regardez au-dessus : lorsque le joueur donne une mauvaise réponse, variable_de_controle passe à -1, il ne pourra donc plus ouvrir le coffre. Mais s'il connaît la bonne réponse, *variable_de_controle* passe à 2 ; il pourra ouvrir le coffre dès qu'il le souhaite. Sauvez et tester votre plugin comme décrit précédemment.

Premier bug

Vous devriez avoir noté que le script fait presque – mais pas vraiment – ce que l'on voulait. Aprrs avoir cliqué sur la bonne réponse, l'inventaire du coffre ne s'ouvre pas comme on l'attendait. Pourtant la logique du script paraît correcte, alors qu'est-ce qui ne va pas ? **Essayons le script suivant (changez la partie correspondante de votre script) :**

```
if (variable_de_controle == 1)
        set bouton to GetButtonPressed
        if ( bouton == -1 )
                return
        elseif ( bouton == 2 )
                MessageBox "Réponse juste"
                set variable_de_controle to 2
        else
                MessageBox "Réponse erronée"
                set variable_de_controle to -1
        endif
elseif ( variable_de_controle == 2 )
        Activate
endif
```

Vous voyez comment j'ai bougé la commande *Activate* dans la section qui teste si variable_de_controle == 2 ? on obtient une séquence « propre » d'évènements et, comme dit plus haut, cela peut-être très important lorsqu'on scripte – évitez toujours de faire trop de choses en une fois !**Bien, lancez le jeu et testez-le.**

Super, maintenant l'inventaire s'ouvre comme prévu, mais quoi encore ? le curseur est tout lent et on ne peut plus fermer l'inventaire ! regardez au-dessus – *variable_de_controle* a été positionnée à 2 et cette valeur ne change plus – cependant le jeu procède continuellement à des "*Activate*", à chaque fois que le script est exécuté, à chaque frame ! C'est pourquoi nous ne pouvons pas fermer l'inventaire – il se rouvre immédiatement. **Donc changez cette partie du script :**

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

Testez à nouveau : maintenant tout marche comme nous le voulions. J'espère que je ne vous ai pas embrouillé avec cette petite balade du côté du débuggage, mais c'est quelque chose d'essentiel à savoir – vous devrez constamment repenser vos scripts et essayer différentes façons de faire pour réussir.

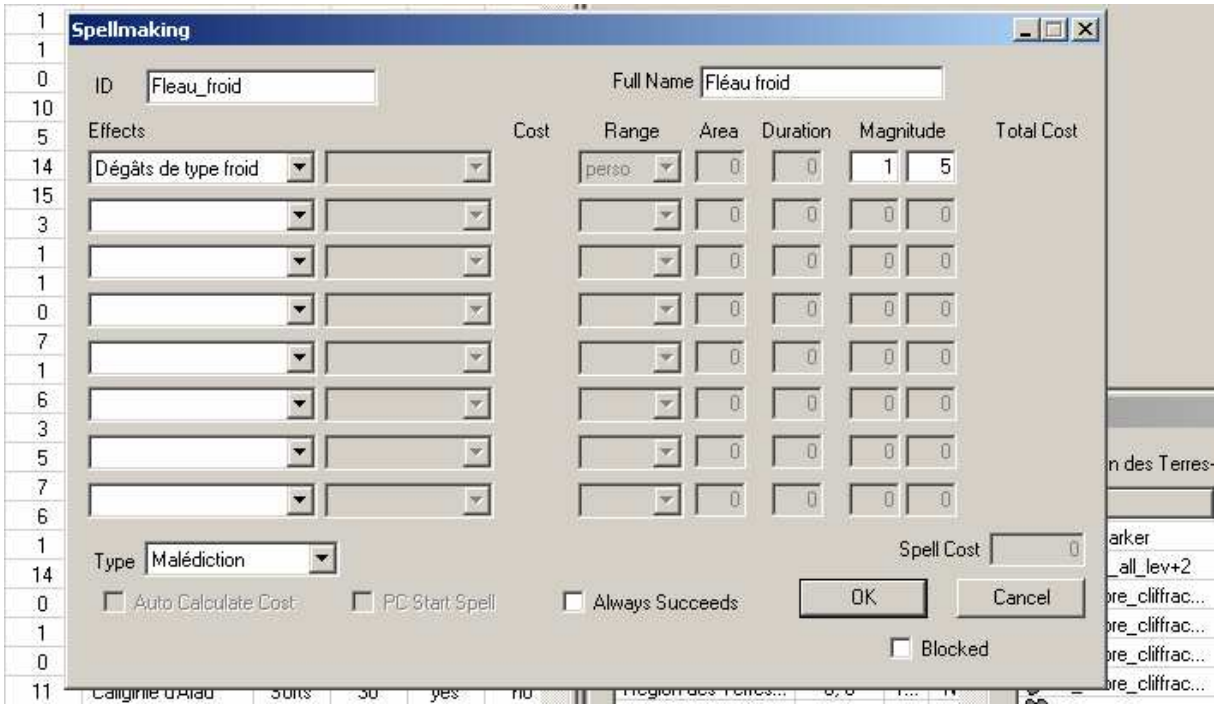
Qu'est-ce qu'il nous manque ? le piège bien sûr !

Placer un sort sur le joueur

Notre coffre placera une malédiction sur le joueur s'il donne une mauvaise réponse.

Commencez par choisir l'onglet spellmaking de l'éditeur, cliquez avec le bouton droit de la souris et sélectionnez "new". Tapez "Fleau_Froid" dans la case ID du sort, nommez-

le "Fléau Froid ", et choisissez le type "Malédiction", des dégâts de type froid magnitude 1-5. Cela devrait ressembler à l'image ci-dessous.



Maintenant, nous devons maudire le joueur. Pour cela, nous utilisons la fonction *AddSpell*. Après quelque temps, nous lèverons la malédiction avec la fonction *RemoveSpell* ; nous aurons donc besoin d'un timer. Modifiez votre script :

```
Begin mon_premier_script

Short variable_de_controle
Short bouton
Float timer

If ( OnActivate == 1 )
	If ( variable_de_controle == 0)
		MessageBox "Sans voix cela crie, sans ailes cela vole, sans dents cela mord,
sans bouche, cela murmure. Qu'est-ce que c'est ?", "Chauve-souris", "Vieille femme", "Vent",
"Spectre"
		Set variable_de_controle to 1
	elseif  variable_de_controle > 1
		activate
	endif
endif

if (variable_de_controle == 1)
	set bouton to GetButtonPressed
	if ( bouton == -1 )
		return
	elseif ( bouton == 2)
		MessageBox "Réponse juste"
		Activate
		set variable_de_controle to 2
	else
		MessageBox "Réponse erronée"
		Player -> AddSpell, "Fleau_froid"
		set variable_de_controle to -1
	Endif
elseif ( variable_de_controle == 2 )
	Activate
	Set variable_de_controle to 3
```

```
elseif ( variable_de_controle == -1 )
	Set timer to ( timer + GetSecondsPassed )
	if timer > 10
		Player -> RemoveSpell, "Fleau_froid"
		set variable_de_controle to -2
	endif
endif
End
```

Regardons ça. Player -> AddSpell, "Fleau_froid" place la malédiction créée auparavant sur le joueur. Remarquez l'utilisation de "Player -> " qui cible effectivement le joueur. Sans cela, nous maudirions le coffre (le script lui étant attaché, c'est l'objet par défaut), ce qui n'a pas de sens …

Ce morceau :

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

…vous montre comment faire un timer dans Morrowind. *GetSecondsPassed* est une fonction qui renvoie le temps en secondes écoulé depuis la dernière frame. Comme ce n'est normalement qu'une fraction de seconde (le script étant appelé à chaque frame), il est préférable d'utiliser un flottant (float) – une variable qui peut stocker un nombre décimal. Quand le timer atteint 10 secondes, nous levons la malédiction en nous assurant de ne le faire qu'une seule fois :

```
		Player -> RemoveSpell, "Fleau_froid"
		set variable_de_controle to -2
```

Ok, sauvez et testez. Ça marche bien, non ? Eh bien, presque. Essayez la chose suivante : faites vous maudire et ouvrez votre inventaire. Attendez. Vous voyez comment la malédiction se termine sans vous blesser ? Bien sûr : le script tourne toujours mais les effets des sorts ne sont calculés que pendant le jeu, et non pas pendant que vous êtes dans le menu. Nous ne voulons pas que le joueur s'en tire à si bon compte ; nous avons donc besoin de mettre quelque chose dans notre script qui l'empêche de s'exécuter quand nous sommes dans les menus. Par chance, il existe la fonction *MenuMode*, qui retourne 1 quand vous êtes dans le menu. Nous pouvons placer ceci au début du script :

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

Souvenez-vous : *return* indique au jeu de stopper l'exécution du script pour la frame courante. Ok, maintenant notre script est complet. Félicitations ! Si vous le voulez, vous pouvez essayer d'autres choses avec ce script. Placez le coffre dans un lieu du jeu et verrouillez-le. Ensuite essayez de le déverrouiller (fonction `Unlock`) en plus de l'activer dans le script. Essayez d'ajouter un son lorsque l'énigme reçoit une réponse correcte (`PlaySound3D, "skillraise"`). Essayez d'utiliser la fonction "`Cast`" à la place de "*AddSpell*". Ceux qui ont essayé ce tutorial ont pu remarquer un bug potentiel : que se passe-t-il si le joueur quitte la zone où se trouve le coffre avant que la malédiction n'ait été levée ? Comment corrigeriez-vous cela ?

Voici le script final :

```
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
```

En savoir plus

Après avoir suivi ce tuto, vous vous demandez sûrement comment continuer à apprendre. Un bon moyen est de regarder les scripts de ce guide ou ceux qui sont déjà présents dans le jeu (ceux de Bethesda ou ceux de mods). Essayez de trouver un script qui se rapproche de ce que vous voulez faire et copier la partie qui vous intéresse pour la modifier à votre convenance. Lisez les informations générales et la description des fonctions que vous avez prévu d'utilisées ci-dessous. Le regroupement des fonctions selon un ordre thématique devrait vous aider à trouver les bonnes.
Enfin, les forums officiels sont une bonne source d'informations (utilisez la fonction de recherche) ; vous pouvez y trouver de l'aide pour un problème spécifique. Pour le reste c'est la pratique, la pratique, la pratique

Information générale : Scripts, Commandes et Syntaxe

Types de scripts

Scripts locaux

Tout script s'exécutant sur un objet ou un acteur dans le jeu (assigné via le champ script de l'objet ou de l'acteur) est un script local. Les scripts locaux ne sont actifs que si la cellule est chargée – la cellule d'intérieur courante, ou la cellule d'extérieurs et toutes ses voisines directes. Quand un objet est en dehors de cette zone, le script ne s'exécute pas mais les variables locales sont sauvegardées.

Scripts globaux

Tout script qui n'est pas attaché à un quelconque objet est un script global, et par défaut n'est pas exécuté jusqu'à ce que vous l'appeliez (voir en-dessous). Comme le script n'est pas attaché à un objet précis, vous devez toujours spécifier les objets sur lesquels vous appliquez une fonction : la ligne suivante marcherait pour un script local attaché à un PNJ :

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

Vous devrez indiquer de quel PNJ il s'agit pour un script global :

```
"ID_PNJ" -> AITravel 1150, 8899, 1110  ; ID_PNJ est l'identifiant unique
                                       ; qui renvoie à un objet de l'éditeur
```

Les scripts globaux sont continuellement actifs une fois qu'ils ont été activés et jusqu'à ce qu'ils soient manuellement stoppés. Ainsi, une fois activés, ils seront exécutés à chaque frame comme c'est le cas pour les scripts locaux. C'est pourquoi il faut les utiliser avec précaution car des scripts globaux en trop grand nombre ou trop compliqués peuvent sérieusement dégrader les performances du jeu.
La commande pour démarrer un script non-actif est :

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

Avec Tribunal et Bloodmoon, on a la possibilité de démarrer automatiquement un script au lancement du jeu. Dans le TESCS, dans le menu ***Gameplay/Edit Start Scripts***, on peut ajouter n'importe quel script à la liste de ceux qui sont lancés automatiquement.

La fonction qui met fin à un script global est :

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

Il est possible d'utiliser la fonction *StartScript* afin de lancer un script lié à un objet ou un acteur. Ils sont appelés scripts ciblés ("*targeted scripts*").

```
"ID_objet" -> StartScript "ID_script"
```

Ces scripts ressemblent à la fois à des scripts locaux (les fonctions appelées feront toujours référence à l'objet ou l'Acteur pointé) et à la fois à des scripts globaux (dans le sens où ils tournent continuellement et ils peuvent être stoppés avec *StopScript*).

Note: voir la section Trucs et Astuces pour le cas des "*targeted scripts*".

Syntaxe

Certaines caractéristiques du TESCS doivent être expliquées avant d'aller plus loin.

Commencer et terminer des scripts

```
Begin ID_Script
End
End ID_Script
```

Chaque script doit commencer par *Begin* et se terminer par *End*. Le nom qui suit le *Begin* sera l'identifiant du script (regardez les autres scripts ou dans la fenêtre propriété de certains objets). Un script peut très bien démarrer par des commentaires, mais la première ligne de code doit être "Begin xxxxxxxxx".
Comme pour tous les objets, il est recommandé de donner à vos scripts un nom unique. Pour ma part, j'utilise GBG_Nom_du_script. Vous vous assurez ainsi une identification aisée, tous vos scripts sont regroupés en un bloc dans la liste et il y a peut de chances que d'autres mods utilisent le même noms. Nommer les scripts en commençant par un underscore (_Nom_du_script) n'est pas recommandé.

Syntaxe générale pour les fonctions:

Les fonctions ne sont pas sensibles à la casse du texte, mais en respectant la convention de Bethesda (c'est-à-dire *GetSpellEffects* à la place de *getspelleffects*), les scripts sont plus faciles à lire ; il est donc recommandé d'adopter cette convention.

```
"ID_objet" -> Fonction, [paramètres]
```

La "flèche" indique l'objet sur lequel est appliqué une fonction. "ID_Objet" est l'identifiant unique donné à chaque objet de l'éditeur (le premier champ de l'***object window*** dans l'éditeur). On a besoin de l'identifiant (ID) et non pas du nom (*Name*) ! Si vous appelez une fonction sans lui donner un identifiant d'objet, la fonction sera appliquée à l'objet par défaut, celui auquel le script est attaché.
Il est possible de référencer un autre objet en tant que paramètre :

```
"ID_objet1" -> Function, "ID_objet2"
```

Note: les fonctions avec la "flèche" ne compileront que si l'objet a déjà été placé dans le monde via l"éditeur (avec au moins une référence), sans cela le script ne compilera pas.
Pour certaines fonctions, utilisez la flèche n'a aucun sens et peut même provoquer des erreurs, notamment pour les fonctions s'appliquant par défaut au joueur (*GetDetected*, *GetPCRank*, etc.)
Si vous référencez des objets non uniques avec la "flèche", la fonction ne s'appliquera qu'à la première instance de cet objet! Donc utiliser cela dans un script :

```
"cliff racer" -> SetHealth, 0
```

n'aura pas l'effet désiré, ça ne tuera qu'un seul de ces piafs ennuyeux. Cependant attaché un script tel que celui-ci à une créature

```
SetHealth, 0
```

fonctionnera, car chaque instance de l'objet *cliff racer* exécutera le script en appliquant la fonction à lui-même.
Des fonctions ne font référence qu'au joueur ou même à aucun objet du tout, par conséquent on ne doit pas utiliser la "flèche". Par exemple :

```
If ( GetPCRank == 0 )
If ( CellChanged == 0 )
FadeOut, 2
```

Syntaxe générale : Virgules, parenthèses et espaces

Le TES Script n'est pas trop regardant quant à la syntaxe. La casse du texte n'est pas prise en compte, les virgules peuvent être omises et les espaces sont le plus souvent ignorés. Néanmoins, je conseillerais d'appliquer les principes suivants :

- Utiliser des virgules pour séparer les paramètres
- Si l'identifiant contient un espace, vous devez utiliser les guillemets : "ID Objet" ; il est préférable d'éviter les espaces : ID_Objet
- Prenez l'habitude de **toujours** laisser un espace après les parenthèses et les opérateurs ; il semble que cela cause parfois des problèmes si vous ne le faites pas : if (variable == 1), et pas : if (variable==1)
  Bien que cela ne pose pas de problèmes la plupart du temps, cela crée des erreurs bizarres et pratiquement indétectables ; il vaut mieux s'habituer à toujours laisser un espace.
- (Selon Dinkum Thinkum:) Les identifiants d'objets ou les variables commençant par un souligné (underscore) donnent des résultats irréguliers dans l'éditeur : à certains endroits, cela marchera très bien tandis qu'à d'autres, l'éditeur ne les tolèrera pas (d'habitude avec des messages d'erreurs qui n'ont rien à voir avec le problème actuel. Par exemple :
  ```
  PlaceItemCell, "_dt_Racial_ClothierNord", "Vivec, Agrippine Hérennia, tailleur", -348, 48, -221, 0
  ```
  marche correctement, mais :
  ```
  _dt_Racial_ClothierNord -> PositionCell, -348, 48, -221, 0, "Vivec, Agrippine Hérennia, tailleur"
  ```
  cause des messages d'erreurs.
  Au moins un des guides ou tutoriaux sur le modding recommande d'utiliser des underscores au début des identifiants d'objets afin qu'ils soient placés au début des listes et soient ainsi plus faciles à retrouver. J'ai suivi cet avis, et beaucoup de gens l'ont fait (en regardant les mods d'autres personnes). Même chose pour les variables : à certains endroits, cela fonctionne et à d'autres non. Ma politique actuelle est de ne PAS utiliser des underscores au début de quoi que ce soit dans le Construction Set.

Commentaires

Les commentaires commencent par un point-virgule ; ils peuvent être placés sur leur propre ligne ou à la suite d'une ligne de code

```
; passe en mode discrétion
Fargoth->ForceSneak
Fargoth->AiTravel -11468.595,-71511.531,173.728     ;va vers la souche
```

Indentation / utilisation des tabulations

Dans votre propre intérêt, utilisez les tabulations dans les constructions if-elseif – c'est plus facile de garder une trace d'eux et donc de ne pas oublier un *endif* à la fin. Dans la section Trucs et Astuces, vous trouverez un lien vers un mode d'édition externe EMACS pour le TES script qui vous dispensera d'une indentation manuelle.

```
If ( variable1 )
       If ( variable2 )
              [faire quelque chose]
       endif
endif
```

vaut mieux que

```
If ( variable1 )
If ( variable2 )
[faire quelque chose]
endif
endif
```

Variables

Types de variables

Il y a trois types de variables dans le langage de script du TESCS : short, long and float. D'après le manuel, elles couvrent les plages de données suivantes :

Short	-32,768 to 32,767 (entiers signés)
Long	-2,147,483,648 to 2,147,483,647 (entiers signés étendus)
Float	3.4E +/- 38 (flottant, 7 décimales)

Apparemment les limites données pour le type Long ne sont pas tout à fait correctes, dans le TESCS, on peut assigner une valeur maximum de 2147483520 (Forum info / Argent). En théorie, il y a aussi des variables de type string (chaîne de caractères) mais à ma connaissance elles ne sont pas implantées. Malheureusement, il n'y a pas non plus de type pour stocker les identifiants d'objets, ce qui limite la puissance du langage dans une certaine mesure.
Les variables peut être ou *locales* (valides dans le script où elles ont été déclarées) ou *globales* (valides dans n'importe quel script).
Note: une variable globale de type long est en fait un float! Dans un script local, un long est codé sur 32 bits. Mais utilisé en tant que variable globale, le nombre de bits utilisé pour le codage chute à 24 ; tant qu'il existe, un long global *est* un float. Mental Elf a découvert cela en essayant d'utiliser un long global en tant que 32 flags ("bit packing") (forum info / mental elf).

Variables locales

Elles doivent être déclarées dans le script :

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

Les variables locales n'appartiennent qu'à une instance spécifique d'un script local. Cela signifie qu'il n'y a pas d'interférence entre des variables locales d'un même script attaché à deux objets différents. Les noms de vos variables ne regardent que vous mais ils doivent commencer par une lettre ; évitez d'utiliser les noms des fonctions (cela provoquera des erreurs lors de l'exécution) et les caractères réservés (-, +, /, *, =, ",),(, etc.) qui donneront des erreurs de compilation. Par exemple "variable-1" n'est pas un nom de variable valide. Les underscores (ma_variable) sont ok. Le point a également une signification (voir "*Accéder à des variables d'autres objets ou scripts*" ci-dessous).

Variables globales

Pour déclarer une variable globale, allez dans le menu ***Gameplay*** et sélectionnez ***Globals***. Cliquez sur le bouton "new", nommez votre variable, choisissez son type et entrez sa valeur de départ, si cela est nécessaire. Par défaut, ce sera 0. Les variables globales sont très utiles pour les quêtes complexes pour lesquelles vous avez besoin de garder les traces des actions du joueur. Elles sont aussi un moyen simple de partager les informations entre différents scripts.
Note: si vous déclarez une variable local possédant le même nom qu'une variable globale, celle-ci sera invisible pour ce script. Ne déclarez PAS une variable globale comme une locale !

Accéder à des variables d'autres objets ou scripts

```
Set … to
```

Si un objet unique possède un script local vous pouvez mettre à jour ses variables depuis l'extérieur du script de la façon suivante :

```
Set Mon_Objet.variable to 100
```
ou
```
Set Mon_Objet.variable to variable_locale
```

Cette méthode change une variable locale dans le script de l'objet. Un script doit être attaché à cet objet pour qu'elle soit valide.
Note: le système regarde le premier objet de la base de données, c'est pourquoi vous devez ne référencer que les objets uniques (qui existe en un seul exemplaire).

L'inverse n'est pas possible :

```
Set variable_locale to Mon_Objet.variable  ; ne marche pas !
```

Utilisez une variable globale pour échanger l'information dans ce sens ou affectez variable_locale à partir de l'**autre** script avec la syntaxe précédente.

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

fonctionne apparemment.
De plus, j'ai récemment réalisé que la syntaxe suivante fonctionne aussi pour les scripts globaux :

```
set nom_du_script_global.variable to 1
```

C'est utile pour éviter de recourir à plus de variables globales que nécessaire ou pour débugger les scripts globaux à la console.

Utilisation des variables dans les fonctions

Malheureusement seules certaines fonctions du langage acceptent les variables en paramètres, ce qui engendre pas mal de limites. Le type des arguments qu'acceptent les fonctions est indiqué dans les sections suivantes.

Note: on peut trouver une alternative en utilisant une boucle while pour certaines fonctions (les fonctions Get-nomf lorsque leur fonction inverse Set-nomf existe) : voir ci-dessous.

Opérations mathématiques

Vous pouvez utiliser les opérandes standards dans les commandes set (et probablement ailleurs, mais je n'ai pas essayé) pour faire des calculs dans vos scripts.

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

La syntaxe est la suivante :

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

A la place de variables, on peut bien sûr utiliser les valeurs elles-mêmes. Je suppose que les règles mathématiques s'appliquent (priorité des opérateurs…). Vous pouvez aussi vous servir des parenthèses, selon ces mêmes règles :

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

Un avertissement : les avis divergent sur l'utilisation de plusieurs opérateurs sur une seule ligne. Certaines personnes ont eu des problèmes avec cela, j'ai moi-même réussi à utiliser quatre opérateurs et variables en une seule ligne. Il semblerait qu'il y ait un problème avec les très longues additions (additionner plus de 20 variables en une ligne de code) ; un mod peut faire crasher le jeu au chargement. Si cela arrive, divisez votre calcul en plusieurs lignes.

Il n'y a pas beaucoup de fonctions dédiées aux mathématiques dans les scripts de Morrowind. Il y a la fonction '*Random*' et Tribunal a ajouté la fonction '*GetSquareRoot*' (voir plus loin). Si vous avez besoin de fonctions plus complexes, téléchargez le Math Mod de Soralis (dispo sur Morrowind Summit). C'est un ensemble de scripts qui vous permet de faire des calculs complexes. Voici un petit aperçu du Readme pour vous faire une idée :

" Ce module ajoute la possibilité d'utiliser des fonctions mathématiques dans les scripts de Morrowind. Voici les scripts qui sont ajoutés : "

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

Malheureusement l'exécution de beaucoup de ces fonctions est lente et elles ne sont pas appropriées pour des calculs en temps réel. Pour sinus et cosinus en particulier, voyez le script de JDGBOLT dans la section Trucs et Astuces qui utilise des valeurs pré-calculées qui rendent l'exécution plus rapide.

Tester des conditions

Utilisation des conditionnelles if… elseif (si...sinon)

```
If (condition)
Elseif (condition)
Else
Endif
```

Une grande partie des scripts repose sur l'utilisation des conditionnelles *if… elseif* et de leurs variations. Il est très important de les comprendre complètement, ainsi que les conditions que l'on peut utiliser à l'intérieur pour tester les états du jeu afin de déclencher des évènements.
En général, une condition est "vraie " quand elle a pour valeur 1 (ou simplement une valeur différente de 0, ce qui signifie que -1 est également "vrai" pour le langage) et "faux" lorsque la valeur est 0. Ainsi certaines fonctions du jeu renvoient "vrai" sous certaines conditions. Par exemple, regardez la fonction *GetAIPackageDone* ci-dessous. Elle renvoie "vrai" (= 1) pour une frame lorsque une *AIPackage* a terminé. Les lignes de codes comprises entre une instruction if et une instruction *endif* (appelées bloc d'instructions) ne seront exécutées que si la condition est évaluée à vrai.
Ainsi :

```
If ( GetAIPackageDone )
        ;[bloc d'instructions]
endif
```

le bloc d'instructions ne sera exécuté que si la fonction *GetAIPackageDone* renvoie la valeur 1.

Plus qu'une simple valeur, une condition peut être un test plus explicite. Des conditionnelles telles que "A égale B" ou "A est plus grand que B" seront évaluées et c'est l'expression toute entière qui sera vraie (égale à 1) lorsque la conditionnelle sera vérifiée. Par exemple :

```
if ( GetAIPackageDone == 1 )
        ;[bloc d'instructions]
endif
```

est équivalent à l'exemple précédent. La seconde syntaxe teste une condition et renvoie "vrai" si la condition est vérifiée. Vous pouvez effectuer vos tests avec les opérateurs suivants :

Opération de comparaison	Opérateur
"Est égale":	==
"Est différent"	!=
"Est plus grand que"	>
"Est plus petit que"	<
"Est supérieur ou égale"	>=
"Est inférieur ou égale"	<=

Si vous avez plusieurs blocs indépendants, chacun d'eux sera évalué séparément. Par exemple, si les blocs suivants se trouvent dans votre script :

```
if ( timer >= 2 )
        ;[Bloc A]
endif
if ( timer <= 3 )
        ;[Bloc B]
endif
```

Lorsque timer est compris entre 2 et 3, les deux conditionnelles seront vraies et le code contenu dans les blocs sera exécuté.
Pour être certain que les deux conditions sont remplies en même temps, les blocs if peuvent être imbriqués :

```
if ( timer >= 2 )
        if ( controlvar == 0 )
                ;[faire quelque chose]
        endif
endif
```

le code du bloc intérieur ne sera exécuté que si les deux conditions (timer >= 2 and controlvar == 0) sont remplies.

Pour des constructions plus élaborées, vous pouvez utiliser *Else* et *Elseif*. *Elseif* teste une condition différente si la conditionnelle précédente a échouée (mais pas si elle a été remplie)

```
if ( timer >= 2 )
        ;[Bloc A, faire quelque chose]
elseif ( timer <= 3 )
        ; [Bloc B, faire quelque chose d'autre]
endif
```

à l'inverse de l'exemple précédent, les deux conditionnelles peuvent ne pas être vraies en même temps. Ou timer est >= 2, et le bloc A est exécuté, ou timer n'est pas >= 2 mais <= 3, ce qui signifie que timer est < 2, et le bloc B est exécuté. Dans tous les autres cas, aucun bloc n'est exécuté. Vous pouvez avoir plusieurs *elseif* les uns à la suite des autres :

```
if ( compteur == 1 )
        ;[Bloc A,faire quelque chose]
elseif ( compteur == 2 )
        ; [Bloc B,faire quelque chose d'autre]
elseif ( compteur == 3 )
        ; [Bloc C, faire quelque chose d'autre]
endif
```

Un *else* crée une "condition par défaut". Le code suivant le *else* sera exécuté si toutes les autres conditions précédentes sont fausses.

```
If ( toto == 1 )
        ;[Bloc A]
elseif ( toto == 2 )
        ;[Bloc B]
else
        ;[Bloc C: faire quelque chose de complètement différent]
endif
```

dans cet exemple, le bloc C sera exécuté si toto n'est ni 1 ni à 2.

Note: D'après mon expérience, il est plus sûr d'utiliser *elseif* à la place de plusieurs if lorsque vous testez plusieurs état d'une même variable.
Il y a un maximum pour le nombre total de conditionnelles *if-elseif* qui peuvent être traitées dans un script. *Else* compte aussi pour une conditionnelle séparé. D'après les dernières infos, la limite est proche de 127 (et pas les 256 mentionnés dans les versions précédentes).
La distance trop importante entre un "if" et un "*endif*" peut aussi provoquer des erreurs. La limite se situe apparemment entre 120 et 240 lignes (Forum info / MentalElf).

While conditions

```
While ( condition )
; instructions à exécuter
EndWhile
```

La commande while est différente du if car ses instructions sont répétées à l'intérieur d'une frame jusqu'à ce que la condition ne soit plus vraie. Un exemple pour mieux comprendre :

```
Short valeur_voulue

SetStrength 0
while( GetStrength < valeur_voulue ) ; valeur non littérale à vérifier
  modStrength 1
endwhile
```

Ceci positionnera la force du joueur à la valeur contenue dans la variable valeur_voulue après une frame. Par contre le script qui suit aura besoin d'un nombre indéterminé de frames pour atteindre ce résultat car la conditionnelle if n'est appelé qu'une fois par frame :

```
if(getStrength < valeur_voulue) ; valeur non littérale à vérifier
  modStrength 1
endif
```

D'un autre côté, le premier exemple peut "geler" l'écran (si la valeur est très haute) tandis que le second ne le fera pas.
Notez que ceci est une solution au problème des fonctions n'acceptant pas les valeurs non-littérales (les variables) en arguments.

Opérations booléennes dans un script

Malheureusement il n'existe aucun opérateur booléen (AND/ET, OR/OU, NOT/NON, XOR/OU EXCLUSIF, …) dans le langage. Vous devrez les construire vous-même en utilisant des structures if…elseif.

A la place de ET :

```
if ( variable1 AND variable2 ); n'existe pas
        [faire quelque chose]
endif
```

vous devrez faire :

```
If ( variable1 )
        If ( variable2 )
                [faire quelque chose]
        endif
endif
```

Pour la construction du OU

```
if ( variable1 OR variable2 )
        [instrucion_i]
endif
```

vous pouvez utilisez les constructions elseif :

```
If ( variable1 )
        [instruction_i]
elseif ( variable2 )
        [instruction_i]
endif
```

Liste des fonctions de scripts

Explication du format

Tout d'abord, je donnerai le nom de la fonction ainsi que les arguments qu'elle prend en paramètres :

[no fix] Code "string", arg_enum, arg_float, [optionnel] (renvoie short)

[no fix] indique que la fonction s'utilise sans "flèche" ce qui signifie qu'elle ne peut être appelée par un acteur spécifique. Les fonctions présentées sans cette étiquette peuvent être appelée par un acteur, un objet ou par les deux.
Code: Nom de la fonction
Les arguments de la fonction : "string" indique une chaîne de caractères, comme un identifiant d'objet. arg_enum indique une valeur littérale (variables non prises en compte), arg_float indique une variable du type spécifié (dans ce cas, float). Les crochets [] indiquent que le paramètre est optionnel. (renvoie short) ou (renvoie float) indique le type de retour de la fonction. J'utiliserai la désignation (renvoie Booléen/short) pour indiquer que la fonction renvoie 0 ou 1 (une variable booléenne, bien que ce soit strictement parlant un float).

Les exemples d'utilisation sont en italiques et sont indentés :

Code "ID", var_enum, var_float

Les exemples de scripts sont placés dans des fenêtres de ce genre :

```
Begin script

[Fonctions du script]

End script
```

A partir de la 8ème édition, les fonctions ajoutées par Tribunal et Bloodmoon ont été déplacées dans leur section correspondante. Elles sont marquées par :

pour les fonctions de Tribunal et


pour les fonctions de Bloodmoon.

Pour utiliser ces fonctions, l'extension correspondante doit être installée (mais pas nécessairement active). Bloodmoon (et l'édition GOTY) incorpore toutes les fonctions de Tribunal.

Travailler avec des objets

Travailler avec les objets de l'inventaire

Ajouter et supprimer des objets de l'inventaire

```
AddItem, "ID_objet", nombre_enum
RemoveItem, "ID_objet", nombre_enum

        Actor -> AddItem, "item_ID", 1
        Container -> RemoveItem, "itme_ID", 5
```

Ces fonctions sont assez simples, ajouter ou supprimer des objets de l'inventaire du joueur ou de n'importe quel autre inventaire, ce qui inclut les containers. Un appel à *RemoveItem* enlèvera l'objet de l'inventaire, il "disparaîtra".
Notes: enlever des objets non présents dans l'inventaire ne plante pas le jeu mais la fonction '*RemoveItem*' soustraira le poids de l'objet à l'encombrement du personnage, MEME SI l'objet n'est PAS dans l'inventaire de ce personnage. Si un script utilise '*RemoveItem*' pour supprimer un objet pesant 4 lb. (livres) et que le personnage ne possède pas, l'encombrement de celui-ci diminuera de 4 lbs. Pour régler ce bug, il faut systématiquement vérifier la présence d'un objet dans l'inventaire avant d'utiliser la fonction '*RemoveItem*' pour le supprimer (Merci à DinkumThinkum pour l'info).
Faites aussi attention si vous voulez enlever un objet en train d'exécuter un script, le jeu plantera ; voir le script d'exemple.
La solution pour régler ce problème est d'utiliser un script séparé, global ou local, pour supprimer l'objet. Cependant si le joueur possède **deux exemplaires ou plus** d'un objet avec un script attaché, l'utilisation de *RemoveItem* sur l'identifiant de l'objet rendra les données d'un des objets restants corrompues (par exemple concernant la santé ou le nombre). S'équiper ou utiliser l'objet corrompu peu causer un plantage (Forum info, DinkumThinkum).

Pour éviter ces deux bugs :

1. Si l'objet n'a pas de script attaché, utilisez *GetItemCount* pour être sûr que le joueur en possède *au moins* 1.

2. Si l'objet a un script attaché, utilisez *GetItemCount* pour être sûr que le joueur en possède *exactement* 1 ou soyez certain que l'objet est unique.

La fonction *Drop* n'a pas ce bug. Cependant elle peut causer des dédoublements, si elle est utilisée avec *OnPCEquip* / *SkipEquip*, qui peuvent être réglés en ajoutant et supprimant n'importe quel objet comme cela est décrit pour ces fonctions.

Notes sur l'utilisation de *AddItem/RemoveItem* dans les dialogues : apparemment **ces fonctions acceptent les variables globales mais seulement dans les résultats de dialogue,** et seulement si vous n'avez pas affecté une valeur dans le même résultat de dialogue (Forum info / Argent ; d'après Argent, la quantité maximum qu'il a réussi à ajouter via une variable grâce à *AddItem* est de 65534 (en utilisant une variable de type long=2147483520)). De plus soyez sûr que la variable en question ne change pas de valeur dans le même résultat de dialogue. Disons que var_a vaut 3 au moment où la ligne est sélectionnée dans le dialogue. Si le champ résultat ressemble à ça :

```
set var_a to var_a + 10 ;
AddItem gold_001, var_a
```

trois drakes seront ajoutés à l'inventaire du PNJ et pas treize. Si vous deviez en avoir besoin, placez tous vos calculs dans un seul champ résultat d'une réponse de dialogue, utilisez la commande "*choice*" et appelez la fonction *AddItem* dans la réponse suivante. De plus, je n'ai pas réussi à ajouter des quantités supérieures à 32767 (2^15). (testé sur le Morrowind de base sans aucune extension) (Forum info / Kir).

Notes sur l'utilisation sur des containers : on a signalé que seul le premier appel à *AddItem* fonctionne sur un container; après cela, le joueur doit accéder manuellement au container dans le jeu (par exemple en ajoutant un objet) avant qu'un nouvel appel à *AddItem* ne fonctionne (information non vérifiée).
Les containers ne peuvent plus être affectés par *AddItem* une fois qu'ils ont été vidés. Si vous voulez ajouté des choses après qu'un container ait été vidé, la seule chose à faire est de supprimer l'objet et d'en placer un nouveau, vide, à la place : la fonction marchera tant qu'il reste au moins un objet dedans. Malheureusement la seule façon de vérifier qu'un container est bien vide est de tester indépendamment la présence de chaque objet du jeu (Forum info / ThePal).

Exemple de script:

On a longtemps discuté de ce script sur les forums (désolé, je ne me souviens plus qui l'a écrit en premier). Il est supposé demander au joueur s'il veut recycler un objet lorsqu'il s'en équipe et remplacer cet objet par la version "recyclée".

```
Begin scr_thing

short bouton
short OnPcEquip
short etat

if ( MenuMode == 1 )
       return
endif

if ( OnPCEquip == 1 )      ; quand on s'équipe avec l'objet
       set etat to 1
       set OnPCEquip to 0     ; ne faire ceci que lorsqu'on s'équipe d'un objet
endif
if ( etat ==1 )
       MessageBox "recycler?" , "oui", "non"
       set etat to 2
elseif ( etat == 2 )
       set bouton to GetButtonPressed
       if ( bouton == 0 )
              PlaySound "mysticism cast"
              player->RemoveItem "item_a", 1 ; cette ligne plante le jeu !!!
              player->AddItem "item_b", 1
              set etat to 0
       elseif ( bouton == 1 )
              set etat to 0        ; une fois fait, on revient à l'état initial
       endif
endif

end
```

Il marche très bien sans la ligne appelant la fonction *RemoveItem*, mais avec celle-ci le jeu risque de planter. En effet le script était attaché à l'objet_a, le script disparaît avec la suppression de l'objet ce qui provoque le crash du jeu. Il faudra donc utiliser un script global pour mettre en œuvre cette idée.

Lâcher des objets au sol

```
Drop, "ID_objet", nombre_enum
```

```
"ID_Acteur" -> Drop, "ID_objet", 1
```

La fonction est supposée faire tomber l'objet "aux pieds de l'acteur appelant". Il semblerait que cela fonctionne correctement uniquement pour le personnage joueur, l'objet tombant à ses pieds. Quand j'ai voulu l'utiliser sur un PNJ, l'objet a bien été enlevé de l'inventaire du PNJ, mais il apparaît aux pieds du joueur.
Note intéressante : s'ils ont réellement l'objet, ils le laisseront tomber, sans modifier les charges et la condition. S'ils ne l'ont pas, une nouvelle instance de cet objet sera créée. Si le joueur lâche un objet qu'il ne possède pas, son encombrement sera réduit du poids de l'objet – même bug que pour *RemoveItem* (Forum info / DinkumThinkum).
Le script SlaveScript de Bethesda contient **des exemples** d'utilisation.

Contrôler les activités dans l'inventaire : ajouter, déposer des objets, utiliser les gemmes spirituelles

[no fix] OnPCAdd **(est une variable locale de type short)**

```
Short OnPCAdd
If ( OnPCAdd == 1 )
```

Cette variable passe à 1 lorsque le PJ ajoute l'objet à son inventaire. Doit être réinitialisée manuellement pour les utilisations futures (set OnPCAdd to 0).
Exemple: voici une partie du script attaché à l'anneau de Fargoth qui donne accès au menu Magie durant la phase de création de personnage :

```
if ( OnPCAdd == 1 ); le joueur a ajouté l'objet à son inventaire
        if ( State == 0 )
                EnableMagicMenu
                MessageBox "Vous avez maintenant accès au menu Magie, dans lequel vous pouvez
voir tous vos pouvoirs, sorts et objets magiques." "Ok"
                set state to 10
                return
        endif
endif
```

[no fix] OnPCDrop **(est une variable locale de type short)**

```
Short OnPCDrop
If ( OnPCDrop == 1 )
```

Cette variable passe à 1 lorsque le PJ dépose l'objet. Doit être réinitialisée pour les utilisations futures (set OnPCDrop to 0).

[no fix] OnPCSoulGemUse **(est une variable locale de type short)**

```
Short OnPCSoulGemUse
If (OnPCSoulGemUse == 1 )
```

Celle-ci passe à 1 lorsque l'objet appelant est une gemme spirituelle et qu'elle a été utilisée ou bien pour recharger ou bien fabriquer un objet. Doit être réinitialisée pour les utilisations futures (set OnPCSoulGemUse to 0).
Exemple : voici comment l'étoile d'Azura peut être réutilisée plusieurs fois :

```
begin AzurasStarScript
; pour l'Etoile d'Azura. Une gemme spirituelle utilisable indéfiniment. Plutôt chouette. Les
enfant l'adorent.

short OnPCSoulGemUse

;ils l'ont utilisé, donc on leur en donne une nouvelle
if ( OnPCSoulGemUse == 1 )
        Player->additem, "Misc_soulgem_Azura" 1
endif
end
```

Forcer le port d'un objet

```
Equip, "ID_objet"

        "ID_Acteur" -> Equip, "p_restore_health_q"
```

(voir aussi la fonction *OnPCEquip* en-dessous)
Avant Tribunal: Partiellement buggée. Cela aurait pu être une fonction des plus utiles. Malheureusement, la plupart des utilisations potentielles ne fonctionne pas : on ne peut PAS tout équiper automatiquement sur le joueur. On ne peut PAS forcer les Acteurs à s'équiper d'armes ou d'armures (cela est complètement déterminé par leurs compétences/talents). On ne peut PAS créer des objets impossibles à retirer, des armures maudites par exemple, etc. On PEUT faire ingurgiter des potions aux Acteurs, d'après ce que j'ai entendu. Cette fonction a été **corrigée avec Tribunal**. Vous pouvez maintenant forcer les Acteurs à s'équiper d'armures, d'armes et de vêtements. Vous POUVEZ maintenant faire toutes les opérations décrites ci-dessus☺. Loué soit Bethesda.

Note:
La fonction *Equip* permet d'équiper quelqu'un avec un objet qu'il n'a pas, il sera ajouté à son inventaire. Cependant, si vous faites comme ça, les scripts sur ces objets ne s'exécuteront pas. Vous devez donc utiliser *AddItem* d'abord, puis *Equip*, même si *Equip* ajoute les objets. Le script démarrera si vous enlevez l'objet du personnage ou de son inventaire et le laissez tomber. Mais tant qu'il est dans l'inventaire, le script ne démarre pas (Forum info, ThePal).

Script de démonstration : ce script (Tribunal requis) maudit un objet (un gourdin en chitine) afin qu'on ne puisse plus le déséquiper. Même en utilisant les raccourcis. Débile! Maintenant il vous faudra vous battre avec ce gourdin pendant le restant de vos jours, ce qui ne devrait plus être très long ☺ Le joueur peut toujours se servir de la magie.

```
Begin cursed_item
objet_maudit

short etat
short OnPCEquip


if ( OnPCEquip == 0 ) ; l'objet n'est pas équipé
        if ( etat == 0 ); si le gourdin n'a encore jamais été équipé, ne rien faire.
                return
        else
                Player -> Equip, "gourdin_maudit" ; rééquipe l'objet!
                MessageBox "C'est un objet maudit, il ne veut pas quitter votre main" ;raille
le joueur
        endif
else
        if ( etat == 0 ) ; c'est la première fois qu'on s'en équipe. Le piège se referme
                set etat to 1
        endif
endif

End
```

Détecter si un objet a été équipé

```
[no fix] OnPCEquip          (est une variable locale de type short)

        Short OnPCEquip
        If ( OnPCEquip == 1 )
```

Le PJ s'est équipé de l'objet (demeure vrai tant que l'objet est équipé)
Cette variable (**qui doit être déclarée !**) passe à 1 si le joueur s'équipe de l'objet appelant. Elle reste à "vrai" tant que l'objet reste équipé, mais revient à 0 si on s'en déséquipe. Dans certains cas, vous voudrez probablement la repasser à 0 manuellement :

```
if ( OnPCEquip == 1 )       ; lorsque l'objet est équipé
       [faire quelque chose]
       set OnPCEquip to 0     ; ne le faire qu'une fois par évènement equip…
endif
```

La prochaine fois que vous enlèverez l'objet pour vous en rééquipez, les instructions dans [faire quelque chose] seront exécutées une nouvelle fois. Vous pouvez aussi vous en servir comme une variable d'état quand un effet s'exécutera. Notez que cela fonctionne en mode menu :

```
If (MenuMode ==1)
       if ( OnPCEquip == 1 )       ; lorsque l'objet est équipé
              [faire quelque chose]
              set OnPCEquip to 0     ; ne le faire qu'une fois par évènement equip…
       endif
endif
```

Ce script s'exécutera lorsque vous serez dans l'inventaire, dès que l'objet sera équipé tandis que le script suivant ne sera exécuté qu'après avoir quitté le menu :

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

if ( OnPCEquip == 1 )       ; lorsque l'objet est équipé
       [faire quelque chose]
       set OnPCEquip to 0     ; ne le faire qu'une fois par évènement equip…
endif
```

Vous pouvez trouver un autre **Script de démonstration** avec la fonction *Equip* au-dessus.

Notes: j'ai testé *OnPCEquip* avec les types d'objets suivants :
Vêtements (*Clothing*)
Armure (*Armor*)
Armes (*Weapons*)
Livres/Parchemins (*Books/Scrolls*) : voir la section Trucs et Astuces pour une utilisation correcte
Objets divers (*Miscellaneous items*)
Lumières utilisables (*Usable lights*)
Sondes (*Probes*)

Les Potions et les Ingrédients ne positionnent *OnPCEquip* que lorsque vous utilisez *SkipEquip* au même moment, sans quoi l'objet est apparemment 'détruit' avant que la fonction ne l'enregistre ! Les outils de réparation souffrent aussi de ce problème et, encore plus bizarre, les alambics ont des effets inverses : ils ne déclenchent *OnPCEquip* que lorsque *PCSkipEquip* n'est PAS positionné (Forum info / ManaUser).
Apparemment les livres (et peut-être d'autres types objets qui possèdent un comportement bizarre ?) placent *SkipEquip* à 1 au lieu de *OnPCEquip* ! A ce sujet, voir la section Trucs et Astuces.

Désactiver la possibilité d'équiper un objet

[no fix] PCSkipEquip (est une variable de type short)

```
       Short PCSkipEquip
       Set PcSkipEquip to 1
```

Placez la variable à 1 pour rendre un objet inéquipable. Très utile pour faire afficher des messages concernant la rupture d'un sceau d'un livre ou d'un objet équivalent. Examinez le script *SealedTreasuryReport* dans l'éditeur pour avoir une vue d'ensemble des possibilités. Elle est également très bien pour utiliser les objets de type vêtements comme des déclencheurs de scripts en la combinant avec la fonction *OnPCEquip* (voyez mon mod

d'escalade - *climbing mod* – par exemple, l'équipement est en fait un gant, mais qu'il est impossible d'équiper).
Note: apparemment équiper un livre de l'inventaire fait passer cette variable à 1 (au lieu de placer *OnPCEquip* à 1, comme ça le devrait. Voir la section Trucs et Astuces)
Il existe un bug en utilisant ceci : l'objet qui a la fonction *SkipEquip* se duplique. J'ai pu le constater en me servant des raccourcis clavier et en m'équipant de l'objet à partir de l'inventaire. Pour contourner ce problème, réinitialisez l'inventaire en ajoutant un objet quelconque puis supprimez-le (dans la section du script qui effectue la vérification de *OnPCEquip*). Ne supprimez pas cet objet quelconque par un script qui lui serait attaché : cela provoquerait un crash (voir la fonction *RemoveItem*). Si vous avez beaucoup d'objets *SkipEquip*, faites appeler un script global (*StartScript*) par votre objet, qui ajouterait et supprimerait l'objet, par exemple :

```
Begin doubling_fix

Player -> Additem "Item ID", 1
Player -> RemoveItem "Item ID", 1

StopScript doubling_fix

End
```

Script de démonstration: voici un petit script fait pour un mod loup-garou ; il rend un objet non-équipable sous certaines conditions :

```
Begin non_equippable

; empêche les joueurs lycanthropes de s'équiper des objets de chasseurs de loups-garous pour
des raisons d'équilibre
; si le PC s'en équipe avant de devenir un loup-garou, il peut les porter jusqu'à ce qu'il les
enlève
; mais il ne pourra plus s'en rééquiper. Donc après la transformation, il ne pourra plus s'en
servir

short PCSkipEquip
short OnPCEquip

if ( PCWerewolf != 1); si le joueur n'est pas un loup-garou, il peut utiliser l'armure
        set PCSkipEquip to 0
        return
else
        set PCSkipEquip to 1
endif


if ( OnPCEquip == 1 )
        MessageBox " cet objet est enchanté avec des sorts maudissant les loup-garous. Vous ne
pouvez pas les porter !"
        set OnPCEquip to 0
endif

End
```

Vérifier la présence d'un objet dans l'inventaire

```
GetItemCount, "ID_objet"      (renvoie short)
        Short nombre_d_objets
        Set nombre_d_objets to ( "Mob_ID" -> GetItemCount, "ID_objet" )

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

Cette fonction vérifie l'inventaire de l'objet appelant et renvoie le nombre d'objets de type "ID_objet" qu'il possède.

Réparer des objets

```
[no fix] OnPCRepair           (est une variable de type short)


        Short OnPCRepair
        If ( OnPCRepair == 1 )
```

Une variable qui passe à 1 lorsque le PJ répare l'objet avec le script. Nécessite une réinitialisation manuelle.

```
RepairedOnMe, "ID_objet"              (renvoie Booléen/short)

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

Cette fonction renvoie 1 si l'objet appelant est réparé par un objet de type "ID_objet". ID_objet doit être du type "*Repair Item*" (outils de réparation) et l'objet appelant doit être une arme ou une armure.

```
OnRepair
```

La fonction similaire *OnRepair* est apparemment buggée. Elle devrait passer à 1 dès qu'une réparation est tentée sur l'objet : "renvoie vrai si l'objet appelant est réparé".

Informations sur les objets équipés / portés

```
GetWeaponType (renvoie short)

        If ( Player->GetWeaponType == 0 )
                ;le joueur utilise une lame courte

GetArmorType, armorPart_enum (renvoie short, de -1 à 2)

        If ( Player->GetArmorType, 0 == 2 )
                ;le joueur porte un casque lourd
```

Ces fonctions sont appelées sur un Acteur pour obtenir des informations sur son équipement. *GetWeaponType* renvoie le type de l'arme (voir Table 1.1) courante. *GetArmorType* renvoie la classe de l'armure (voir Table 1.3) du morceau qu'il porte actuellement. Les nombres correspondants aux morceaux de l'armure sont donnés par la Table 1.2. *HasItemEquipped* renvoie 1 si l'Acteur est équipé d'un objet donné et 0 sinon.

Types d'armes (Table 1.1):

Nom du type d'arme	Nombre correspondant
Sans armes	-1
Lame courte, une main	0
Lame longue, une main	1
Lame longue, 2 mains (proches)	2
Contendant, 1 main	3
Contendant, 2 mains (proches)	4
Contendant, 2 mains (éloignées)	5
Lance, 2 mains (éloignées)	6
Hache, une main	7
Hache, 2 mains (proches)	8
Arc	9
Arbalète	10
Arme de lancer	11
Flèche<?>	12
Carreau<?>	13

Morceaux d'armures (Table 1.2):

Nom du morceau de l'armure	Nombre correspondant
Casque	0
Cuirasse	1
Epaulière gauche	2
Epaulière droite	3
Jambières	4
Bottes	5
Gant gauche	6
Gant droit	7
Bouclier	8
Bracelet gauche	9
Bracelet droit	10

Types / classes d'armure (Table 1.3):

Nom du type d'armure	Nombre correspondant
Sans armure	-1
Armure légère	0
Armure Intermédiaire	1
Armure Lourde	2

HasItemEquipped "ID_objet" (renvoie short)

```
        If ( Player -> HasItemEquipped "chitin club" == 1 )
               ;mon pauvre ami !
```

Script de démonstration :

Quand ce script est placé sur un objet, activer une instance de cet objet lui fera des "dommages" basés sur la force et l'arme courante du PJ. Si cette arme est le "Rock Splitter", l'objet est totalement détruit en un seul coup. Lorsqu'il est totalement détruit, il explose en éclats à la figure du PJ à moins qu'il ne se soit équipé d'un bouclier ou d'un casque.

```
Begin breakme

float hitsleft     ;points de vie restants
float hitpercent   ;pourcentage de démolition
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 "Le Rock Splitter déchaîne sa toute puissance..."
      set hitsLeft to 0
   else
      MessageBox "Vous frappez le rocher avec votre arme courante..."
      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 "...et le rocher éclate envoyant des éclats acérés dans vos yeux."
            Player->ModHealth -50
         else
            MessageBox "...et le rocher explose, les éclats mortels ricochant sur votre
casque."
         Endif
      else
         MessageBox "...et le rocher explose, les éclats mortels ricochant sur votre
bouclier."
      Endif
```

```
   else
      set hitpercent to hitsleft / 100
      set hitpercent to 100 - hitpercent
      MessageBox "...et le rocher est éclaté à %.2f pour cent mais reste intact.", hitpercent
   endif

endif
```

Fonction UsedOnMe

```
UsedOnMe, "ID_objet"          (renvoie Booléen/short)
       if ( UsedOnMe, Misc_pot_redware_01 )
```

D'après le fichier d'aide :
"Renvoie vrai si "ID_objet" a été utilisé sur l'objet appelant. Utilisé pour les scripts où un objet A doit faire quelque chose si le joueur utilise un objet B sur A. "
A notre connaissance, cette fonction est buggée.

Bouger et faire tourner des objets

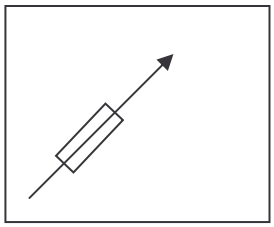
Les fonctions suivantes ne fonctionnent pas sur le joueur, les PNJs ou les monstres. Elles fonctionnent sur les objets statiques, les activateurs, les containers, les objets divers, etc. **Note:** les Acteurs (dont le PJ) ont tendance après un certain temps à tomber à travers des objets qui se déplacent. Cela peut être évité en désactivant et en activant rapidement l'objet se déplaçant à chaque frame – apparemment les informations de collisions sont mises à jour. Les Acteurs peuvent aussi profiter des effets de la lévitation ou de la chute ralentie pour réduire les probabilités de tomber à travers des objets. (pour plus de détails, voir la section Trucs et Astuces sur les '*ridable objects*' de MadMax).

Effectuer un déplacement le long de l'axe d'un objet

```
Move    axe(x/y/z), unités/sec_enum

        Move    x, 100
        ID_objet -> Move Z, 30
```

Bouge l'objet le long de l'axe sélectionné (x,y ou z) à la vitesse choisie. Cette vitesse est exprimée en unités par seconde (21.3 unités pour 1 pied = environ 30,5 cm). Ainsi, la distance parcourue par frame dépendra du nombre de frames par seconde, tandis que la distance parcourue en une unité de temps n'en dépendra pas. Le mouvement est basé sur le système de coordonnées local de l'objet. Ainsi, un mouvement positif sur l'axe y déplacera toujours l'objet vers l'avant le long de son vecteur local :



Note: *Move* ne marchera pas sur les Acteurs, ce qui inclue le joueur. Cependant, cela marchera sur les acteurs morts (Forum info / Argent). Comme pour toutes les fonctions utilisant la "flèche", *Move* nécessite que l'objet soit placé dans le monde du jeu via l'éditeur, avant de pouvoir l'utiliser dans un script:

```
PlaceAtPC "Mon_Objet", 1,1,1
Mon_Objet-> Move x, 10
```

ne marchera pas si "Mon_Objet" n'a pas déjà été placé, mais un script local peut tourner sur "Mon_Objet" qui utilise

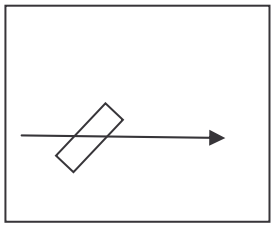
```
Move x, 10
```

Effectuer un déplacement le long de l'axe du monde

MoveWorld axe(x/y/z), unités/sec_enum

```
MoveWorld z, 100
ID_objet -> MoveWorld Z, 30
```

Déplace l'objet le long de l'axe du monde sélectionné (x, y ou z) à la vitesse choisie. Cette vitesse est exprimée en unités par seconde (21.3 unités par pied = environ 30,5 cm). Ce mouvement est basé sur l'axe du monde : un mouvement positif sur l'axe z déplacera toujours l'objet vers le haut, sans tenir compte de local rotation : dans le monde, Z correspond à haut/bas (en augmentant, on va vers le haut), X est l'est/l'ouest (en augmentant, on va vers l'est) and Y est le nord/le sud (en augmentant, on va vers le nord).



Note: MoveWorld ne fonctionnera pas sur les acteurs, ce qui inclut le joueur.

Voici un **exemple** d'un script tiré des forums : une plateforme se déplace lorsque le joueur se trouve dessus :

```
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
```

CellUpdate

CellUpdate

Buggée!

D'après Bethesda : met à jour la position des objets dans la cellule courante. Cette fonction devrait être appelée lorsqu'on déplace des objets sur de grandes distances. Le jeu garde la trace des objets en se basant sur leur cellule d'origine et si un objet change de cellule, son script attaché peut ne pas s'exécuter correctement.
La partie concernant la non-exécution est tout à fait vraie. Des objets peuvent disparaître s'ils sont déplacés trop loin de leur lieu d'origine. Malheureusement à chaque fois que je veux me servir de cette fonction, j'obtiens une erreur d'exécution : "*need to add function code for function CellUpdate*", ie besoin d'ajouter le code de la fonction CellUpdate.

Note: une solution à ce problème (requiert Tribunal) est de désactiver et d'effacer (`SetDelete`) l'objet à la 'rencontre' d'un évènement régulier (pour les '*ridable objects*', il s'agit de l'entrée dans une nouvelle cellule) et d'immédiatement créer une nouvelle version (`PlaceItem`) à la même position grâce à un script global (voir le script de MadMax pour le bateau de l'Académie de Pêche). Voir la section Trucs et Astuces pour une explication en profondeur de MadMax lui-même. Ça marche comme un charme car l'objet ne quitte jamais réellement sa cellule d'origine.

Régler la position (l'autre façon de créer des déplacements)

SetPos, axe, float_enum_pos (float_var pour Tribunal/Bloodmoon)

```
SetPos, z, 477
ID_objet -> SetPos X, 466
```

Cette fonction (à l'inverse des fonctions *Move* et *MoveWorld*) fonctionne aussi avec les Acteurs, et donc le joueur. L'axe est x, y ou z. La valeur float place la position ou l'angle de l'objet à cette valeur. On fait référence au système de coordonnées local de l'objet.
Note: avec Tribunal, cette fonction accepte les variables locales de type float, mais seulement à l'intérieur des cellules actives courantes. C'est approprié pour les extérieurs, on ne peut pas déplacer les objets à une distance arbitraire, l'endroit cible doit se situer dans l'une des cellules actives (la cellule courante plus ses voisines). (Forum info / reposté par Srikandi). Notez aussi que, tandis que les objets peuvent être placés par *SetPos* à n'importe quelle position (sans qu'une collision soit détectée), la fonction vérifiera toujours pour les acteurs s'il y a collision ; dans ce cas-là, ils ne se déplaceront pas toujours comme prévu.

Script de démonstration: ce script est fait pour les caisses flottantes dans les égouts de Longsanglot (Tribunal). Il montre comment *SetPos* et *SetAngle* peuvent être employées à la place de *MoveWorld* et *Rotate* pour produire des mouvements fluides :

```
begin floatAboveStartHeight

float   timer
float   swingTime
float   startAngle
float   startHeight
float   currangle
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
```

Positionner un objet dans le monde ou dans une cellule intérieure

Position, float_enum_x, float_enum_y, float_enum_z, float_enum_zRot
(pour les extérieurs)(floats acceptés dans les extensions)

PositionCell, float_enum_x, float_enum_y, float_enum_z, float_enum_zRot, "ID_cellule"
(pour les cellules intérieures/extérieures) (floats acceptés dans les extensions)

```
position -23515, -15355, 3355, 90
Player -> position -23515, -15355, 3355, 90
"ID_Acteur" -> PositionCell, -254, 475, -376, 360, "Balmora, Club du Conseil"
```

L'application classique de cette fonction est l'anneau de téléportation, transportant le joueur vers certaines destinations. Cependant, elles peuvent aussi être utilisées pour bouger (*warp*) les PNJs ou des objets vers un nouveau lieu. Notez que dans Morrowind seul, ces fonctions n'acceptent que des valeurs littérales en arguments. (cela a probablement changé avec Tribunal : pas sûr que ce soit le cas de toutes les versions ou seulement dans les extensions mais : *Position/PositionCell* peuvent prendre des variables de type float, mais elles doivent être LOCALES ! (info par Indigo Rage).
Z_Rot ne s'exprime pas en degrés (0-360°) mais en minutes (1° = 60 min): donc, si vous voulez qu'une personne soit face à l'est, utilisez 5400. Sud, 10800. Ouest, 16200.

Note: il faut être conscient qu'utiliser *PositionCell* dans le champ résultat d'un dialogue n'est pas fiable et peut causer des crashs. La façon pour Bethesda de faire cela correctement est d'utiliser *StartScript* pour démarrer un script qui exécute la téléportation. (Forum Info/Emma). Utiliser cette fonction sur des objets de l'inventaire du joueur cause des crashs (Forum info/Nigedo).

Script de démonstration : un anneau de téléportation simple pourrait ressembler à ça :

```
Begin TeleportScript
; à attacher à un anneau

short statut
short bouton
short OnPCEquip

if ( MenuMode == 1 )
        return
endif

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

If ( statut == 10 ); affichage du menu
        MessageBox " Téléporte-moi à ", "Balmora", "Vivec", "Annuler"
        Set statut to 20
Elseif ( statut == 20 ); attente de la réponse
        Set bouton to GetButtonPressed
        If ( bouton == -1 ) ; toujours pas de réponse
                Return
        Elseif ( bouton == 0 ); Balmora sélectionnée
                Player -> PositionCell -21278, -17613, 534, 0, "Balmora (-3, -3)"
        Elseif ( bouton == 1 ); Vivec sélectionnée
                Player -> Position 29872, -82108, 578, 180
        Elseif ( bouton >= 2 ); Annuler sélectionné
                Set statut to 0
        Endif
Endif

End
```

Notez que les 2 cibles sont des cellules en plein air et qu'on utilise différents formats. Si vous essayez une téléportation vers un lieu hasardeux (collision avec un objet ou sortie dans le

vide), vous serez placé à l'endroit sûr suivant.
Vous pouvez aussi replacer les PNJs ou des objets avec cette fonction.

Replacer un objet dans sa position initiale

SetAtStart

```
SetAtStart
ID_objet -> SetAtStart
```

Ceci replace l'objet dans sa position initiale, telle qu'elle a été définie dans l'éditeur, avant qu'un mouvement ou une rotation ait eu lieu. Par exemple, regardez le script de la plateforme mouvante sous le sujet "Bouger le long de l'axe du monde ". Voyez la fonction *Move* pour un script de démonstration.

Placer un objet près du PJ

[no fix] PlaceAtPC, "ID_objet", nombre_enum, distance_enum, direction_enum

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

Cette fonction place une nouvelle instance de "ID_objet" près du joueur. La fonction vous laisse choisir la direction -relative au joueur- où l'objet apparaît et la distance (en unités). Si cet endroit n'est pas sûr (en l'air, dans un mur, etc) l'objet sera placé sur l'un des autres axes ou à l'endroit même où se tient le joueur (ses pieds). (Erratum: merci à Isildur et Esteban pour m'avoir signalé que nombre_enum et distance_enum étaient intervertis dans les versions précédentes).

direction correspond à :

0 = devant
1 = derrière
2 = à gauche
3 = à droite

Placer des objets près d'autres objets

PlaceAtMe "ID_objet" nombre_enum, distance_enum, direction_enum

```
Objet->PlaceAtMe 'ID_objet' nombre_enum distance_enum direction_enum
```

La fonction *PlaceAtMe* est semblable à *PlaceAtPC*, si ce n'est que la fonction n'est pas centré sur le PJ. Bloodmoon se sert de cette fonction pour placer des attaquants à différents endroits selon la distance du PJ à un moment donné. Cela permet au script de donner l'impression qu'un grand nombre d'ennemis continue d'arriver, entre autres choses.

```
; FAIT APPARAITRE UN CHASSEUR A L'ENDROIT APPROPRIE, INCREMENTE UN COMPTEUR ET REINITIALISE LE
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
```

Créer une nouvelle instance d'un objet avec PlaceItem

```
[no fix] PlaceItem "ID_objet", float_var_X, float_Y, float_Z, float_Zrot

[no fix] PlaceItemCell "ID_objet", "ID_cellule", X, Y, Z, Zrot
```

Ces fonctions sont utilisées pour créer de nouvelles instances d'objets. *PlaceItem* créera une référence à l'objet "ID_objet" aux coordonnées (X, Y, Z) avec une rotation Z Zrot dans la cellule courante. **Z_Rot ne s'exprime pas en degrés (0-360°) mais en minutes (1° = 60 min) : donc, si vous voulez qu'une personne soit face à l'est, utilisez 5400. Sud, 10800. Ouest, 16200.**
La fonction accepte les variables (locales) de type float. *PlaceItemCell* fait la même chose que *PlaceItem* excepté qu'il vous permet de spécifier une cellule plutôt que de créer l'objet dans la cellule courante. Pour chaque fonction, si la cellule cible pour la référence est une cellule extérieure et si les coordonnées fournies se situent en dehors de cette cellule, alors l'instance sera placé dans la cellule contenant les coordonnées.
C'est un ajout sympathique qui vous permet d'ajouter des choses dans le monde sans le savoir préalablement placés dans l'éditeur.
JOG a posté l'information intéressante suivante à propos de *PlaceItemCell*:
Lorsque j'ai essayé PlaceitemCell la première fois, je pensais que c'était une bonne fonction pour placer les objets sans avoir recours à une "cellule de stockage" qui contiendrait les objets jusqu'à ce que le jeu en ait besoin. J'ai vite réalisé qu'on ne peut pas faire référence à ces objets dans les scripts. (Pour les "disable" par exemple.) Le script ne compilera pas tant qu'un objet au moins ne sera pas utilisé [par le jeu] afin que la commande disable fasse référence à cet objet ; donc on a toujours besoin de PositionCell.

(Note de GBG : vous pouvez contourner ceci dans la plupart des cas en ayant un script attaché à l'objet afin de vous passer de la notation pointée - la flèche – et d'avoir des fonctions telles que disable qui font lui font référence par défaut.)

DinkumThinkum a ajouté :
PlaceItemCell aurait été parfaite pour ce que je voulais faire jusqu'à ce que je découvre que le PNJ placé disparaissait si je sauvais puis rechargeais avant d'aller dans la cellule où il se trouvait.

(Note de GBG : cela peut être évité dans la plupart des cas en testant l'entrée du joueur dans la cellule et en plaçant le PNJ en fonction de ce test. Je pense toujours que PlaceItem est une fonction très utile.)

Script de démonstration :

Attacher ce script à un objet demande au joueur, s'il l'active, le type d'un objet puis crée une référence de celui-ci à 100 unités au-dessus de l'objet activé avec une rotation Z de 45°. Si le type clef est sélectionné, l'objet est créé aux mêmes coordonnées et orientation dans la cellule "salle des clefs" au lieu de la cellule courante.

```
Begin makethingsimple

short questionAsked   ;question posée, oui ou non
short button          ;bouton correspondant à la réponse
float myX             ;coordonnées de l'objet à placer
float myY
```

```
float myZ

if ( MenuMode )
	return
endif

if ( OnActivate == 1 )
		if ( questionAsked == 0 )
			MessageBox, "Créer un nouveau..."  "...Pot"  "...Clef"
			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

end
```

Rotation et angles

Bethesda utilise à la fois les degrés et les minutes (1° degré = 60 minutes) comme unités dans les fonctions :
GetAngle [°] (de -180 à 180 °)
Setangle [°]
Position [min]
PositionCell [min]
PlaceItem [min]
PlaceItemCell [min]
Rotate [° / seconde]
RotateWorld [° / seconde]

Faire tourner un objet

Tout comme les déplacements décrits ci-dessus, les objets peuvent aussi effectuer une rotation, selon un axe local ou l'axe du monde, et déterminer l'angle courant :

```
Rotate , axe, angle/sec_enum

RotateWorld, axe, angle/sec_enum
```

```
	Rotate, z, -30; tourne dans le sens contraire des aiguilles d'une montre, 30° par seconde, autour de l'axe z de l'objet
	ID_objet -> Rotate, Y, 100
```

L'axe peut être x, y, ou z. Remarquez que, comme les fonctions de déplacements, les valeurs données à *Rotate* ou *RotateWorld* est un paramètre exprimant la vitesse (et pas un angle) ; si vous voulez faire tourner un objet à 90°, utilisez *SetAngle* (pour une rotation instantanée) ou utilisez *Rotate* en le combinant avec *GetAngle* pour vérifier jusqu'à quel point l'objet a été tourné. Ces fonctions ne peuvent être utilisées sur les Acteurs.

Régler l'orientation (*Setting Angles*)

SetAngle, axe, float_enum_angle

```
SetAngle, z, 30
ID_objet -> Setangle, z, 25
```

Cette fonction place un objet suivant un angle spécifique du monde. L'axe est x, y ou z. La valeur float place l'angle de l'objet appelant à cette valeur. On fait référence au système de coordonnées local.
Note: d'après des tests effectués à la console, cette fonction n'affecte pas les Acteurs. Avec Tribunal, elle accepte les variables locales de type float, mais seulement à l'intérieur des cellules actives. C'est approprié pour les extérieurs, on ne peut pas déplacer les objets à une distance arbitraire, l'endroit cible doit se situer dans l'une des cellules actives (la cellule courante plus ses voisines). (Forum info / reposté par Srikandi). Pour les acteurs, voir la fonction "Face, x, y"
Script d'exemple : Voir la fonction *SetPos*

Fonctions d'échelle (Scale Functions)

GetScale (float)
SetScale newScale_float
ModScale scaleChange_float

```
If (doonce == 0 )
        ID_objet -> SetScale 0.1
        Set doonce to 1
endif
```

Ces fonctions servent à déterminer ou modifier l'échelle d'une référence. Elle doit être comprise entre 0 et 10 (exclus) ; les valeurs supérieures à la plage 0,5-2 sont autorisées contrairement à la description originale de Bethesda (info de Mode Locrian). Le petit script ci-dessus permet donc de dépasser les limites imposées dans le TESCS.
Notes: vous ne devriez pas appeler "*SetScale*" à chaque frame, ou du moins pas dans les extérieurs ou dans les situations où le FPS (**NdT**:Frame Per Second) est critique.
L'échelle est réinitialisée à une valeur comprise entre 0.5 et 2.0 lorsque vous rechargez. N'utilisez donc pas de flag *done/fait* pour ne l'appeler qu'une seule fois. Soit vous l'appelez régulièrement (toutes les 10 frames), soit vous testez avec "*GetScale*" et réinitialisez l'échelle lorsque ça ne va pas :

```
if ( GetScale != 5 )
SetScale, 5
endif
```

Un autre moyen est d'utiliser les scripts de démarrage de Tribunal et Bloodmoon. Cela ne sera fait qu'une fois et vous serez sûr que l'échelle sera mise à jour à chaque chargement. (Forum info / JOG).
Script de démonstration:

Un script plus complexe de Bethesda vous montre comment faire grandir et rétrécir un objet. Lorsque ce script est placé sur une référence, une activation permet au joueur de changer l'échelle de cette référence.

```
Begin scalescript

short questionAsked
```

```
short button

float direction
float currscale
float tempscale

if ( MenuMode )
	return
endif

if ( OnActivate == 1 )
		if ( questionAsked == 0 )
			MessageBox, "Que cet objet..."  "...Grandisse"  "...Rétrécisse"
			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

end scalescript
```

Déterminer le lieu, la position relative et le déplacement

Détecter si le joueur est à l'intérieur ou à l'extérieur

[no fix] GetInterior (renvoie Booléen/short)

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

Pas de documentation sur cette fonction ! (Merci à XP-Cagey et Killgore)
Cette fonction renverra 1 si la cellule courante est une cellule intérieure et 0 si c'est une cellule extérieure. Voici un script global réalisé par Killgore. Si vous voulez l'essayer, démarrez-le en tapant "StartScript Outside_Check" dans la console.

```
Begin Outside_Check
short doonce

if (MenuMode == 1)
Return
EndIf

if (doOnce == 0) ;si on se trouve dans une nouvelle cellule
                 ;ou que le script vient juste de démarrer

if ( GetInterior == 1 )
```

```
        MessageBox "1: intérieur"
elseif ( GetInterior == 0 )
        MessageBox "0: extérieur"
else
        MessageBox "le sinon mystérieux "
endif

set doOnce to 1
Return
endif

if (doOnce == 1)
if (CellChanged == 0)
        Return
else ;si le joueur change de cellule dans cette frame..
        set doOnce to 2 ;on attend une frame supplémentaire
endif

if (doOnce == 2) ;alors recommence
        set doOnce to 0
        Return
endif

End Outside_Check
```

Déterminer la cellule où se trouve le joueur

[no fix] GetPCCell, "ID_cellule" (renvoie Booléen/short)

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

La fonction *GetPCCell* teste la présence du joueur dans la cellule spécifiée. Elle renvoie 1 si le joueur est dans la cellule spécifiée, 0 sinon. Les '*matchings*' partiels sont supportés : GetPCCell, "Vivec" renverra vrai pour les cellules "Vivec", "Vivec, canaux du quartier étranger " et "Vivec, temple", etc.

Script de démonstration :

Ce script de Bethesda vérifie que le PC quitte une certaine zone pour enlever un certain objet d'un PNJ :

```
Begin DrothPost

if ( GetJournalIndex "MS_EstateSale" >= 70 )
        if ( GetPCCell "Longsanglot, maison de Géon Auline" ==  0 )
                "Geon Auline"->RemoveItem "silver dagger_droth_unique" 1
                Journal MS_EstateSale 80
                StopScript DrothPost
        endif
endif

End DrothPost
```

Distance d'un objet à un autre

GetDistance, "ID_objet" (renvoie float)

```
        "ID_objet1" -> GetDistance, "ID_objet2"
```

Cette fonction renvoie la distance (en unités) séparant deux objets. Dans la première syntaxe, il s'agit de la distance entre l'objet appelant (auquel le script est attaché) et l'objet nommé. Elle peut être utilisée pour déclencher une attaque ou d'autres évènements, ou pour déterminer à peu près l'endroit où se trouve le joueur afin de démarrer un script.
Voici un morceau de script issu de Morrowind seul :

```
; issu d'un script attaché au PNJ Ashamanu:
; Ashamanu donnera l'entrée de journal 60 lorsque le joueur est assez proche
```

```
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
```

Limitations :

- *GetDistance* requiert que l'objet passé en paramètre soit placé dans le monde du jeu (dans l'éditeur) et a la case *references persist* cochée (ou qu'il soit naturellement persistant comme un PNJ)
- Notez que cette fonction doit être utilisée seulement avec des identifiants uniques ou dans des environnements où vous savez qu'il n'existe qu'une seule instance de cet identifiant – sans cela, le moteur du jeu se contentera de la première instance de l'ID qu'il trouvera et renverra cette distance – probablement pas la distance que vous vouliez. Ainsi un script qui prévient le joueur de la présence d'un poisson carnassier situé à moins de 800 unités devra être attaché à l'ID du poisson et vérifier la distance au joueur qui est unique, et non pas l'inverse.
- Si vous déterminez la distance d'un objet que vous êtes en train de bouger avec *Move* ou *MoveWorld*, *GetDistance* donnera toujours la **distance de sa position initiale** (celle définie dans l'éditeur). Utilisez *GetPos* et le bon vieux Pythagore ($c^2 = a^2 + b^2$) pour déterminer les distances dans ces conditions.

Déterminer la position et l'orientation d'un objet

GetPos, axe(x/y/z)

```
	ID_objet -> GetPos, z
```

Quand vous bougez des objets avec les fonctions *Move/MoveWorld* décrites ci-dessus, vous voudrez sûrement obtenir des informations sur l'endroit où ils se trouvent. Cette fonction marche sur les Acteurs et les objets. Dans le script suivant, j'utilise cette fonction pour contrôler les mouvements d'une source lumineuse (un feu) pour faire un feu de camp dont les flammes démarrent doucement et meurent le soir en fonction de l'horaire de la journée – la position Z initiale de l'objet feu est 511:

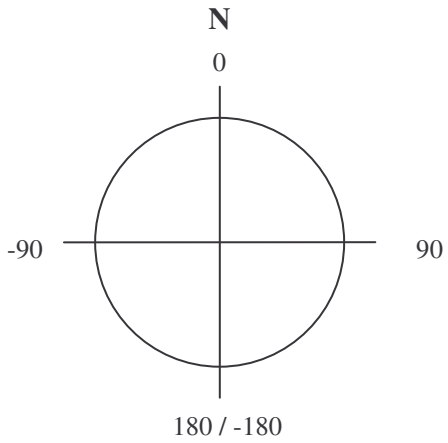
```
Begin _HB_Scheduled fire

short control_fire
; le script est attaché à un PNJ qui garde le feu.
; ******* ceci contrôle l'horaire du feu:
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 ; diminue le feu
		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 ; augmente le feu
		else
			HB_Light_Fire_camp -> enable
		endif
	endif
endif
end
```

GetAngle , axe(x/y/z) **(renvoie float)**

```
If ( ID_objet -> GetAngle, z == 180 )
```

La fonction *GetAngle* renvoie l'angle du monde, et pas l'angle local. Les angles du monde vont de 0 à 180 et de 0 à -180 (voir figure pour l'axe z).
Note: cela marche sur les acteurs et les objets, cependant pour le joueur (et je suppose pour les autres acteurs), seul l'axe z est pertinent - *GetAngle*, x ou y renvoie toujours 0.



Ligne de vue (*Line of Sight*) :

GetLOS, ID_objet **(renvoie Booléen/short)**

```
ID_Acteur -> GetLOS, Player
```

Pas de documentation:

GetLineOfSight (renvoie Booléen/short)
(peut-être celle-ci marche-t-elle mieux ? je n'ai pas testé)

Cette fonction détermine si l'objet appelant a l'objet référencé dans sa ligne de vue. Cela ne semble pas fonctionner pour les objets n'étant pas des acteurs, pour ce que j'ai pu en voir. Cela ne prend pas l'orientation en compte, donc le mot "vue" n'est pas à prendre dans son sens littéral. (Voir "Serait-elle en train de me regarder?" dans la section Trucs et Astuces.)

Note: *GetLOS* est une fonction qui ralentit l'exécution, ne laissez pas le script l'appeler à chaque frame.

Script de démonstration:

```
Begin balynScript
float timer
short doOnce
[…];références au journal
Set timer to ( timer + GetSecondsPassed )
if ( timer < 5 ); un timer pour éviter de tester trop souvent (pour éviter les problèmes de performances)
        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
```

Déterminer si un acteur a été détecté par un autre acteur

[no fix] GetDetected, "ID_Acteur" (renvoie Booléen/short)

```
        If ( GetDetected, Player == 1 )
```

Renvoie vrai si **n'importe quel** Acteur appelant peut détecter "ID_Acteur" (merci pour la correction, ThePal!). Cette fonction renverra 0 si l'acteur s'est caché, c'est-à-dire en cas de succès de sa compétence de discrétion, ou en utilisant des sorts d'invisibilité ou de caméléon. D'après le fichier d'aide, c'est une fonction qui ralentit l'exécution, ne l'appelez pas trop souvent (faites un compteur pour ne l'appeler que toutes les 3 secondes).

Script de démonstration : le joueur doit s'approcher d'un objet sans être détecté – sinon il est pris sur le fait

```
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 ) ; pour être sûr que GetDetected n'est appelée qu'une fois toutes les 5 sec.
        Return
endif

Set timer to 0

if ( GetDistance Player <= 1024 )
        if ( player->GetDistance "com_chest_02 " <=128 )
                if ( GetDetected Player == 1 )
                        ForceGreeting ; Le joueur est pris sur le fait et sera puni
                        Journal EB_Bone 50
                endif
        endif
endif

End jeanneScript
```

Déterminer lorsque le PJ quitte une cellule

[no fix] CellChanged

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

CellChanged renvoie 1 pour une frame lorsque le joueur change de cellule. Si le script appelant cette fonction est local, elle se déclenchera à l'entrée du joueur dans la cellule où il est actif – quitter la cellule ne déclenche pas la fonction car le script se termine avant que le changement de cellule ne soit effectif (Merci à Klinn pour cette correction). Il existerait un

léger bug : se téléporter en dehors de la cellule ne déclencherait pas la fonction (pas de confirmation).

Script de démonstration: dans le *SlaveScript*, qui gère la libération des esclaves dans le jeu, *CellChanged* provoque le départ des esclaves – ils sont partis vers un monde meilleur :

```
Begin SlaveScript
[…]
if ( slaveStatus == 3 )
        if ( GetCurrentAIPackage == 3 )
                AIWander 512 0 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
```

un autre bon exemple est le Gateway Haunt's script. Le spectre revient toujours lorsque vous ne regardez pas :

```
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
```

Détecter si le joueur voyage ou est en prison

[no fix] GetPCInJail (renvoie booléen)
[no fix] GetPCTraveling (renvoie booléen)

```
        if ( GetPCTraveling == 1 )
```

Bloodmoon ajoute cette fonction pour vérifier si le PJ est en plein voyage (sur un échassier des marais). Renverra 1 si on voyage/est en prison, zéro sinon. Elle est utilisée dans le script de transformation en loup-garou pour empêcher le joueur de se transformer dans ces cas-là :

```
if ( PCWerewolf != 1 ) ; NE PAS EXECUTER SI LE PJ EST UN LOUP-GAROU
        return
endif

if ( GetPCinJail == 1 )
        return
endif

if ( GetPCTraveling == 1 )
```

```
        return
endif
```

Déclencher une action lorsqu'un acteur se tient sur un objet

GetStandingPC (renvoie Booléen/short)

Renvoie 1 si le PJ se tient dessus.

GetStandingActor (renvoie Booléen/short)

Renvoie 1 si N'IMPORTE QUEL Acteur (dont le PJ) se tient dessus.

```
        If ( Object_Id -> GetStandingPC == 1)
                […déclenche un piège horrible]
        endif
```

C'est une fonction très utile pour déclencher des évènements, surtout dans les cellules intérieures. C'est aussi une excellente fonction pour les pièges. Vous pouvez faire un objet "activateur" à partir de n'importe quel fichier nif d'un objet statique (dont les couloirs, tapis etc.) et déclencher certains évènements lorsque le joueur (ou un autre acteur) marche dessus.
Mon script sert à allumer un feu dans un hall dès que le joueur fait un pas sur une partie spécifique du sol :

```
Begin HBHallLighting

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

end
```

HB_hallfire est une variable globale, utilisée pour allumer le feu. Voici le script pour le feu :

```
Begin HBHallfireon

if ( HB_hallfire == 1)

        if ( GetPos, z, < -736 )
                MoveWorld, z, 3 ; le feu grandit jusqu'à une hauteur max.
                if ( GetPos, z, > -780)
                        enable
                endif
        endif
else
        disable
endif

end
```

Blesser un acteur se tenant sur un objet

HurtStandingActor, float_Points_de_Vie/s

```
        HurtStandingActor -3
        ID_objet -> HurtStandingActor 1

        Float hurt_variable
        HurtStandingActor, hurt_variable
```

Cette fonction affecte la santé d'un Acteur (dont le PC) qui se tient sur un objet. Les valeurs positives réduiront la vie, les valeurs négatives soigneront (points de vie par seconde). Utiliser des valeurs négatives modifiera votre santé même au-delà du maximum, soyez donc prudent si vous voulez vous en servir de cette manière. La fonction accepte les variables de type float.

Script de démonstration : l'effet le plus connu est sûrement les fleuves de laves :

```
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 de dommage par seconde

End lava
```

Fonctions de collision avec les objets

GetCollidingPC (renvoie Booléen/short)
GetCollidingActor (renvoie Booléen/short)

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

HurtCollidingActor, dommages_enum

```
        HurtCollidingActor, 100
        ID_objet -> HurtCollidingActor, 100
```

Ces fonctions permettent à un objet d'interagir avec les Acteurs qui entrent en collision avec lui. GetCollidingPC renvoie 1 si le PJ est actuellement en contact/collision et 0 sinon. GetCollidingActor fonctionne de la même manière mais renverra 1 si n'importe quel Acteur (autre que le PJ) entre en contact avec l'objet. HurtCollidingActor diminue la santé de l'Acteur en collision de dommages_enum points par seconde.

Script de démonstration :

Lorsque ce script est placé sur un objet, tout acteur touchant cet objet se blessera. Un message différent s'affiche selon qu'il s'agisse du PJ ou de quelqu'un d'autre.

```
Begin hurtActor

if ( GetCollidingPC == 1 )
        MessageBox "Vous criez de douleur lorsque vous touchez le rocher."
Elseif ( GetCollidingActor == 1 )
        MessageBox "Vous entendez un cri de douleur proche."
Endif

HurtCollidingActor 100

End
```

Vérifier l'activation d'un objet et l'activer

Un objet est habituellement activé lorsque vous pressez la barre d'espace pour l'utiliser. La plupart des objets exécute une action standard lorsqu'ils sont activés : les portes s'ouvrent les coffres affichent leur contenu, les Acteurs entament le dialogue etc. La fonction *OnActivate* vous permet d'intercepter cette action standard pour faire quelque chose d'autre ou vérifier une condition d'abord :

OnActivate

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

Cela passe à 1 pour une frame lorsque l'objet est activé. *OnActivate* se réinitialise par lui-même dès que la fonction est appelée ; un seul script peut récupérer avec succès le retour de *OnActivate* et vous ne devriez avoir qu'un seul appel à *OnActivate* dans votre script à un moment donné. *OnActivate* court-circuite le comportement normal de l'objet - Pour lancer l'action standard de l'objet une fois que *OnActivate* a été appelé, vous devez utiliser la commande *Activate* :

```
Activate

        activate (si le script est placé sur l'objet à activer)
```

Activate peut être utilisé pour déclencher l'action standard d'un objet après qu'*OnActivate* ait été appelé.

Les actions standard exécutées par *Activate* sont :

Portes -> Ouverture (peut-être buggé dans le MW de base, mais pour la v1.6.1820 ça marche parfaitement).
LoadDoors -> Rien (buggé ?)
Containers -> Ouverture (montre le contenu)
Livres/Parchemins -> Affiche le texte à lire *.
Activators -> Rien
Acteurs -> Dialogue
Arme -> Ramasser
Armure -> Ramasser
Miscellaneous/Divers -> Ramasser

Pas testé, mais je suppose :
Equip. Lights -> Ramasser
Other Lights -> Rien
Probes (Sondes) -> Ramasser
Aparatus (Alchimie) -> Ramasser

*(mais semble désactiver l'option "prendre", peut aussi être responsable d'une boucle sans condition do-once. Voir la section Trucs et Astuces pour des solutions)

Exemple de script :

Le script suivant montre bien l'usage de *OnActivate*. Il est attaché à un objet de type container, un coffre dont le piège ne se déclenche PAS comme le ferait ceux du jeu d'origine :

```
Begin Trap_script

short done

if ( OnActivate == 1 )
        if ( done == 1 ) ;condition do-once
                Activate
                return
        else
                Cast, "flame", Player ;blesse le joueur
                set done to 1
                Activate ; Appelle l'action standard : le coffre s'ouvre
        endif
```

```
endif

End trap_script
```

Notes: d'après mes tests, la fonction *Activate* ne peut pas être utilisée sans qu'il y ait *OnActivate* dans le même script. Les fonctions *Activate* et *OnActivate* peuvent cependant être dans des parties distantes du script. Mais *Activate* ne marchera que si *OnActivate* a été appelée d'abord. On a aussi rapporté que l'objet doit avoir été activé au moins une fois dans les 72 heures précédentes ; sinon le jeu oublie apparemment que la fonction *OnActivate* a été appelée auparavant. Testez donc ce script (je l'ai attaché à une porte) pour voir par vous-même :

```
Short done
if ( OnActivate == 1 )
	MessageBox "Merci de m'avoir activé"
endif

if ( done == 0 )
	if ( GetDistance, Player < 100 )
		set done to 1
		MessageBox "Le joueur est proche"
	endif
endif

if ( done == 1 )
	if ( GetDistance, Player > 200 )
		MessageBox "Sésame"
		Activate
		set done to 0
	endif
endif

End
```

Il y a eu pas mal de retours sur les difficultés à utiliser *OnActivate* en général ; selon moi la meilleure façon de procéder est d'utiliser plus d'un *OnActivate* dans un script. Soyez aussi conscient que pour les objets de l'inventaire, *OnPCEquip* est la fonction à utiliser à la place de *OnActivate* ; celle-ci n'est pas appelée lorsque qu'on bouge un objet vers l'icône du joueur.

Info issue de UESP: il y a une caractéristique de la fonction *Activate* restée sans documentation : spécifier le joueur après la fonction par exemple :

```
begin RemoteContainer
   short OnPCEquip

   if ( OnPCEquip == 1 )
      set OnPCEquip to 0
      "dh_remote_chest_01"->Activate, player
   endif

end
```

Si le container est persistant (les références persistent), ce script devrait ouvrir le container quelque soit l'endroit où se trouve le joueur. C'est un bon moyen de créer des containers 'portables' en attachant un script similaire à un anneau ou d'autres objets de ce type.

Verrouiller/Déverrouiller des portes ou des coffres

```
Lock, short_enum_locklevel
Unlock

	Ma_Porte -> Lock, 50

GetLocked	(renvoie Booléen/short)
```

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

(ne concernent que les objets de type portes ou containers)

Ces fonctions sont utilisées pour verrouiller et déverrouiller les portes et les containers. La fonction `GetLocked` renvoie 1 lorsque l'objet appelant est verrouillé. `Lock` verrouille l'objet à la difficulté spécifiée (0-100). *Lock* 0 produit un effet bizarre : la porte/le container ne pourra plus être choisie ou ouverte. `Unlock` déverrouille n'importe quelle serrure, quel que soit le niveau de difficulté de celle-ci.

Exemple de script :

Voici un script écrit par qwert : le coffre fonctionne comme un dispositif d'entraînement pour la compétence Sécurité en le reverrouillant constamment :

```
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 ) ; se sert du timer pour reverrouiller après 10 secondes
        "Storm_Chest_Trainer"->Lock 50
endif

End
```

Animer les objets

Il y a un groupe de fonctions qui vous permet d'utiliser des animations spécifiques, celles-ci étant définies dans un modèle (fichier .nif). Vous pouvez vous renseigner sur le nom des groupes d'animations dans la fenêtre de prévisualisation (***preview window***) puis parcourir les différents groupes d'animation ou en regardant dans la fenêtre "***base animation***" du menu "***Character***". Un excellent résumé des groupes d'animations pour Acteurs se trouve ici:
http://morrowind.preik.net/animationgroups.html
mais seuls ceux listés dans la fenêtre "***base animation***" peuvent être appelés par cette fonction. Des animations supplémentaires peuvent être chargées à un modèle via le bouton animation dans les propriétés de l'objet. Voir les danseuses de la Maison des Plaisirs Terrestres de Désèle à Suran pour un exemple.
Certains modèles n'ont pas de groupes d'animation mais les différentes bannières (sous les activateurs) sont de bons exemples pour voir ce que cela veut dire. Des exemples de *GroupName* sont : idle, idle2, idle3, walk, etc.
Ces fonctions ne marchent pas sur le PJ.

`PlayGroup, GroupName, [Flags]`

```
PlayGroup, walk, 1
```

Joue l'animation définie par *GroupName*. Les flags optionnels peuvent être utilisés pour démarrer le groupe de différentes manières (voir plus bas).

`LoopGroup, GroupName, Nombre_enum, [Flags]`

Joue l'animation définie par *GroupName*. L'animation sera répétée autant de fois que le nombre spécifié ; on revient ensuite à l'animation Idle. Les flags optionnels peuvent être utilisés pour démarrer le groupe de différentes manières (voir plus bas).

SkipAnim

Empêche l'animation courante d'être jouée pour cette frame.

Flags:

0 = Normal
L'animation courante sera jouée jusqu'au bout, puis la nouvelle animation commencera à partir du début.
1 = Immediate Start
L'animation courante s'arrêtera quelque soit la frame ; la nouvelle animation commence.
2 = Immediate Loop
idem mais démarre une boucle.

Note: *PlayGroup* ne fonctionne pas sur le PJ. Si Bloodmoon est installé (pas spécialement coché dans les Data Files), certaines animations de PNJs sont chamboulées. Appelées à la console, elles peuvent sembler correctes, mais si vous insérez NPC->PlayGroup, group, 1 dans un script, vous serez peut-être surpris de constater que l'animation est différente de celle que vous auriez attendu (Forum info / Kir). Vous devrez faire des tests pour retrouver l'animation souhaitée (voir la section sur *AIWander* pour une liste des mouvements Idle des PNJs ; Kir travaille actuellement sur un outil appelé "NPC Animation Explorer", ou explorateur d'animations des PNJs, gardez un oeil là-dessus!).

Exemple de script :

Script original attaché à chacune des bannières extérieures. Elles bougent différemment en fonction de la météo :

```
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

end OutsideBanner
```

Rendre les objets disponibles ou indisponibles (Enable/disable)

Enable
Disable

```
        "ID_objet" -> enable

GetDisabled    (renvoie Booléen/short)
        If ( GetDisabled == 1 )
                Return
        Endif
```

La fonction *Disable* fait totalement disparaître un objet dans le monde du jeu, ce qui signifie qu'il n'est pas interprété ou traité (les scripts attachés sont cependant toujours actifs). La fonction *Enable* rend un objet visible avec lequel on peut de nouveau interagir. *GetDisabled* (renvoie 1 si l'objet est indisponible) peut être utilisée pour connaître le statut courant d'un Objet. Ces fonctions sont très puissantes et peuvent être utilisées pour échanger différents modèles de *statics* (une maison normale remplacée par des débris, etc). Elles sont utilisées dans le processus de construction des places fortes.
Exemple de script:

Un exemple du jeu est le *SlaveScript* qui fait disparaître les esclaves libérés une fois que le joueur a quitté la cellule :

```
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 ; les esclaves disparaissent une fois libérés et lorsque le joueur
quitte la zone
        endif
endif

end slaveScript
```

Avertissement : désactiver les sources lumineuses
Il semblerait qu'il y ait un problème dû au moteur du jeu lorsque les lumières sont désactivées – les Acteurs et d'autres types d'objets sont toujours éclairés tandis que le monde autour ne l'est plus. Je n'ai pas testé à fond pour voir si cela peut être évité mais un bon moyen de s'y prendre est de supprimer physiquement la lumière à un endroit distant (quelques mètres sous terre) plutôt que de la désactiver. Une autre astuce d'Indigo: si vous placez via la fonction *Enable* une lumière 'négative' (c'est-à-dire qu'elle génère des ténèbres à la place de la lumière) après que vous ayez désactivé la lumière normale, le problème de l'illumination disparaît.

Supprimer complètement une référence

[no fix?] SetDelete flag_enum

```
        SetDelete 1
```

La fonction *SetDelete* peut être utilisée avec *Disable* pour supprimer un objet plus efficacement. *SetDelete 1* marque la référence à effacer et *SetDelete 0* réinitialise le flag. Cela peut être utile pour des optimisations. Certains objets possèdent des scripts qui simule leur ramassage en rendant indisponible la référence activée et qui ajoute ensuite un nouvel objet dans l'inventaire. Il existe toujours une référence, indisponible, de l'objet à cet endroit (ce qui prend de la mémoire et du temps machine car le script attaché à cet objet s'exécute toujours à chaque frame). Si la référence est marquée pour être effacée alors elle n'existera plus.. Si la référence provient du fichier master, elle est toujours là mais elle sait qu'elle ne le devrait pas et n'a donc ni représentation graphique ni scripts actifs. Si elle a été créée dans le jeu, elle sera effacée.

Note : *SetDelete* plantera Morrowind si une autre fonction s'exécute sur l'objet dans la même frame.

Ce script **provoquera un crash** :

```
Begin _spell_effect
float timer
rotate y 120 ; crash causé par ceci
if ( timer < 3 )
        set timer to ( timer + GetSecondsPassed )
else
        disable
        setdelete 1
endIf
```

La solution est de d'abord rendre l'objet indisponible et d'utiliser ensuite *GetDisabled* et Return pour effacer l'objet de manière sûre:

```
Begin _spell_effect

DontSaveObject

float timer

if ( GetDisabled == 1 )
        setdelete 1
        return
endIf

rotate y 120

if ( timer < 3 )
        set timer to ( timer + GetSecondsPassed )
else
        disable
endIf
```

Une autre solution, proposé par Soralis, est d'utiliser une variable locale "deletobj" comme un flag:

```
if ( deleteobj = 1 ) ;variable locale, positionnée lorsque vous voulez effacer
    if ( deletetimer == 0 )
        Disable
    endif
    if ( deletetimer < 10 )
```

```
        set deletetimer to ( deletetimer + 1 )
    endif
    if ( deletetimer == 10 )
        SetDelete, 1
    endif
    Return
endif
```

De plus, vous devriez toujours appeler *SetDelete* à partir d'un script local attaché à l'objet que vous voulez supprimer. N'utilisez jamais Objet->SetDelete 1 car cela provoque des crashs lorsque Morrowind tourne. Vous ne devriez pas non plus supprimer un objet présent dans l'inventaire : l'encombrement en sera faussé. Si vous êtes forcé de supprimer un objet présent dans l'inventaire du PJ, essayez la commande *Drop* avant *Disable* et *Delete*. Il faut aussi savoir que, parfois, les effets magiques peuvent causer des problèmes avec la fonction *SetDelete* (Forum Info/Dan_Wheeler).

Ne pas sauvegarder les changements appliqués à un objet

DontSaveObject

Appelez cette fonction si vous ne voulez pas conserver dans la sauvegarde les changements appliqués à un objet.
Servez-vous de *DontSaveObject* sur des objets :
1. qui sont activés/désactivés (via *Enable*/*Disable*) pendant le jeu, comme pour les différentes phases de construction de votre place forte.
2. qui se déplacent, tels que les '*ridable objects*'.

En utilisant *DontSaveObject,* vous n'aurez pas le message d'erreur "*save game data has changed*" ("les données de la sauvegarde ont changé") qui apparaît au chargement d'une sauvegarde et où l'état d'un objet a changé ; par exemple, il a été déplacé ou activé/désactivé. Cela est dû au fait que l'état (activé/désactivé, déplacé) de l'objet est stocké dans la sauvegarde. (Forum info / IndigoRage).
Dans le jeu original, cette fonction est utilisée dans le script *SignRotate* et dans celui qui suit :
Script de démonstration :

```
Begin diseaseAscended

DontSaveObject

;les dormeurs élevés ont tous des maladies dues au Fléau...
if ( CellChanged == 0 )
       return
endif
AddSpell "ash woe blight"
AddSpell "black-heart blight"
AddSpell "chanthrax blight"
AddSpell "ash-chancre"

End
```

Scripter les PNJs : IA et Déplacement

Faire marcher un PNJ vers un nouveau lieu

AiTravel, float_enum_x, float_enum_y, float_enum_z, [reset]

```
Acteur -> AITravel, 1359, 2700, 1045
```

Pour qu'un PNJ se déplace entre des lieux différents du monde, on utilise la fonction *AITravel*.
Les variables x, y, z sont les coordonnées du monde. Vous pouvez les déterminer en déplaçant votre caméra au dernier point du déplacement ou en sélectionnant un point de la *path grid* ou les coordonnées d'un objet proche, affichées en bas de la fenêtre d'objet. L'utilité du flag optionnel reset est inconnu.

Il est important, lorsque vous utilisez cette fonction dans les scripts, de fournir des conditions pour que le package IA ne soit appelé qu'une fois. Regardez le script limité suivant :

```
Begin Travel

AiTravel, 1359, 2700, 1045

End Travel
```

Ce script ne fonctionnera pas car il est exécuté continuellement : le PNJ se bloque et ne peux jamais atteindre le lieu désiré.

```
Begin Travel
Short do_once

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

End Travel
```

Celui-ci devrait marcher ; le PNJ se dirigera vers les coordonnées désignées dès que le script deviendra actif, c'est-à-dire dès que la cellule sera chargée.

Vérifier que le PNJ a effectué son déplacement

GetAIPackageDone (renvoie Booléen/short)

```
if ( GetAIPackageDone == 1 )
                [faire quelque chose]
endif
```

Pour vérifier qu'un PNJ est arrivé à destination, on utilise la fonction *GetAIPackageDone*. Elle renvoie 1 pour une frame lorsque le package IA courant a fini.

Utilisez-la pour vérifier qu'un déplacement s'est bien fini. Le script suivant montre comment lier plusieurs commandes *AITravel* dans un script, en utilisant une variable d'état et une structure *if-elseif* :

```
Begin TravelLoop

short etat
float timer
```

```
if ( menumode == 1 ) ; si le menu est ouvert, ne rien faire
	return
endif

;début de la marche
if ( etat == 0 )
	if ( player -> GetDistance HB_adros_darani < 5000 )
		set etat to 5
	endif

;****************** il commence son voyage
elseif ( etat == 5 )
	SetHello 0
	AITravel -8144, -19409, 728 ;nouvelles coordonnées -> point 1
	set etat to 10

elseif ( etat == 10 )

	if ( GetAIPackageDone == 1 )		;il a atteint le point 1
		set etat to 40
	endif

elseif ( etat == 40 )

	AITravel -9147, -19459, 720 ; nouvelles coordonnées -> point 2
	set etat to 50

elseif ( etat == 50 )

	if ( GetAIPackageDone == 1 )		;il a atteint le point 2
		set etat to 60
	endif

elseif ( etat == 60 )
	AITravel -8144, -19409, 728 ;nouvelles coordonnées -> point 1
	set etat to 70

elseif ( etat == 70 )

	if ( GetAIPackageDone == 1 )		;il a atteint le point 1
		set etat to 80
	endif

elseif ( etat == 80 )
	AITravel -6640, -18496, 1040 ;nouvelles coordonnées -> point 0
	set etat to 90

elseif ( etat == 90 )
	if ( GetAIPackageDone == 1 )		;il a atteint le point 0
		set etat to 0
	endif

endif

End TravelLoop
```

On peut aussi trouver de bons exemples utilisant *AITravel* dans le "*lookoutscript*" (Fargoth cachant son trésor) et le script *CharGenWalkNPC* (le garde qui marche dans le bateau au début du jeu). Ou regardez mon plugin "*Traveling Merchants*" (Marchands ambulants en VF) pour une vraie folie de *AITravel* ☺ !
Si vous comptez utiliser intensément cette fonction, ayez conscience des problèmes suivants : si vous vous reposez ou si vous quittez une cellule dans laquelle *AITravel* s'exécute, le script ne détectera jamais le signal *GetAIPackageDone*, ce qui signifie que votre PNJ restera collé sur son chemin une fois que vous reviendrez ou que vous vous réveillerez. Les lignes de code suivantes permettent de reprendre l'exécution du script (adapté pour le script précédent)

```
; ******* Sauvetage pour le script qui s'est arrêté
; ******* pour continuer le script après que le joueur a quitté une cellule ou après un repos
If ( Player -> GetDistance, HB_adros_darani < 5000 )
	if (GetCurrentAIPackage == -1) ; vérifie l'inactivité
```

```
                set timeout to ( timeout + GetSecondsPassed )
                if ( timeout >= 3) ; attend quelque temps.
                                        ; de courtes périodes d'inactivité arrivent souvent
                        set etat to (etat - 10) ; l'arrêt apparaît à
         ; AIPAckageDone - saut pour errer ("wander") à nouveau.
                        set timeout to 0
                endif
        else
                set timeout to 0
        endif
endif
```

Forcer un Acteur à se tourner dans une direction

Face x_enum, y_enum

```
        "ID_Acteur" -> Face, -1334, 334
```

A ma connaissance, cette fonction n'accepte pas les variables (au moins en-dessous de la version 1.6.1820), mais on a rapporté que c'était possible – ça dépend sûrement de la version. Avec cette fonction, les PNJs se tourneront vers les coordonnées indiquées. Apparemment elle interrompt l'animation en cours. En l'utilisant sur des PNJs se déplaçant, ceux-ci stoppent, se tournent, puis continuent leur trajet, dès que la rotation est terminée. (Forum info / JOG, Dan_Wheeler).

Définir des déplacements aléatoires pour les Acteurs

AiWander, portée_enum, durée_enum, temps_enum, [idle1], [idle2], [idle3], …[idle9], [reset]

```
        "Actor_ID" -> AIWander, 512, 5, 0, 0,20,0,0,10,30,0,0,0
```

c'est l'algorithme de déplacement aléatoire utilisé par la plupart des PNJs. Le PNJ se déplace sur la *path grid*, change de direction aléatoirement et adopte une attitude oisive (*idle*) entre les deux.

- `Portée`: détermine la distance jusqu'à laquelle l'Acteur ou la créature vagabondera à partir de son origine.
- `Durée`: probablement le temps (en heures) durant lequel le personnage exécutera le package (avant qu'il ne se réinitialise, ce qu'il semble se passer si vous partez ou dormez ; pas sûr)
- `Temps` : on présume qu'il s'agit du temps de démarrage du package s'il y a une durée
- `[idle1], …[idle9]`: chances des mouvements d'oisiveté.

  Les *Idles* sont (testés dans le jeu):

- Humain (homme) :
  Idle1: Reste immobile
  Idle2: Bascule son poids d'une jambe à l'autre
  Idle3: Regarde derrière
  Idle4: Se gratte la tête, secoue la tête
  Idle5: Replace le vêtement ou l'armure au niveau de l'épaule
  Idle6: Baille et s'étire
  Idle7: Regarde ses mains et scrute les alentours furtivement
  Idle8: Place une main sur la poitrine, comme s'il avait un infarctus
  Idle9: Aggripe son arme, se touche la tête
- Humain (femme) – comme ci-dessus mais :
  Idle5: Main sur la hanche

- Khajiit femelle – comme Humain (homme) mais
  Idle9: Se gratte la tête, secoue la tête

Pour qu'un Acteur reste au même endroit, vous pouvez utiliser : `AIWander, 0, 0, 0`
Note: le nombre d'idles et certaines descriptions n'étaient pas listées correctement dans les versions précédentes (corrigée avec la 8eme édition – tout le mérite va à Whoopa).
Voici un **script d'exemple** qui affiche toutes les idles à la suite (cela peut être utile pour choisir quelle animation vous voulez pour votre PNJ):

```
Begin Animtest

float timer
short compteur
set timer to ( timer + GetSecondsPassed )

if ( timer > 10 )
	set timer to 0
	set compteur to ( compteur + 1 )
	if ( compteur >= 18 )
		set compteur to 0
	endif
endif

if ( compteur == 1 )
	AIWander 0, 0, 0, 100, 0, 0, 0, 0, 0, 0, 0, 0
	MessageBox "Idle 1 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 3 )
	AIWander 0, 0, 0, 0, 100, 0, 0, 0, 0, 0, 0, 0
	MessageBox "Idle 2 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 5 )
	AIWander 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0, 0
	MessageBox "Idle 3 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 7 )
	AIWander 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0, 0
	MessageBox "Idle 4 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 9 )
	AIWander 0, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0, 0
	MessageBox "Idle 5 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 11 )
	AIWander 0, 0, 0, 0, 0, 0, 0, 0, 100, 0, 0, 0
	MessageBox "Idle 6 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 13 )
	AIWander 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 0, 0
	MessageBox "Idle 7 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 15 )
	AIWander 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 0
	MessageBox "Idle 8 , 100"
	set compteur to ( compteur + 1)
elseif ( compteur == 17 )
	AIWander 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100
	MessageBox "Idle 9 , 100"
	set timer to 0
	set compteur to ( compteur + 1)
endif

End
```

Faire activer des objets par les Acteurs

```
AiActivate "ID_objet"
AiActivate , ID_objet, [reset]

	Acteur -> AIActivate "Objet"
```

D'après Bethesda: "Ce package dit à l'Acteur d'activer l'ID de l'objet spécifié. Un package puissant et il faut en convenir sous-utilisé et sous-testé."

Dans le Morrowind de base, cette fonction est souvent buggée, sauf pour ce qui est de faire boire une potion à un PNJ. **Avec Tribunal** il semblerait que ça a été corrigée, au moins dans une certaine mesure. J'ai pu l'utiliser avec succès pour qu'un PNJ ramasse une arme, ouvre une porte normale, passe à travers un marqueur *load door*. *LoadDoors* ne fonctionne que si le *door marker* vers lequel il téléporte est dans la même cellule intérieure, ou à l'intérieur des cellules extérieures chargées (celle du PJ et celles qui l'entourent) – sinon le jeu plante. Je l'ai aussi testé avec un activateur (le bouton pour ouvrir la Porte des Ames).
Les précautions usuelles s'appliquent avec les fonctions d'IA (être sûr que l'Acteur n'est pas trop loin, avoir une bonne grille IA en place, que rien ne bloque le chemin, etc.). Bien que cela n'a pas été testé à fond, il semblerait qu'aucun signal *AIPackageDone* soit émis, mais on peut utiliser d'autres conditions pour affecter un *AIPackage* différent à un PNJ (voir exemples ci-dessus).

Type d'objets	**Activation**
PNJ	Dialogue
Container	Ouverture
Porte	Ouverture
Load door	Ouverture/téléportation (même cellule seulement)
Arme, Armure, divers, etc	Ramasse
Livre/Parchemin	Lecture <ce qui signifie quoi pour un PNJ?>
Activateurs	Exécute ce qui a été défini par le script

Script de démonstration: voici quelques scripts que j'ai fait. Ils montrent comment positionner les conditions pour déterminer quand le PNJ a terminé son action.

```
Begin TT_opendoor ;ouverture d'une porte par un PNJ

short doonce
short AIState ;état de l'IA


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 ); dès que la porte commence sa rotation
		MessageBox "Fait"
		AIWander 30, 5, 0, 0,20,0,0,10,30,0,0
		set doonce to 2
	endif
endif


end
```

```
Begin TT_pickmace ;le PNJ ramasse une masse
short doonce
short AIState  ;état de l'IA
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 ); lorsque le PNJ a la masse dans son
inventaire
                MessageBox "Fait"
                AIWander 512, 5, 0, 0,20,0,0,10,30,0,0
                set doonce to 2
        endif
endif

end
```

```
Begin TT_openloaddoor ; ouverture d'une LoadDoor
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 ); la position a changé. La cible est dans la même cellule
        MessageBox "Fait"
                AIWander 30, 5, 0, 0,20,0,0,10,30,0,0
                set doonce to 2
        endif
endif

end
```

Suivre et escorter

AiFollow, "ID_Acteur", durée_f_enum, x_f_enum, y_f_enum, z_f_enum, [reset]

AiFollowCell, "ID_Acteur", "ID_cellule", durée_f_enum, x_f_enum, y_f_enum, z_f_enum, [reset]

Acteur -> AIFollow, "Mob2ID", 0,0,0,0

Le package IA "*Follow*" (Suivre) force un Acteur à suivre de près un autre Acteur. Vous pouvez l'utiliser afin qu'un PNJ ou une créature suive le joueur, mais aussi pour former une caravane. L'extrait suivant provient d'un de mes scripts : il montre l'utilisation inconditionnelle de la fonction :

```
elseif ( state == 20 ) ;variable d'état à 20
        HB_guar_pack_adros_ -> AIFollow, HB_adros_darani, 0, 0, 0, 0
        AITravel -8144, -19409, 728 ;nouvelles coordonnées -> point 1
        set state to 30 ;variable d'état passe à 30
```

comme aucune destination ou durée n'est donnée, le guar suivra le PNJ tant qu'aucune autre commande n'est donnée. Comme pour les autres commandes d'IA, posez des conditions pour que chaque commande *AIFollow* ne soit appelée qu'une seule fois, et pas à chaque fois que le script est exécuté, c'est-à-dire à chaque frame.
Une fois la durée, l'ID de la cellule et la destination atteintes, l'*AIPackage* se termine (ce que vous pouvez tester avec la fonction *GetAIPackageDone* décrite ci-dessus pour la fonction *AITravel*). La fonction *AiFollowCell* vous permet d'utiliser une cellule intérieure comme destination
La signification de l'option reset est actuellement inconnue.

AIEscort, "ID_Acteur", durée, x, y, z, [reset]
AIEscortCell, "ID_Acteur", "ID_cellule", durée, x, y, z, [reset]

Cette fonction permet de programmer un Acteur afin qu'il guide le joueur vers un point précis. L'Acteur attendra le joueur si la distance devient trop importante, et reprendra sa route

lorsqu'il s'approchera de nouveau (Merci à Kir pour cette info). Vous pouvez en voir un exemple pendant la phase de création de personnage – le garde qui vous escorte de la cale du navire vers la rampe du second niveau (Merci à MisterSmileyFaceDude). La signification de l'option reset est inconnue.

Vérifier quel package IA est actuellement exécuté

Pour scripter des Acteurs complexes, il peut être précieux de savoir quel mode IA est actuellement actif afin de réaliser des actions scriptées dépendantes de cela.

```
GetCurrentAIPackage (renvoie short)

If ( GetCurrentAIPackage == 2 )
        [faire quelque chose]
endif
```

Les valeurs renvoyées sont :

Aucun	-1
Errance	0
Voyage	1
Escorte	2
Suivre	3
Activer	4
Poursuivre	5

Forcer un déplacement discret (sneak)

```
ForceSneak
ClearForceSneak
        "ID_Acteur" -> ForceSneak

GetForceSneak  ( renvoie Booléen/short)
        If ( "ID_Acteur" -> GetForceSneak == 1 )
```

La commande *ForceSneak* place l'Acteur en mode discrétion ; tous les mouvements seront discrets. *ClearForceSneak* met fin au mode *ForceSneak*. Malheureusement, il ne semble pas exister de commande pour forcer la course (ajoutée dans Tribunal). *GetForceSneak* renvoie 1 si l'Acteur appelant est en mode *ForceSneak*. Regardez le script *LookoutScript* pour un exemple. En voici un morceau :

```
elseif ( walkstate == 2 )
        Fargoth->ForceSneak ; entre dans le mode discrétion
        Fargoth->AiTravel -11468.595,-71511.531,173.728     ;va vers l'arbre
        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 ; met fin au mode discrétion
        Fargoth->AiWander 0 0 0 0 0 0 0 0 0

                if ( timer > 3 )
                        Fargoth->ForceSneak ; retourne en mode discret
                        Fargoth->AiTravel -11410.590,-72057.188,133.644     ;va vers le mur
                        set walkstate to 5
```

```
            endif
```

Forcer la course ou le saut : les fonctions de déplacements des PNJs (Tribunal)

ForceRun
ClearForceRun
GetForceRun (short)

ForceJump
ClearForceJump
GetForceJump (short)

ForceMoveJump
ClearForceMoveJump
GetForceMoveJump (short)

Ces fonctions contrôlent toutes les déplacements des PNJs. La fonction *ForceRun* oblige le PNJ à toujours courir lorsqu'il se déplace, la fonction *ForceJump* oblige le PNJ à sauter constamment, et la fonction *ForceMoveJump* oblige le PNJ à sauter lorsqu'il se déplace. Les versions *Get* renvoie 1 si le PNJ spécifié se trouve actuellement dans le mode donné par la fonction et 0 sinon. Les fonctions *Clear* permettent de désactiver le mode correspondant. Un PNJ ne peut se trouver que dans un seul mode à la fois. La priorité est la suivante : Sneak > Running > Jump > MoveJump.

Script de démonstration:

Ce script permet de contrôler, via un objet, le type de mouvement d'un Athlète, un PNJ se déplaçant sans interruption autour d'une place rectangulaire.

```
Begin AthleteControl

short question_posee
short bouton
short isrunning ;PNJ en train de courir ? : oui ou non
short isjumping ;PNJ en train de sauter ? : oui ou non

if ( MenuMode )
   return
endif

if ( OnActivate == 1 )
   set isrunning to ( Athlete->GetForceRun )
   set isjumping to ( Athlete->GetForceMoveJump )
   if ( question_posee == 0 )
      if ( isrunning )
         MessageBox, "Arrêter la course de l'Athlète? "  "Oui"  "Non"
      else
         MessageBox, "Faire courir l'Athlète? "  "Oui"  "Non"
      endif
      set question_posee to 1
   endif
endif

if ( question_posee == 1 )
   set bouton to GetButtonPressed
   if ( bouton == -1 )
   else
      if ( isrunning == 0 )
         if ( bouton == 0 )
            Athlete->ClearForceMoveJump
            Athlete->ForceRun
         endif
      else
         if ( bouton == 0 )
            Athlete->ClearForceRun
         endif
```

```
      endif
      if ( isjumping )
         MessageBox, "Arrêter les sauts de lAthlète? "  "Oui"  "Non"
      else
         MessageBox, "Faire sauter l'Athlète? "  "Oui"  "Non"
      endif
      set question_posee to 2
      set bouton to -1
   endif
endif

if ( question_posee == 2 )
   set bouton to GetButtonPressed
   if ( bouton == -1 )
   else
      if ( isjumping == 0 )
         if ( bouton == 0 )
            Athlete->ClearForceRun
            Athlete->ForceMoveJump
         endif
      else
         if ( bouton == 0 )
            Athlete->ClearForceMoveJump
         endif
      endif
      set question_posee to 0
      set bouton to -1
   endif
endif
end
```

Détecter les actions du joueur : course, saut, discrétion ?

```
[no fix] GetPCSneaking (short)
[no fix] GetPCRunning (short)
[no fix] GetPCJumping (short)

       if ( GetPCRunning )
```

Ces fonctions renvoient 1 si le PJ exécute l'action correspondante et 0 sinon. Comme Morrowind ne possède pas de fonctions pour tester directement les entrées du clavier, ces fonctions fournissent un moyen pour vérifier si le joueur a appuyé sur un certain bouton. En conséquence, elles ont été utilisées pour des motifs de contrôle, par exemple pour des bateaux mobiles, des créatures que l'on peut monter ou dans mon mod escalade.

Script de démonstration :

Lorsque ce script est placé sur un PNJ et que le joueur s'est équipé d'un objet appelé "*scissors*" (paire de ciseaux), des avertissements sous forme de *MessageBox* seront affichées en fonction de l'action courante du joueur.

```
Begin momscript

short warn ;utilisée pour n'afficher les messages qu'une seule fois pendant une action donnée

if ( player->HasItemEquipped "scissors" )
       if ( warn != 1 )
              if ( GetPCRunning )
                     MessageBox "Ne courez pas avec des ciseaux!"
                     set warn to 1
              endif
       endif
       if ( warn != 2 )
              if ( GetPCJumping )
                     MessageBox "Ne sautez pas avec ces ciseaux! Vous pourriez vous crever un
oeil!"
                     set warn to 2
              endif
       endif
```

```
        if ( warn != 3 )
                if ( GetPCSneaking )
                        MessageBox "Vous ne pouvez pas soustraire ces ciseaux de ma vue!"
                        set warn to 3
                endif
        endif
else
        set warn to 0
endif

end
```

Détecter le fait d'être prêt au combat

GetWeaponDrawn (short)
GetSpellReadied (short)

```
        if ( player -> GetWeaponDrawn )
```

On peut utiliser ces fonctions pour déterminer si un Acteur a sorti son arme ou s'il est prêt à lancer un sort.

Script de démonstration : ce script global affiche des messages de notification basés sur l'état des armes et des sorts du joueur.

```
Begin player_notifications

short weapstate ;variable d'état pour les armes
short spelstate ;variable d'état pour les sorts

if ( player->GetWeaponDrawn ) ;si l'arme est sortie
        if ( weapstate != 1 )  ;et qu'elle n'était pas déjà sortie à la frame précédente
                set weapstate to 1
                MessageBox "L'arme du joueur est sortie."
        Endif
Else    ;pas d'arme sortie
        if ( weapstate != 0 ) ;et que le joueur l'avait dans les mains à la frame précédente
                set weapstate to 0
                MessageBox "L'arme du joueur est au fourreau."
        Endif
endif

if ( player->GetSpellReadied )
        if ( spelstate != 1 )
                set spelstate to 1
                MessageBox "Le sort du joueur est prêt à être lancé."
        Endif
else
        if ( spelstate != 0 )
                set spelstate to 0
                MessageBox "Le joueur ne veut plus lancer de sorts."
        Endif
endif
end
```

Faire tomber quelqu'un

Fall

```
        Acteur -> Fall
```

Semble donner un petit coup de pouce à un PNJ qui n'aurait plus de sol sur lequel se tenir. Fait aussi tomber les créatures volantes. Utilisée pour le gars aux parchemins de Vol D'Icare. Lorsque j'ai essayé de l'utiliser sur le joueur pour mon mod escalade, il semblerait que cela téléporte parfois le joueur directement au sol.

Partage d'inventaire et autres fonctions pour les compagnons

[no fix] companion (est locale de type short)

```
short companion
Set companion to 1
```

Tribunal a introduit la possibilité de "partager" l'équipement avec les PNJs ou les créatures. Pour activer cette option, vous devez définir une variable locale de type short nommée "*companion*" et la placer à 1. La placer à 0 désactive le partage. Cette méthode est utilisée pour les mercenaires et les animaux de charges.

[no fix] minimumprofit (est locale de type float)

```
Float minimumprofit
If ( minimumprofit < 0 )
```

Il semble qu'il y ait une autre variable positionnée par le jeu ; il s'agit probablement de la différence entre la valeur courante de tous les biens et de l'or moins la valeur de départ. Si elle devient négative, le mercenaire peut partir.
Script de démonstration : voici la partie correspondante du script de Calvus (le mercenaire de Longsanglot). Cette section gère les changements d'état lorsque le contrat de Calvus s'achève, soit parce que le contrat a expiré, soit parce que le joueur a pris les affaires de Calvus. Le partage est initialisé via le dialogue (en affectant 1 à *companion*), et non pas dans le script lui-même

```
if ( GetJournalIndex Merc_Calvus_Quit < 1 ) ;si Calvus n'est pas encore parti
        if ( Contract_Calvus == 1 )          ;si Calvus a un contrat
                if ( minimumProfit < 0 )     ;Calvus part car le joueur a pris ses affaires
                        AiWander 128 6 0 40 30 20 0 0 0 0 0 0 0
                        Set Companion to 0; arrêt du partage
                        StopScript Contract_Calvus
                        Set Contract_Calvus to 0
                        ForceGreeting
                        return
                else
                        if ( Contract_Calvus == 0 )   ;gère Calvus après qu'un contrat a expiré
                                AiWander 128 6 0 40 30 20 0 0 0 0 0 0 0
                                Set Companion to 0; arrêt du partage
                                if ( GetJournalIndex Merc_Calvus < 10 )
                                        Journal Merc_Calvus 10;le premier contrat a expiré
                                else
                                        Journal Merc_Calvus 20; un contrat plus récent a expiré
                                endif
                        endif
                endif
        endif
endif
```

[no fix] StayOutside (est locale de type float)

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

Une variable très utile pour la gestion des compagnons. Utilisée dans un script; elle force celui à qui elle a été assignée à rester (et attendre) à l'extérieur de n'importe quelle cellule intérieure où rentrerait le joueur (il revient automatiquement au retour du joueur).(Forum info / Grumpy)

Race, Faction et Rang dans la Faction

(traduction par Emma Indoril)

Déterminer la race

GetRace, "RaceID" (renvoie Booléen/short)

```
Player->GetRace "Dark Elf"
```

Renvoie 1 si l'objet ciblé par le script (PJ/PNJ) appartient à la race indiquée par "*RaceId*".

Exemple : Voici un script global, utilisé par Bethesda afin de définir une variable (PCRace) qui sera utilisée dans les dialogues afin de déterminer la race auquel appartient le PJ:

```
begin RaceCheck

;script global qui ne s'exécute qu'une fois pour vérifier la race du PJ afin qu'elle puisse
être utilisée dans les dialogues

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
```

Déterminer le statut du PJ au sein de sa faction :

[no fix] GetPCRank, FactionID_enum (renvoie short)

```
if ( GetPCRank "Telvanni" == 9 )
```

Renvoie le Rang occupé par le PJ au sein d'une faction. Par défaut, utilise la faction de l'interlocuteur si FactionID n'est pas défini. Renvoie un nombre entre 0 et 9, et -1 si le PJ ne fait pas partie de la Faction.

Exemple : Un Acteur/Objet ciblé par ce script ne sera visible dans le jeu (`enable`) uniquement si le PJ ne fait pas partie de la Maison Rédoran.

```
Begin bandenIndarysScript
if ( CellChanged == 0 )
        Return
endif

if ( GetPCRank "Redoran" == -1 )
        Enable
else
```

```
        Disable
endif
End
```

[no fix?] GetPCFacRep, [FactionID]   (renvoie short?)

Renvoie probablement la réputation du PJ au sein de sa Faction. Commande non utilisée par Bethesda, et non testée.

SameFaction   (renvoie Booléen/short)

Renvoie 1 si le PJ et de la même Faction que le PNJ auquel le script est rattaché.

PCExpelled ["factionID"]   (renvoie Booléen/short)

Renvoie 1 si le PJ a été renvoyé de la Faction auquel appartient le PNJ appelant la fonction ; une faction ; cette fonction peut également être utilisée avec une faction spécifique, différente de celle du PNJ. Pour un exemple de script, voir ci-dessous avec la fonction *PCClearExpelled*.

Modifier l'appartenance à une Faction et les réactions

[no fix] PCJoinFaction ["FactionID"]

Cette fonction fait du PJ un membre d'une Faction spécifique.
Le terme « *FactionID* » est optionnel. S'il n'est pas utilisé dans le script, Le Jeu utilisera la Faction du PNJ qui appelle la fonction.

LowerRank
RaiseRank

Elève ou abaisse le rang de la personne dans la Faction auquel elle appartient. Dans le jeu de base, souvent utilisé dans les dialogues.

[no fix] PCLowerRank
[no fix] PCRaiseRank

Elève ou abaisse de 1 le rang du PJ dans la Faction à laquelle appartient le PNJ sur lequel est calqué le script Si le PJ ne fait pas partie de la Faction concernée, le rang sera fixé à 1.

Exemple:

```
Begin treboniusScript
;si vous êtes dans l'Arène et que la quête du maître de guilde
;(guildmaster quest) est active...
;met à jour le journal et augmente le rang du joueur
;lorsque Trebonius meurt
short doOnce
short nolore

if ( doOnce == 1 )
        Return
endif
if ( GetJournalIndex MG_Guildmaster < 50 )
        Return
endif
if ( GetPCCell "Vivec, Arena" == 0 )
        Return
endif
if ( duelActive == 0 )
        Return
endif
if ( OnDeath == 1 )
```

```
        Set DuelActive to 0
        Set doOnce to 1
        PCRaiseRank "Mages Guild"
        PCRaiseRank "Mages Guild"
        Journal MG_Guildmaster 100
endif
End
```

[no fix] PCExpell ["FactionID"]

Renvoie le PJ de sa Faction.

[no fix] PCClearExpelled ["FactionID"]

Enlève le marqueur “Renvoyé de sa Faction”. Autrement dit : réintègre le PJ dans sa Faction.

Exemple : Ce script réintègre le PJ dans sa Faction, après qu’un certain temps se soit écoulé.

```
Begin expelledMG

;ce n’est qu’un modèle
;supposé être placé sur un objet présent dans chacune des guildes des Mages.

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
```

[no fix] ModPCFacRep, var_enum, ["FactionID"]
[no fix] SetPCFacRep, var_enum, ["FactionID"]

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

Définit ou modifie la réaction d’une Faction spécifique envers le PJ.

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

Définit ou modifie la réaction d'une faction envers les membres d'une autre faction.

Exemple : Voici un extrait du script « MoonAndStar ». Cette partie du script a deux fonctions :

- Le PJ intègre la Faction « Nérévarine »
- Le script modifie les réactions des Factions « Maison Rédoran » et « Temple des Tribuns » envers la Faction « Nérévarine ».

```
;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
```

Déterminer et Modifier la Réputation et la Disposition

***Get/Mod/Set*Reputation**

***Get/Mod/Set*Disposition**

Fait probablement référence à la disposition de base (comme définie dans le TES CS, sans aucun modificateur appliqué)

Fonctions Spécifiques aux Loups-Garous

Changer d'attributs

SetWerewolfAcrobatics

```
Acteur -> SetWerewolfAcrobatics
```

Cette fonction change les attributs d'un objet (PNJ), pour ceux des Loups Garous. Cela modifie les talents et attributs, pour les faire correspondre au réglage *fWerewolfXXXX*. Dans la plupart des cas, cela signifie des hautes valeurs en Force, Agilité, Acrobatie etc…et les autres caractéristiques à 0.

```
Player -> AddSpell "werewolf vision"
Player -> AddSpell "werewolf regeneration"
Player -> SetWereWolfAcrobatics
```

Changer la couleur de Secunda

[no fix] TurnMoonWhite
[no fix] TurnMoonRed

Ces deux fonctions ont un rôle très simple : elles changent la couleur de Secunda (la petite lune blanche), la faisant passer de blanche à rouge, puis à nouveau blanche.
Cela n'a aucune incidence directe sur le déroulement du jeu, c'est juste un effet esthétique.

Cet effet est utilisé au cours de la quête principale dans Bloodmoon, d'où le titre !

```
if ( doOnce == 0 )
	TurnMoonRed
	set doOnce to 1
endif
```

Déterminer le nombre de victimes d'un Loup-Garou

[no fix] GetWerewolfKills (renvoie short ?)

Cette variable tient le compte du nombre de PNJ tué par le Loup-Garou. Chaque fois que le PJ, sous sa forme de Loup Garou, tue un PNJ, on rajoute 1 au compteur.
Le compteur est remis à 0 lorsque le PJ reprend forme humaine.

```
if ( GetWerewolfKills > 0 )
	; faites du code pour apaiser la faim du PJ
endif
```

Vérifier si une créature est sous forme de Loup-Garou

IsWerewolf

```
	If (Acteur -> IsWerewolf ]
```

Cette fonction permet de déterminer si l'acteur ciblé par le script est sous forme de Loup-Garou. La fonction peut être utilisée indifféremment sur le PJ ou d'autres créatures.

```
if ( Player->IsWerewolf != 1 ) ;NE PAS EXECUTER SI LE JOUEUR N'EST PAS SOUS FORME DE LOUP-
GAROU
	return
endif
```

Se transformer en Loup-Garou

BecomeWerewolf
UndoWerewolf

```
	Acteur->BecomeWerewolf
	Acteur->UndoWerewolf
```

Ces fonctions transforment la cible en loup-garou et la rétablit dans sa forme originale.

IMPORTANT : Utiliser cette fonction de manière inconsidérée peut endommager votre partie. Certaines quêtes et variables dépendent de l'utilisation de ces fonctions…
Donc si vous voulez faire joujou avec la console, et vous transformer n'importe quand en Loup-Garou… Vous êtes prévenus, ne venez pas pleurer après… *(Ce message vous a été proposé par votre développeur et ami WormGod).*

```
if ( OnPCEquip == 1 )
        Player->BecomeWereWolf
        Set OnPCEquip to 0
Endif

Set timer to ( timer + GetSecondsPassed )

If ( timer > 10 )
        Player->UndoWereWolf
Endif
```

Variables globales spécifiques aux Loup-Garous

[no fix] PCknownWerewolf (variable globale de type short)

Variable globale indiquant si le PJ est connu comme étant un Loup-Garou

[no fix] PCWerewolf (variable globale de type short)

Est mise à 1 lorsque le PJ est transformé en Loup-Garou. Utilisée pour le contrôle de nombreux scripts.

[no fix] WerewolfClawMult (variable globale de type float)

Multiplicateur pour les dégâts causés par les griffes des Loup-Garous
Se référer aux scripts du Loup-Garou pour plus de détails.

Texte et Dialogue

Tutorial très bref

Le dialogue est un art en lui-même qui ne peut pas être entièrement traité ici. Cependant, en créant des quêtes, vous aurez souvent à lier les dialogues et les scripts pour la contrôler et obtenir certains effets. C'est pourquoi je vais faire une brève introduction.

Le concept de dialogue dans MW

Dans Morrowind, le dialogue est organisé en base de données. Celle-ci est structurée de la façon suivante :

Niveau supérieur :

Les différentes "divisions" du dialogue sont :

Topics: les réponses du sujet actuel dans la fenêtre de dialogue du jeu
Greetings: les phrases de bienvenue lorsque vous entamez le dialogue avec un Acteur.
Persuasion: les réponses obtenues après vos tentatives de persuasion (succès ou échecs).
Journal: les entrées de votre journal.
Voices: les fichiers sonores .mp3 (et les sous-titres) que les Acteurs prononcent lorsque vous vous approchez d'eux, lorsqu'ils sont blessés, fuient etc.

Sous-catégorie 1 :

Chacune des divisions du niveau supérieur possède des sous-catégories que j'appelle sujets – pour la division *Topics*, ce sont les "mots-clefs" (topics) dont les Acteurs ont connaissance. Ce sont les mots en surbrillance ("les liens hypertextes") dans les textes in-game et listés dans la partie droite de la fenêtre de dialogue. Pour *Journal,* ce sont les différents journaux (normalement un par quête). Pour *Voices* ce sont les différentes catégories de réponses sonores qui se "déclenchent" en fonction des conditions in-game appropriées etc. Pour *Persuasion,* ce sont les refus de services ou d'infos ou les messages de succès/d'échecs des tentatives de persuasion. Pour *Greetings,* ce sont les catégories générales de greetings (maladie, offre de quête...). voici comment Bethesda les a utilisé (forum info / Emma) :

- Greetings 0: le PNJ est alerté
- Greetings 1: Quêtes (les quêtes où le fait d'être un vampire, un criminel, un exhibitionniste, malade n'est pas important).
- Greetings 2: le joueur est un vampire/est nu
- Greetings 3: Traîtres à la Morag Tong
- Greetings 4: Crime et maladie
- Greetings 5: Quêtes
- Greetings 6: Factions
- Greetings 7: Classes, Fin du jeu, Esclaves
- Greetings 8: Vêtements (les greetings généraux concernant l'apparence du joueur)
- Greetings 9: Lieux.

Sous-catégorie 2 :

Pour chaque sujet, il peut y avoir une ou plusieurs sous-entrées – ce sont les réponses réelles. Pour *Topic* et *Greetings,* ce sont les réponses réelles des Acteurs à la phrase du topic. Pour *Journal,* ce sont les différentes entrées décrivant la progression de la quête. Ces entrées existe

sous forme de liste chaînée, ce qui signifie que chaque entrée contient un lien (invisible) vers la prochaine réponse et vers la réponse précédente (on obtient communément un message d'erreur lorsqu'un dialogue est effacé ou transformé, par exemple en nettoyant un mod avec TESAME ou en chargeant plusieurs mods qui changent le même topic. Le jeu n'est plus certain de l'ordre des entrées. Cela n'a parfois pas d'importance mais cela signifie aussi que des réponses sont isolées parce que l'ordre n'est plus le bon – cela dépend des conditions.
Ces sous-entrées s'accompagnent de conditions que vous pouvez modifier dans la fenêtre d'édition de dialogue. Dans la section *Speaker Condition*, vous trouverez une liste de conditions générales sur la gauche – vous pouvez y définir quel Acteur (ID de l'Acteur) ou groupe d'Acteurs (Race, Classe, Disposition...) connaît potentiellement cette réponse. Il y a aussi 2 conditions sur le PJ (faction et rang). À droite, vous pouvez utiliser un maximum de 6 conditions "libres" faisant référence à l'Acteur, au PJ, à d'autres choses comme l'état de variables globales – beaucoup d'options : à voir par vous-même. Voyez par vous-même les entrées de journal, les stats du joueur, les variables locales ou globales, les objets de l'inventaire et les nombreuses autres fonctions, certaines équivalentes aux fonctions utilisées dans les script, d'autres uniquement spécifiques au dialogue (voir ci-dessous).

Une importante mais déroutante caractéristique de l'éditeur de dialogue est l'option filtre (en bas à gauche). Lorsque vous sélectionnez une ID d'Acteur, vous ne voyez plus que les sujets que cet Acteur peut potentiellement connaître (comme décrit ci-dessus). Souvenez-vous en lorsque que vous créez un **nouveau** topic (peut-être spécialement pour ce PNJ) : il ne contient aucune réponse. Ainsi l'Acteur ne peut pas le "connaître" et donc le topic nouvellement créé n'apparaît pas ! Sélectionnez le champ vide au tout début de la liste pour revoir tous les topics, construire une réponse pour le nouveau topic que l'Acteur puisse "connaître" – ensuite vous pouvez utiliser le filtre de nouveau, si vous le souhaitez.

Comment fonctionnent les dialogues

Pour savoir si un **topic** de dialogue avec un PNJ est disponible, le jeu vérifie :

1. si le PNJ "connaît" le topic – cela est déterminé par l'ensemble des conditions du champ "speaker condition" – si le PNJ peut satisfaire les conditions d'au moins une réponse de ce topic, il le "connaît" it.
2. si le joueur le connaît. Le PJ connaît un topic ou bien s'il (le mot correspondant au topic) est apparu précédemment dans une réponse donnée par un PNJ ou s'il a été spécifiquement ajouté avec la fonction *AddTopic*. Notez que l'ajout d'un topic à la liste de ceux que connaît le PJ ne se produit que si le mot-clef devient disponible dans le même dialogue avec le PNJ qui prononce le mot-clef. Un exemple : vous avez du skouma dans votre inventaire et un armurier vous refuse toute transaction. Le mot "skouma" apparaît dans le dialogue et il existe un topic "skouma", mais il n'est pas connu des armuriers. Après cela, vous parlez à un érudit qui *connaît* le sujet...mais vous, toujours pas. Le topic apparaît dans votre liste seulement lorsqu'il (ou quelqu'un d'autre qui connaît ce sujet) utilise le mot skouma dans une de ces réponses (Forum info / Kir).
3. si le PJ et l'Acteur "connaissent" le topic, il apparaîtra dans la liste des topics et sera mis en surbrillance dans le texte – il peut maintenant être sélectionné par le joueur et on lui donnera une réponse appropriée.

Lorsque le jeu doit sélectionner la réponse correcte de la liste de réponses de ce topic, il fait les choses suivantes :

- Il vérifie, en commençant **en haut de la liste,** si les conditions pour une réponse sont atteintes, ce qui signifie que **toutes** les conditions définies pour cette réponse doivent renvoyer "vrai".

- Si oui -> cette entrée est utilisée pour l'affichage de la fenêtre de dialogue.
- Si non -> le jeu passe à l'élément de la liste qui suit.

Règle spéciale pour les greetings : si aucun des éléments de la liste ne convient, on passe à l'objet de niveau 1 suivant (par exemple : si aucun greeting de "Greeting 0" n'est vrai, on passe à ceux de "Greeting 1").
Règle spéciale pour le Journal : il n'y a qu'une condition ici, l'index (appelé par la fonction `Journal`). Il doit s'agir de la valeur exacte.

Une fois qu'une réponse a été choisie, le jeu

- Affiche le texte dans la fenêtre de dialogue (ou joue un mp3 pour les réponses *Voice*)
- Exécute les commandes de scripts placées dans le champ ***Result*** (tout en bas). Vous pouvez vous servir de toutes les fonctions de scripts. Les conditions (commande if) sont aussi autorisées (Merci à Kir et Manauser pour cette info).

Soyez conscient que le champ *Result* vous permet d'échanger des informations avec les scripts (par exemple en affectant des variables ou en ajoutant des entrées de journal) et que vice versa les scripts peuvent influencer les dialogues en plaçant des conditions qui seront testées (par exemple vous pouvez vérifier des variables locales et globales dans le champ Speaker Condition). Le script le plus simple qui influence les dialogues est le script nolore qui sert juste de flag afin que les Acteurs continuent d'utiliser leur dialogue standard.
Note : les scripts du champ *Result* ne sont pas compilés par le CS. Vous pouvez y écrire n'importe quelles idioties et MW ne s'en plaindra pas jusqu'à ce que la réponse soit prononcée par un PNJ et le script compilé à la volée. Si le script est suffisamment complexe pour avoir des inquiétudes par rapport à la syntaxe, il vaut mieux copier/coller son code dans la fenêtre standard et essayer de le sauver. C'est plus sûr que d'utiliser le bouton "*Error Test/Check Results*", car cela peut changer les valeurs prédéfinies des variables globales. D'un autre côté, un script qui n'est compilé qu'à l'exécution permet quelques effets intéressants, comme adresser le contenu d'un autre mod sans avoir à le dupliquer dans le mod courant, ce qui est impossible dans les scripts conventionnels (Forum info / Kir).

Quelques règles d'or

- **Les réponses les plus spécifiques doivent être au tout début de la liste, les réponses par défaut doivent être en bas !** Souvenez-vous que la première qui renvoie vrai est sélectionnée. Vous ne devez donc pas avoir une réponse concernant tous les habitants de Vivec au-dessus d'une ne concernant qu'un PNJ spécifique de Vivec.
- **Si vous voulez qu'un PNJ soit capable de parler de quelque chose de spécial avec le PJ, vous devez introduire le mot du topic, par exemple dans un greeting ou dans les "dernières rumeurs ".** Sinon, vous pouvez aussi utiliser un script appelant la fonction *AddTopic*.
- **N'utilisez pas des mots normaux pour les topics de journal.** Topic, Greeting, Journal sont en fait dans la même base de données – c'est pourquoi les topics de journal utilise un format tel que A1_dreams. Si cela avait été seulement "*dreams*", le mot "*dreams*" aurait pu être considéré comme une réponse de dialogue.
- **N'effacez jamais un topic qui appartient au jeu original Morrowind/Tribunal.** C'est très difficile à réparer et causera des erreurs sévères dans les sauvegardes des joueurs. (Emma)
- **Si vous utilisez la section Greeting, ne placez pas vos propres greetings au début de celle-ci.** Les greetings supérieurs appartiennent à une certaine quête et doivent être

laissés au tout début afin qu'ils s'affichent toujours. Vous pouvez placer vos greetings en-dessous. (Emma)

Dialogue 101

Ce qui suit résume la plupart des problèmes les plus fréquents rencontrés avec les dialogues. Cette liste a été conçue à partir d'une discussion sur un forum avec les contributions de Klinn, Emma et GarryB.

Tip 1) **Mes nouveaux topics ont disparu !** regardez le champ ***Filter*** situé en-dessous de la liste des topics. Ne mettez aucun filtre en sélectionnant la ligne vide en tête de liste. Pour appeler l'éditeur de dialogue, il est conseillé d'utiliser le bouton de la barre d'outils principale plutôt que celui de la fenêtre des propriétés des PNJs.

Tip 2) **Mon PNJ s'acharne à poser la même question encore et toujours !** Assurez-vous que vous avez placé les réponses *au-dessus* de la question. Cela peut sembler bizarre mis c'est ainsi que cela fonctionne.

Tip 3) **Mon PNJ connaît les sujets courants !** Pour empêcher un PNJ d'avoir les topics standards à propos des traditions de Morrowind, attachez-lui le script "*NoLore*". Si vous avez déjà un script d'attaché sur ce PNJ, ajoutez **Short NoLore** dans vos déclarations de variables.

Tip 4) **Mon PNJ possède toujours des topics supplémentaires !** Quelques topics généraux peuvent apparaître selon la faction ou la classe du PNJ. Par exemple les membres de la Légion Impériale possèderont automatiquement les topics concernant la faction, l'Empire, et d'autres.

Tip 5) **Comment ajouter des topics pour un PNJ en particulier ?** Après avoir créer le topic et ses ***Info/Responses***, dans la zone ***Speaker Conditions***, mettez l'ID de votre PNJ.

Tip 6) **J'ai ajouté des topics mais mon PNJ ne les a pas !** Deux possibilités : le PJ doit déjà avoir entendu (lu) le mot ou l'expression du topic avant de pouvoir le questionner à ce sujet. On le fait habituellement en incluant le topic dans le greeting du PNJ. Deuxième possibilité : certaines Speaker Conditions peuvent empêcher l'apparition du topic. Même si vous filtrez le dialogue pour un PNJ, certains topics dépendent de l'avancement du joueur dans le jeu, d'une entrée de journal spécifique, etc...

Tip 7) **Comment changer l'ordre de mes réponses ?** Utilisez les flèches gauche et droite de votre clavier pour bouger un ***Info/Response*** en haut ou en bas de la liste.

Tip 8) **Comment créer un dilaogue pour une créature ?** N'importe quelle créature peut avoir son propre dialogue. On le fait de la même façon que créer un dialogue pour un PNJ, à une différence près.
Vous devez créer le dialogue SANS AUCUN FILTRE ACTIF (c'est-à-dire que le champ sous les topics doit rester vide). Une fois créé, vous pouvez filtrer le dialogue pour votre créature.

Tip 9) **Quelles sont les utilisations typiques du champ dialogue *Result* ?** Emma a listé les commandes les plus utiles et les plus fréquemment utilisées :

- Player->AddItem "mon objet" 1 (un objet spécifique est ajouté à l'inventaire du joueur)
- Player->RemoveItem "mon objet" 1 (un objet spécifique est enlevé de l'inventaire)
- ModDisposition 5 (le PNJ aimera davantage le joueur de 5 points)
- cast "mon_sort" player (le PNJ lancera un certain sort)

- AiFollow Player 0 0 0 0 0 (le PNJ commence à suivre le joueur)
- AiWander 0 0 0 0 0 0 0 0 0 0 0 0 0 (le PNJ arrête de suivre le joueur)
- SetFight 100 (le PNJ commence à attaquer le joueur)
- StartCombat->player (le PNJ commence à attaquer le joueur)
- StopCombat (ouaip, vous avez deviné : arrête le combat)
- StartScript "mon_script_global " (commence un certain script)
- Set companion to 1 (si vous avez ajouté "short companion" à votre script, le PNJ partagera ses biens avec vous ; **Tribunal requis**)
- SetHealth 100 (place la santé du PNJ à 100 – la même commande peut être utilisée pour placer les autres caractéristiques et compétences/talents, par exemple SetMagicka, SetLongBlade etc.)
- disable (le PNJ disparaît instantanément)
- goodbye (force le joueur à terminer la conversation. Peut être utile pour le cas où il faut éviter de continuer la conversation alors que le PNJ a été désactivé avec la commande disable)

Tip 10) **J'ai créé beaucoup de dialogues, comment puis-je vérifier les fautes de frappes et cie ?** Vérifier la syntaxe et les fautes de grammaire peut se faire aisément en utilisant les fonctions import et export du Construction Set. Exportez le "nouveau" dialogue vers un fichier, utilisez votre éditeur de texte favori pour effectuer vos corrections et importez le dialogue corrigé. C'est plus facile que de circuler dans une myriade de topics, greetings et entrées de journal. (Forum info / GarryB)

Fonctions se rapportant au dialogue

Un grand nombre des fonctions qui suivent ne s'utilisent pas seulement dans les scripts mais aussi dans le champ ***Result*** de la fenêtre d'édition de dialogue.

Afficher des messages

```
[no fix] MessageBox, "Message", [var1], [var2], ["bouton1"], ["bouton2"]
```

La commande *MessageBox* permet d'informer le joueur. Normalement il s'agit d'un petit cadre avec du texte qui s'affiche en bas de l'écran pendant quelques secondes ou jusqu'à ce que le joueur clique sur un bouton si elle en contient. Il y a une limite de 9 boutons par *MessageBox*. Si la fenêtre de dialogue est ouverte, le contenu de la *MessageBox* s'affiche à l'intérieur ! il sera dans une couleur différente pour montrer que le texte ne fait pas partie du dialogue. Par exemple, "Ok, je vais lever la malédiction. *Il lève la malédiction.*" *MessageBox* possède différents modes d'opérations. La plus simple est l'affichage d'un message en bas de l'écran de jeu pendant quelques secondes ; le script suivant affiche un message lorsque l'objet auquel il est attaché est équipé :

```
Begin informplayer
Short OnPCEquip       ;variable indiquant si l'objet est équipé ou non

if ( MenuMode == 1 )
       return
endif

if (OnPCEquip ==1 ) ;l'objet est équipé
       MessageBox "L'épée vibre dans votre main"
       Set OnPCEquip to 0
Endif

End informplayer
```

Dans le second mode d'opération, le message reste à l'écran jusqu'à ce que le joueur presse un bouton :

```
MessageBox "Ulyah lève ses mains et énonce la formule. Vous allez maintenant être transporté
dans la région de Shéogorad", "ok"
```

Dans le troisième mode d'opération, vous pouvez utilisez les *MessageBox* pour obtenir une décision de la part du joueur grâce aux boutons de la *MessageBox* et la fonction *GetButtonPressed* :

[no fix] GetButtonPressed (renvoie short)

Renvoie le bouton pressé si la *MessageBox* en contient ; on commence à 0. Renvoie -1 tant qu'aucun bouton n'a été pressé.
Script de démonstration :

```
Begin choices

Short bouton
Short statut
Short OnPCEquip ;à déclarer en tant que variable, sinon il y aura des erreurs

if ( OnPCEquip ==1 )
	MessageBox, "L'épée vibre dans votre main. Voulez-vous vraiment vous en équiper ?",
"oui", "non"
	Set OnPCEquip to 0 ;pour n'afficher la MessageBox qu'une seule fois
	Set statut to 1
Endif
If ( statut == 1); attente de la décision des joueurs
	Set bouton to GetButtonPressed
	If ( bouton == -1 ); pas de bouton sélectionné : ne rien faire
		return
	Elseif ( bouton == 0 ); continue normalement
		Set statut to 0 ; réinitialise pour la prochaine fois
	Elseif ( bouton == 1 )
		Player -> drop, "power_sword" ; force le joueur à lâcher son arme
		Set statut to 0
	Endif
Endif

End
```

Note : si vous utilisez les scripts de démarrage de Tribunal pour afficher une *MessageBox* avec un bouton dès que le jeu est chargé, vous devez laisser un délai d'une frame, sinon le pointeur de la souris ne s'affichera pas (Forum info / DinkumThinkum).
Vous pouvez entrer un retour de chariot dans les *MessageBox*, mais il faut éditer l'esp avec un éditeur hexadécimal. Mettez quelques caractères non usuels comme || et sauvez l'esp. Ensuite éditez l'esp, recherchez vos caractères || et remplacez-les par 0D0A (le codage hexadécimal pour le retour chariot).(Forum info / qarl)

Afficher des variables et des *text defines* dans une *MessageBox*

Pour afficher des variables dans une *MessageBox*, vous devez utiliser une syntaxe décrivant le format du nombre à afficher. ATTENTION – le fichier d'aide original contient de nombreuses erreurs à ce sujet !

```
MessageBox "Il vous reste %.0f jours", jours_restants
```

Le symbole % indique la variable. Le nombre après le point détermine le nombre de décimales affichées. "f" signifie que la variable est de type float. Le fichier d'aide liste quelques types (f pour float, D pour short et long, et S pour les string) : parmi ceux-ci, je n'ai réussi qu'à faire marcher f. Cependant, %g et %G marche parfaitement pour les variables de

type short et long (merci à Niyt Owl). Vous pouvez écrire %.3g, mais les décimales seront tout simplement ignorées. Les désignateurs ne sont pas vraiment spécifiques au type de la variable : %.3f affichera aussi des variables de type short ou long.
Les variables de type string sont mentionnées dans le fichier d'aide, mais à ma connaissance ne sont pas implémentées ; vous pouvez cependant utiliser les *text defines* du dialogue dans les *MessageBox* mais n'utilisez PAS % : pour les *text defines*, dans les scripts, on utilise ^ à la place (merci à Ragnar_GD):

***Text defines* :**

```
^PCName Le nom du joueur
^PCClass       La classe du joueur
^PCRace La race du joueur
^PCRank Le rang du joueur dans la faction de l'interlocuteur
^NextPCRank    Le rang suivant du joueur dans la faction de l'interlocuteur

^Cell          La cellule dans laquelle se trouve le joueur
^Global La valeur de n'importe quelle variable globale. Les Floats s'affiche en 1.1, comme pour
^Gamehour
```

Note: vous pouvez aussi afficher une variable globale normalement en utilisant la syntaxe %.1f, ce qui produira le même résultat. Si vous utilisez le text define ^Global dans un livre, Morrowind risque de crasher si vous accédez ou modifiez la variable globale lorsque le livre est ouvert. À éviter à tout prix ! (Forum Info/Chris_K)

```
^Name          Le nom de l'interlocuteur
^Race          La race de l'interlocuteur
^Class  La classe de l'interlocuteur
^Faction       La faction de l'interlocuteur. S'il n'appartient à aucune faction, ce sera un
blanc
^Rank          Le rang de l'interlocuteur
```

Note : les éléments de cette liste ne fonctionneront comme ils le feraient dans un dialogue car les text defines font référence aux valeurs du PJ par défaut, et pas à celles de l'Acteur appelant. ^Name et ^PCName afficheront tous les deux le nom du PJ.

Script d'exemple : Un script tout bête montrant toutes les syntaxes possibles :

```
Begin test1

short var_1
long var_2
float var_3
; GameHour est une variable globale de type float
;on attribue des valeurs bidons aux variables
set var_1 to 1
set var_2 to 2
set var_3 to 3


MessageBox "^PCName, vous avez %g tête, %G mains et %.5f ors. Certains pourraient dire qu'il
se fait tard ici : ^cell. Il est ^GameHour ou plus exactement %.2f !", var_1, var_2, var_3,
GameHour

End
```

Ajouter un topic

```
[no fix] AddTopic, "Topic"

       AddTopic, "Topic"
```

Une fois que vous avez construit votre topic de dialogue dans le TESCS, vous vous rendrez compte que vous ne pouvez toujours pas parler de ce sujet avec le PNJ en question car, pour

le jeu, vous ne connaissez toujours pas ce topic. Il y a deux moyens de changer cela : ou vous introduisez le topic dans un autre sujet de conversation (par exemple, un greeting personnalisé) ou vous vous servez d'un script, ce qui est sensé s'il s'agit d'un topic évident dont le joueur voudrait s'enquérir sans qu'il ait été amené par la conversation. Par exemple, si vous voyez un PNJ placé sous une chute d'eau, vous voudrez sûrement lui demander "n'êtes-vous pas trempé?" même si celui-ci ne vous en parle pas.
Pour faire cela, attachez simplement un petit script au PNJ :

```
Begin AddSpecialDialogue

AddTopic, "aren't you getting wet?"

End AddSpecialDialogue
```

Note : vous devez avoir défini un topic possédant cet ID avant de faire le script, sinon le compilateur s'en plaindra.
Vous ne pouvez pas supprimer un topic via les scripts ; cependant, vous pouvez placer une condition se référant à une variable locale du script dans les ***Speaker Condition*** de l'éditeur de dialogue ; elle sera utilisée pour remplir cet effet.
AddTopic ajoute le topic à la liste des topics connus (*"known topics"*) du **joueur.** Utiliser "ID_Acteur"->AddTopic "blabla" est donc faux.

Commencer et terminer un dialogue

ForceGreeting

ForceGreeting peut être utilisé afin que les Acteurs commencent le dialogue. Lorsque *ForceGreeting* est appelé, la fenêtre de dialogue s'ouvre et l'Acteur utilisera un greeting en fonction des paramètres de son dialogue. Par conséquent, si vous souhaitez que l'Acteur commence par un greeting spécial, vous devez d'abord le définir via l'éditeur de dialogue du TESCS. La fonction ne tient pas compte de l'endroit où se trouve le PNJ ; elle fonctionnera toujours ; il vaut donc mieux la coupler avec une condition *GetDistance* ou *GetPCCell* .

Script d'exemple : ce script utilise un bel ensemble de conditions avant d'initier la commande *ForceGreeting*

```
Begin balynScript ; concerne la quête d'empoisonnement donnée au siège de la Morag Tong
                  ; attaché à Balyn Omavel
float timer
short doOnce

if ( GetJournalIndex "DA_Mephala" < 40 ) ;avancement de la quête insuffisant
       Return
endif
if ( GetJournalIndex DA_Mephala >= 60 ) ;quête trop avancée ou terminée
       Return
endif
Set timer to ( timer + GetSecondsPassed )
if ( timer < 5 ) ; le script ne s'exécute que toutes les 5 sec.
       Return
endif
Set timer to 0

if ( doOnce == 0 )
       if ( GetDistance Player <= 1024 ) ;joueur suffisament proche du PNJ
              if ( player->GetDistance "hlaalu_loaddoor_ 02_balyn" <=256 )
              ;joueur proche de la porte du PNJ
                     if ( GetLOS Player == 1 ) ;joueur dans la ligne de vue du PNJ
                            ForceGreeting
                            Journal DA_Mephala 55
                            set doOnce to -1
                     endif
              endif
       endif
```

```
endif

End
```

[no fix?] Goodbye

Goodbye oblige le joueur à mettre fin au dialogue. Après que cette fonction ait été appelé (plus souvent dans la section ***Result*** du topic que dans les scripts), le PJ ne dispose que l'option *Goodbye* (Au revoir) et ferme ainsi la fenêtre du dialogue.

Forcer le dialogue pour un joueur lycanthrope (Bloodmoon)

[no fix] AllowWereWolfForceGreeting (variable de type short)

```
        Short AllowWereWolfForceGreeting
```

Cette variable permet d'utiliser *ForceGreeting* pour un joueur en forme de loup-garou. Utilisé pour les scripts : cariushuntscript, dulkscript, heartfanghuntscript. Apparemment il suffit de déclarer la variable ; pas besoin de la placer à une valeur spécifique.

Exemple:

```
Begin dulkScript

short doOnce
short playerwolf
short AllowWerewolfForceGreeting

if ( GetJournalIndex BM_FrostGiant2 == 10 )
        if ( doOnce == 0 )
                if ( GetDistance Player <= 512 ) ; joueur proche
                        if ( Player->IsWerewolf == 1 ) ; joueur  en forme de loup-garou ?
                                ForceGreeting
                                set doOnce to 1
                        endif
                endif
        endif
elseif ( GetJournalIndex BM_FrostGiant2 == 70 )
        if ( doOnce == 1 )
                if ( GetDistance Player <= 512 )
                        ;if ( Player->IsWerewolf == 1 )
                                ForceGreeting
                                set doOnce to 2
                        ;endif
                endif
        endif
endif

End dulkScript
```

Questions à choix multiples

[no fix] Choice, "choix 1", choix1_enum ["choix 2", choix2_enum, …]

```
        Choice "oui", 1, "Non, certainement pas!", 2
```

(Dialogue seulement!)
Fonction utilisée dans le champ ***Result*** afin de poser une question au joueur ou pour "continuer" un discours trop long. Une fois que le joueur aura fait son choix, le même topic sera vérifié à nouveau ; la bonne réponse sera affichée en vérifiant les conditions function / choice / = / choice_enum dans les ***speaker conditions*** de l'éditeur. Pas de signification dans les scripts à ma connaissance.

Info supplémentaire de Riiak : contrairement aux *MessageBox*, on peut encapsuler plusieurs fois la fonction. Cela signifie que vous pouvez avoir deux appels à la fonction *Choice* dans le même champ ***result*** afin d'obtenir un nombre maximum de 10 choix ; de la même façon, 3 appels à la fonction permettra d'avoir jusqu'à 15 choix différents. Je ne connais pas la limite à ce procédé : je pense qu'on a un maximum de 5 choix pour un appel donné.

Mettre à jour le journal et tester les entrées

[no fix] Journal, "ID_Journal", Index_enum

```
Journal, MG_BCShroomsCombat, 10
```

Ajoute une entrée de journal à votre journal in-game. L'entrée doit avoir été préalablement définie dans l'éditeur de dialogue. L'index indique quelle partie du topic est ajoutée au journal. Attention en utilisant des noms simples pour les topics de journal, adoptez la convention des deux lettres de Bethesda (voir l'exemple ci-dessus) – sinon l'entrée peut très bien apparaître comme une réponse normale dans une conversation, tout comme les autres, si l'expression correspondant au topic apparaît dans le dialogue !

(D'après une info de forum / Melian)

[no fix] SetJournalIndex "ID_Journal" index_enum

```
SetJournalIndex "MG_BCShroomsCombat" 99
```

SetJournalIndex placera l'index à la valeur voulue, que l'entrée de journal pour cette valeur existe ou non (utile pour des simples flags qui ne nécessitent pas leur propre entrée).
Note : Lorsque vous rechargez, l'index sera réinitialisé à la valeur la plus haute de l'entrée existante dans le journal. Par conséquent, on peut aussi utiliser cette méthode pour détecter que le joueur a rechargé le jeu :

```
if ( ( GetJournalIndex "nullos" ) != 100 )
   Messagebox "Vous venez juste de recharger, tricheur !!!"
   setjournalindex "nullos" 100
endif
```

"nullos" est n'importe quel topic du journal qui n'a pas de texte pour l'index 100.

C'est plus facile d'utiliser cette fonction dans les dialogues : envoyez le joueur au "test du courage", et placez l'index du journal à une certaine valeur dans le champ ***Result***. Lorsque le joueur revient et que l'index diffère alors le joueur a échoué au test (info sur cette fonction fournit par JOG).

[no fix] ClearInfoActor

Cette fonction est utilisée dans la section ***Result*** d'un topic de dialogue – utiliser celle-ci pour stopper l'apparition du topic dans le journal du PJ (section "Sujets" (?)). Utile pour éviter d'encombrer la section des topics avec des informations inutiles.

[no fix] GetJournalIndex, "ID_Journal" (renvoie un short)

```
If ( GetJournalIndex, MG_BCShroomsCombat == 10 )
```

Cette fonction renvoie l'index de l'entrée de journal la plus haute (ou la dernière ?) pour le topic de journal que le joueur a reçu. C'est très commode pour garder une trace de l'avancement de la quête ; un script peut ainsi réagir en fonction des parties de la quête déjà effectuées.

Script de démonstration : Voici un petit script qui montre l'usage des fonctions *GetJournalIndex* et *Journal* dans le jeu :

```
Begin attack_slave

short nolore

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

if ( GetJournalIndex "MV_SlaveMule" < 102 ); si le PJ n'a pas encore atteint un certain point
        If ( GetDistance, "Rabinna" < 512 )
                Rabinna->AiWander 0 0 0 0 0 0 0
                StartCombat, "Rabinna"
                Journal "MV_SlaveMule", 102  ; ajoute une entrée de journal
        endif
endif

End
```

Fonctions réservées uniquement au dialogue

Parmi les fonctions disponibles pour les conditions de dialogues, il y en a certaines qui n'ont pas d'équivalent direct dans les fonctions de scripts. En utilisant le dialogue (via un *ForceGreeting*, un dialogue parlé ou des PNJs utilisés dans des quêtes) et le champ *Result* du dialogue, vous pouvez toujours utiliser ces fonctions à des fins de scripts, par exemple en affectant une valeur à une variable globale. Des exemples de ces fonctions sont :

PC Sex (dialogue)

Renvoie 0 si le joueur est de sexe masculin et 1 s'il est de sexe féminin.
Note : il s'agit du seul moyen connu pour déterminer le sexe du joueur – vous pouvez utiliser une variable globale qui contient cette valeur – le moyen le plus discret est d'utiliser une "expression parlée" (*Voice*), par exemple placer un greeting silencieux au sommet de la liste des greetings pour tous les PNJs : celui-ci affecte une valeur à une variable globale "*PC_Sex*", 1 s'il s'agit d'un homme et 2 s'il s'agit d'une femme. Le topic devra être filtré pour n'être actif que lorsque *PC_Sex* est égale à 0.

Talked to PC (dialogue)

Vaut 1 si l'interlocuteur a déjà parlé au PJ et 0 sinon. Vous pouvez utiliser cela pour faire dire quelque chose de spécial à quelqu'un la première fois que vous lui parler – ou le marquer comme "connu" par le joueur à des fins de scripting.
Il semblerait que *TalkedtoPC* se réinitialise à 0 si le joueur reste plus de 72 heures (temps de jeu) en dehors de la cellule du PNJ. Cette limite de temps s'applique également pour les autres fonctions liées aux PNJs telles que "*forceGreeting*": le PNJ est disponible pour un *ForceGreeting* à partir d'une cellule différente seulement pendant 72 heures. Si vous utilisez *PositionCell* sur le PNJ une fois par jour (même sans changer sa position), la limite des 72 heures ne s'applique plus du tout. (Forum info /info sur la limite de temps par Cortex, merci à Srikandi pour me l'avoir signalé). Cette astuce concernant les Acteurs qui rompent leur connexion avec vous impose que la cellule où vous les envoyez soit différente de celle où vous les avez initialement rencontrés (Forum info / Cortex).

Cela implique qu'il ne doit pas s'agir de leur cellule de départ (définie dans l'éditeur) ou qu'il doit s'agir d'une cellule que vous n'avez pas visitée. Dans mon cas, j'avais un intérieur où je les envoyais dans ce but ; l'une ou l'autre des explications peut indiquer pourquoi ça marche.
Après que vous les avez rencontrés, ils sont envoyés là chaque jour même s'ils s'y trouvent déjà après le premier envoi.

Rank Requirement (dialogue)
Cela vérifie que vous êtes suffisamment "qualifié" pour le prochain dans la faction de l'interlocuteur.
Renvoie 0 si votre *Faction Reputation* et vos compétences/talents ne sont pas suffisantes.
Renvoie 1 si vous avez les pré-requis en compétences/talents mais pas la *Faction Reputation*.
Renvoie 2 si vous avez la *Faction Reputation*, mais des compétences/talents insuffisantes.
Renvoie 3 si vous êtes qualifié.

PC Clothing Modifier (dialogue)
Ceci correspond à la valeur totale de tous les habits et pièces d'armure que le joueur porte. La valeur de cet équipement change la disposition des PNJs.

Friend Hit (dialogue)
Utilisé dans les dialogues pour le cas où vous attaquez un membre de votre groupe (comme un compagnon qui vous suit)
Les valeurs de retours sont :
0 = n'a jamais été touché
1 = le PJ l'a frappé 1 fois
2 = le PJ l'a frappé 2 fois
3 = le PJ l'a frappé 3 fois
4 = le PJ l'a frappé 4 fois et le PNJ/la créature ne combat pas le PJ

Changer le paramètre Hello

Get/Mod/SetHello
Changer cela modifie TOUTES les références à cet Acteur. Extrait du fichier d'aide : Hello équivaut à la distance à laquelle l'Acteur s'arrête, fait face au PJ et dit bonjour. Le paramètre (par défaut 30) est multiplié par le paramètre de jeu iGreetDistanceMultiplier (par défaut 7). Ainsi, une valeur de 30 équivaut à une distance de 210 (soit moins de 10 pieds/3 mètres).

Variables de dialogue utiles

Certaines variables de type short sont utilisées par Bethesda pour bloquer certains dialogues. Elles doivent simplement être déclarées dans le script de l'Acteur, aucune valeur n'est requise.
On vérifie leur présence grâce aux filtres Not Local décrit dans le fichier d'aide :

Nolore	Bloque la plupart des dialogues généralistes,
NoIdle	Bloque les Idle voice, utilisé pour les vampires
NoFlee	Bloque les Flee voice (Fuite), utilisé pour les vampires
NoHello	Bloque les Hello voice, utilisé pour les vampires

Cela est vrai si l'interlocuteur n'a pas cette variable locale. Contrairement à la plupart des fonctions "Not", la valeur de la variable est importante pour celle-ci. Le dialogue et la variable elle-même doivent être positionnés à 0. Voici un tableau sur son fonctionnement :

Not Local (dans le dialogue)	La Variable Existe (o/n)	Valeur (dans le script)	Pass? (l'interlocuteur dira cela)
= 0	Non	NA	Oui
= 0	Oui	0	Non

= 0	Oui	5	Oui
= 1	Non	NA	Oui
= -3	Oui	-3	Non

Modifier et tester les Talents, les Attributs et autres « *stats* »

(traduction par Emma Indoril)

Get, Set, et *Modify "stats" – Remarques Générales*

```
GetStat                (renvoie float)
SetStat, var_float
ModStat, var_float

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

Le langage de script du TESC dispose d'un grand nombre d'instructions servant à modifier les « Stats » du PJ et des acteurs (PNJ, monstres…), de l'Intelligence Artificielle etc… Remplacer *Stat* par n'importe quelle caractéristique, talent, réglages d'IA, réputation, résistance… (**Voir la liste complète dans l'Appendice**). Une valeur positive sera ajoutée à la valeur actuelle, de la « Stat » concernée, une valeur négative sera soustraite.
GetStat renvoie un nombre flottant correspondant à la valeur actuelle de « Stat » - pas la valeur maximale, ni celle de « base » (le minimum de chaque race.), mais bien la valeur actuelle, celle qui est utilisé dans le jeu.

Exemple : Vous avez 50 en Force, mais vous souffrez d'une maladie, et votre Force n'est plus que de 30. C'est cette valeur (30) qui sera utilisée dans le jeu (pour calculer par ex. votre poids transportable) et qui sera renvoyée par `GetStat`.

`SetStat` règle la valeur de « Stat » (**Valeur de Base et Actuelle**) à une valeur donnée.
`ModStat` ajoute ou retranche une valeur donnée, **conjointement à la valeur de base et actuelle** de « Stat ».
`ModStat` ne peut faire passer la valeur d'un attribut au-delà de sa limite naturelle (100) alors que `SetStat` en est capable.
Ceci étant généralement valable pour les autres « Stats ».

Attention : Certaines « Stats » (Résistance, par ex.) peuvent avoir des valeurs négatives (Vulnérabilité) tout autant que des valeurs positives (résistance), et ne sont pas forcément limité à un score de 100.

Ces fonctions peuvent être utilisées de nombreuses manières et vous les présenter toutes dans un script n'est pas évident.
Jetez un œil sur le script « Marksman toggle » dans la section « Trucs et Astuces », c'est un bon exemple, de même que le script « Ressusciter un Acteur mort », qui utilise la fonction `ModHealth`.

Ces commandes ont un large champ d'application : elles peuvent être utilisées pour des objets spéciaux, des malédictions ou bénédictions, pour récolter des informations sur les forces et les faiblesses du PJ, pour modifier les réglages de l'IA des PNJ, les rendre plus agressifs si le joueur les a insulté, les rendre moins loquaces la nuit tombée etc…
Une autre application possible est de modifier les scores d'armures et d'armement pour permettre au PNJ de permuter son équipement. Et utiliser une arme de poing plutôt qu'un arc lorsque l'attaquant est trop près…

Dans cette édition, chacune de ces fonction est décrite dans le chapitre qui lui est le plus approprié (magie dans magie, etc…)

Note du traducteur : Dans ce chapitre je n'ai pas trouvé de bonne traduction pour le mot « Stats », qui recouvre, vous l'aurez compris, toutes les valeurs chiffrées qui décrivent le Personnage joueur, j'ai donc laissé le mot Stat, faute de mieux, et je suis ouvert à toutes les propositions…
Emma Indoril, sur le Forum de www.wiwiland.com

Déterminer et changer les stats des Acteurs et du joueur:

Déterminer et changer les attributs :

```
Get/Mod/SetStrength          //Force
Get/Mod/SetIntelligence      //Intelligence
Get/Mod/SetWillpower         //Volonté
Get/Mod/SetAgility           //Agilité
Get/Mod/SetSpeed             //Rapidité
Get/Mod/SetEndurance         //Endurance
Get/Mod/SetPersonality       //Personnalité
Get/Mod/SetLuck              //Chance
```

Déterminer et changer la Santé, la Magie, la Fatigue:

```
Get/Mod/SetHealth            //Santé
Get/Mod/SetMagicka           //Magie
Get/Mod/SetFatigue           //Fatigue
```

Ces fonctions renvoient, modifient ou positionnent les "fonctions" vitales du PJ. Pour les PNJs et le joueur, les fonctions Get indiquent la santé/magie/fatigue courante. *GetHealth* fonctionne aussi sur les armes/armures mais elle ne renvoie que la santé maximum. On ne connaît aucune fonction qui indiquerait la santé courante d'un objet (forum info, Mana User).

Utilisation spéciale avec *ModStat* seulement :

```
ModCurrentHealth, var_float
ModCurrentMagicka, var_float
ModCurrentFatigue, var_float
```

Tandis que *ModHealth* change la même quantité à la fois pour le niveau courant et pour le niveau maximum de la santé d'un Acteur (c'est-à-dire que même un Acteur en bonne santé sera affecté), *ModCurrentHealth* n'affecte que la santé courante et on ne peut pas dépasser la valeur maximum de santé pour cet Acteur (donc en faisant `ModCurrentHealth, 10000` à un Acteur possédant 70 de Santé et une santé courante de 35, sa Santé sera placée à 70 – en faisant `ModHealth, 10000`, cette santé serait passée à 10035).

```
GetHealthGetRatio            (renvoie float)
```

Cette fonction renvoie le ratio de santé d'un Acteur ; la valeur renvoyée est comprise entre 0 et 1 : par exemple, 1 signifie 100% de vie, 0,9 90% et 0 signifie, eh bien...mort je suppose. Elle remplace la fonction *GetHealthRatio*, listée par erreur dans le fichier d'aide original.
Si vous voulez connaître la santé maximum d'un Acteur (souvenez-vous que *GetHealth* renvoient vos points de vie **courants**), vous pouvez utiliser ceci (un simple produit en croix/règle de trois) :

```
Float Vie_Maximum
Float Vie_Courante
Set Vie_Courante to "Actor ID"->GetHealth
Set Vie_Maximum to (Vie_Courante / "ID_Acteur" -> GetHealthGetRatio)
```

Déterminer et changer les compétences/Talents :

Changer les compétences d'armes peut vous permettre de changer l'arme dont se sert par défaut un PNJ. Cela ne semble pas fonctionner aussi bien pour les armures – le PNJ l'armure qui correspond le mieux aux talents tels qu'ils ont été définis dans le TES CS. Si vous connaissez l'identifiant de l'objet et que vous possédez Tribunal ou Bloodmoon, vous pouvez utiliser la fonction *Equip* pour forcer le PNJ à le porter. (Forum Info/Vorwoda_the_Black) Voir la section Trucs et Astuces pour un exemple. Ce qui n'est pas évident, c'est la plage de valeurs acceptables pour une compétence ; ce n'est pas seulement 0-100 comme on pourrait s'y attendre. En fait, il apparaît que les compétences sont stockées comme des floats. Vous pouvez donc les positionner à des nombres beaucoup plus grands mais il reste quelques vérifications : vous ne pouvez pas utiliser des valeurs négatives et les valeurs décimales seront ramenées à des valeurs entières lorsque que vous sauvez/chargez (Thanks FreshFish).

```
Get/Mod/SetBlock              //Parade
Get/Mod/SetArmorer            //Armurerie
Get/Mod/SetMediumArmor        //Armure intermédiaire
Get/Mod/SetHeavyArmor         //Armure lourde
Get/Mod/SetBluntWeapon        //Arme contendante
Get/Mod/SetLongBlade          //Lame longue
Get/Mod/SetAxe                //Hache
Get/Mod/SetSpear              //Lance
Get/Mod/SetAthletics          //Athlétisme
Get/Mod/SetEnchant            //Enchantement
Get/Mod/SetDestruction        //Destruction
Get/Mod/SetAlteration         //Altération
Get/Mod/SetIllusion           //Illusion
Get/Mod/SetConjuration        //Invocation
Get/Mod/SetMysticism          //Mysticisme
Get/Mod/SetRestoration        //Guérison
Get/Mod/SetAlchemy            //Alchimie
Get/Mod/SetUnarmored          //Combat sans armure
Get/Mod/SetSecurity           //Sécurité
Get/Mod/SetSneak              //Discrétion
Get/Mod/SetAcrobatics         //Acrobatie
Get/Mod/SetLightArmor         //Armure légère
Get/Mod/SetShortBlade         //Lames courtes
Get/Mod/SetMarksman           //Précision
Get/Mod/SetMercantile         //Marchandage
Get/Mod/SetSpeechcraft        //Eloquence
Get/Mod/SetHandToHand         //Combat à mains nues
```

Déterminer et changer le niveau

```
Get/Mod/SetLevel (ne fonctionne qu'avec Set et Get)
```

Positionne le niveau d'un Acteur et c'est tout. Les compétences et les stats ne sont pas automatiquement augmentés ; de même le menu de montée de niveau n'apparaît pas lorsque vous l'appelez pour le joueur.

Combat

Engager et terminer un combat

```
StartCombat, "ID_Acteur"
StopCombat
```

```
"ID_Acteur1" -> StartCombat, "ID_Acteur2"
"ID_Acteur1" -> StopCombat
```

StartCombat et *StopCombat* sont utilisés pour placer un Acteur en mode Combat ou pour revenir en mode normal. *StartCombat* forcera l'Acteur appelant à attaquer l'Acteur passé en argument. Bien que *StopCombat* semble "sûr" si on l'utilise à "chaque frame", mieux vaut fournir une condition do once avant de s'en servir, sinon il se pourrait que l'Acteur ne fasse rien du tout. À l'inverse, *StopCombat* est plus dangereux à utiliser car l'Acteur est totalement sans défense : il ne répliquera s'il est attaqué (ce qui vous permet néanmoins d'en faire un véritable pacifiste...).
Une fois en mode combat, les paramètres d'IA de l'Acteur s'appliquent normalement, par exemple si un Acteur possède un taux de fuite élevé, il s'enfuira malgré la commande *StartCombat*. Pour cette raison, vous verrez souvent que le taux d'attaque (commande **SetFight**) est aussi modifié lorsqu'un combat est engagé :

```
If ( GetDeadCount, "Mon ami" > 0 ) ; l'ami du PNJ est mort
       StartCombat, Player ;engage le combat
       SetFight, 100       ;règle l'agressivité
endif
```

Détecter une attaque

```
[no fix] OnPCHitMe    (est une variable locale de type short)
```

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

Une variable locale (et pas une fonction, vous devez la déclarer en tant que variable comme ci-dessus) qui passe à 1 lorsque le joueur frappe l'Acteur appelant. Elle doit être réinitialisée manuellement. Il semble que la présence de cette variable "court-circuite" le comportement normal du PNJ : il n'attaquera plus de lui-même. Si vous ne voulez pas que l'Acteur reste passif, vous devez vous servir de la fonction *StartCombat* manuellement (voir exemple ci-dessous). Une fois que l'Acteur est en mode combat, *OnPCHitMe* ne signale plus les coups du PJ. Sauf si, selon une information des forums, *OnPCHitMe* se réinitialise à 0 si un autre Acteur frappe l'Acteur appelant après que le PJ l'ait fait ; elle repasse à 1 si le joueur le frappe à nouveau.

Note: Selon une info de Nigedo, *OnPCHitMe* enregistre aussi la tentative de crime du PNJ si l'Acteur a un taux d'alarme suffisamment élevé.

Exemple de Script : Un exemple de mon mod Marchands Ambulants dans lequel un guar protège sa charge s'il n'est pas en mode *AIFollow* ; script attaché au guar :

```
Begin _HB_Adros_GuarDefend ;le guar d'Adros se défend

float timer
short en_attaque
short OnPCHitMe

if ( OnPCHitMe == 1 ) ;guar frappé/pris pour cible
       set en _attaque to 1
       set OnPCHitMe to 0
endif
```

```
if (en_attaque == 1)
        StartCombat, player ; attaque le joueur
        set timer to ( timer + GetSecondsPassed)
        if ( timer >= 1 )
                set timer to 0
                if ( GetLOS, HB_adros_darani  == 1) ;si le guar voit son proprio, il l'"appelle"
                        HB_adros_darani -> StartCombat, Player ;attaque du proprio
                        set en_attaque to 0    ;réinitialisation
                endif
        endif
endif

End
```

GetAttacked (renvoie Booléen/ short)

```
        If ( Acteur -> GetAttacked == 1 )
```

Renvoie 0 si l'Acteur n'a jamais été attaqué et 1 s'il a déjà été attaqué.

Exemple de script: Uupse protège 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, "ID_Acteur" (renvoie Booléen/short)

```
        If ( Acteur -> GetTarget, Player == 1 )
```

Renvoie 1 si l'objet cible du combat est ID_Acteur (0 sinon). Cela peut être utilisé pour déterminer si un Acteur combat le joueur (ou un autre Acteur unique).

Exemple de Script: montre comment un Acteur peut être programmé pour "ne pas combattre la loi". Extrait du script "frelene acques script" – c'est une Rédoran qui vous suit mais qui ne vous aidera pas si vous vous battez contre des gardes rédorans (NdT : ici ce serait plutôt un ordonnateur spécifique).

```
[…]

if ( GetCurrentAiPackage == 3 )
;si follow est le package courant, positionner followNow et continuer...

        set followNow to 1
        SetHello 0

        if ( GetTarget "ordinator wander_hp" == 1 )
        ;si elle cherche à attaquer ce PNJ, arrêter le combat
                StopCombat
        endif

        if ( GetPCCell "Vivec, Hlaalu Prison Cells" == 0 )
                ;si follow a terminé, le PNJ est arrivé...
                Journal HR_Stronghold 144
                set followNow to 0
                AiWander 256 0 0 40 20 20 0 0 0 0 0 0
                ForceGreeting
                SetHello 30
        endif
[…]
```

```
HitOnMe, "ID_Arme"            (renvoie Booléen/short)
HitAttemptOnMe, "ID_Arme"     (renvoie Booléen/short)
```

Ces fonctions renvoient vrai (1) pour une frame si l'Acteur appelant est frappé avec succès ou si on a essayé de le frapper avec une arme spécifique. *HitOnMe* n'est utilisé que dans le script du coeur de Lorkhan (*LorkhanHeart script*) (ne le regarder que si vous avez terminé le jeu ou que les spoilers ne vous dérangent pas !). Je suppose que cette fonction est parfaite pour les scripts du genre "vous ave besoin de telle arme pour tuer tel monstre".

Fonctions d'IA Get/Mod/Set relatives au combat : Fight, Flee, Alarm

***Get/Mod/Set*Fight**

Information tirée du fichier d'aide :
Le paramètre *fight* d'un Acteur détermine s'il est enclin à attaquer le PJ. *Mod/Set Fight* semble aussi affecter les nouvelles références de l'Acteur que vous pourriez ajouter après un appel à cette fonction. Lorsque le paramètre *fight* d'un Acteur vaut 100, il attaque le PJ.Les actions du joueur augmente (ou diminue) ce paramètre. Les voici :

Le tableau suivant donne le comportement général qui en résulte

100	Attaquera toujours
95	Attaquera si le PJ est proche (3000 unités)
90	Attaquera si le PJ est proche (2000 unités)
80	Attaquera si le PJ est proche ou s'il ne vous aime pas (1000 unités, 40 Disp)
70	Attaquera si vous êtes proche et qu'il ne vous aime vraiment pas (1000 unités, 35 Disp)
60	Attaquera s'il ne vous aime pas et que vous vous approchez (Disp inférieure à 30)
50	Attaquera s'il vous déteste (Disp à 0)
40	Attaquera s'il ne vous aime pas et que vous vous approchez. (500 Unités, Disp 10)
30	Attaquera s'il vous déteste et que vous avez commis un crime.
20	Attaquera s'il vous ne aime pas et que vous avez commis de multiples crimes.
10	Attaquera s'il vous déteste et que vous avez commis plusieurs crimes à son encontre.
0	N'attaquera QUE S'IL est attaqué d'abord.

Action du PJ	**Valeur par défaut**	**Formule du paramètre de jeu**
Distance du PJ	20 - (Char Distance * 0.005)	iFightDistanceBase - (Char Distance * fFightDistMult)
Attaque l'Acteur	100	iFightAttack
Disposition	(50 - Disposition)*1	(50 - Disposition) * fFightDispMult
Vol à l'étalage	5 * Item Value	fAlarmStealing * Item Value
Pick Pocket	25	iAlarmPickPocket
Décès	25	iAlarmTresspass
Sarcasmes	Déterminée à partir de la formule de Persuasion	
Intimidation	Idem	
Corruption	idem	

***Get/Mod/Set*Flee**

Changer cela modifie TOUTES les références de l'Acteur (voir note).
Placer ce paramètre à une valeur plus élevée rendra la probabilité de fuite de l'Acteur plus importante ; peut ne pas toujours produire ce résultat car l'Acteur se base sur d'autres facteurs (comme la quantité de dommages qu'il peut faire ou les autres stratégies dont il dispose, par exemple la magie ou le combat à distance). Le comportement est fortement influencé par un nombre de paramètres de jeu (*GameSettings*) listés ci-dessous ainsi que par certains mods (ceux de wakim et de maxpublic) qui ont modifiés ces valeurs pour obtenir des comportements de fuite plus réalistes.

***Get/Mod/Set*Alarm**
Changer cela modifie TOUTES les références de l'Acteur (voir note).
Info du fichier d'aide : lorsqu'un crime est commis et qu'un PNJ l'a repéré, celui-ci criera quelque chose au joueur et par la même occasion en informera les autres PNJs du secteur.
Lorsqu'un PNJ entend un cri d'alarme, les paramètres basés sur l'alarme sont ajustés. Plus le paramètre d'alarme est élevé, plus il sera en colère.
Si le PNJ a son paramètre à 100 et qu'il a eu vent du crime, il mettra la tête à prix du PJ pour une certaine somme d'or.
De plus, si le PNJ est de la classe "Garde", il aura un comportement supplémentaire :
Intercepter le PJ, en lui courant après et en l'arrêtant.
Si le niveau de crime du PJ (*CrimeLevel*) est supérieur à 10000, ils attaqueront sur le champ au lieu d'entamer le dialogue.
Les gardes attaqueront aussi à vue toute créature qui s'en prend à quelqu'un (et donc le PJ).

Note : lorsque vous utilisez ces fonctions pour modifier les paramètres d'un Acteur, vous modifiez la référence courante de l'Acteur ET la définition de l'Acteur. Ce qui signifie que si vous rencontrez pour la première fois un nouvel Acteur pour cet ID, il possèdera la nouvelle valeur des paramètres *Alarm/Fight*. De même, si vous quittez une cellule où se trouve un Acteur qui possède toujours les vieilles valeurs, que vous vous reposez pendant 3 jours (pour le purger de la mémoire) puis que vous revenez dans la cellule, il se basera sur la valeur présente dans la définition de l'Acteur, c'est-à-dire le nouveau paramètre (Forum info / Cortex).

Garder la trace des meurtres et des knockouts

OnDeath (renvoie Booléen/short)

```
If ( Acteur -> OnDeath == 1 )
```

Renvoie 1 pour une frame si l'Acteur est tué. *OnDeath* semble se réinitialiser une fois qu'il a été utilisé. Cela signifie aussi qu'un seul script seulement peut détecter la mort de cette façon ; si vous avez un script global et un script local qui utilisent cette fonction, seul le script global détectera *OnDeath*. Une autre solution serait d'utiliser la fonction *GetHealth*. Dans le script suivant, seul le premier message sera affiché (forum info Argent/ThePal):

```
Begin personScript

if ( OnDeath ) ;sous-entendu si OnDeath == 1
       MessageBox "1"
endif

if ( OnDeath )
       MessageBox "2"
endif

End
```

OnMurder (renvoie Booléen/hort)

```
If ( Acteur -> OnMurder == 1 )
```

Renvoie 1 pour une frame si l'Acteur est assassiné. Les conditions pour *OnMurder* ne sont pas vraiment claires – basé sur le contexte utilisé dans le jeu, il semble que *OnMurder* passe à 1 lorsque votre crime est rapporté aux gardes ("votre crime a été rapporté aux autorités"). Donc un meurtre ne se produit que si vous tuez quelqu'un "illégalement" ET que vous êtes vu(e).

Script de démonstration : ce script positionne une variable utilisée dans le topic de dialogue "Hortator des Rédorans" afin de déterminer si le joueur a tué un conseiller :

```
begin RedoranCouncilor

;ne s'embarrase pas des topics généraux...
short noLore

;pour HT_Monopoly
short mageMonopolyVote

;pour le dialogue de l'Hortator...
if ( OnDeath == 1 )
	if ( OnMurder == 1 )
		Set RedoranMurdered to 2
	else
		Set RedoranMurdered to 1
	endif
endif

End
```

OnKnockout (renvoie Booléen/short)

```
If ( Acteur -> OnKnockout == 1 )
```

Renvoie vrai pour une frame lorsque l'Acteur est inconscient (c'est-à-dire K.O. au combat à mains nues)

[no fix] GetDeadCount, "ID_Acteur" (renvoie short)

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

Cette fonction renvoie le nombre de références (individus) du type "ID_Acteur" qui ont été tués. Une fonction utile pour scripter les quêtes afin de vérifier qu'un PNJ est toujours vivant. Notez qu'il existe une fonction équivalente pour les dialogues. On peut imaginer d'autres utilisations, par exemple construire une réputation sur le nombre de certains monstres tués : ceux-ci s'enfuiront après que vous en ayez massacré plus de 100, etc.
Script de démonstration :

GetDeadCount est souvent utilisé pour vérifier qu'un certain PNJ est mort. Il est conseillé d'utiliser "> 0" dans ces cas-là, car on ne sait jamais si un mod n'ajoute pas une nouvelle instance de cet identifiant ; il vaut donc mieux jouer la sûreté.

```
Begin araraUvulasScript

short noLore

if ( CellChanged == 0 )
	return
endif

if ( GetDeadCount "Neloth" > 0 )
	Disable
endif
```

```
End
```

Ressusciter un Acteur mort

Resurrect

```
gateway_haunt->Resurrect
```

Cette fonction ramène un Acteur à la vie. Ses stats et inventaire seront réinitialisés, en fait il "*respawn*". Il y a un bug si cette fonction est utilisée sur le joueur – cela empêchera le joueur (et tous les Acteurs) d'utiliser la magie. Après avoir sauvegardé et rechargé, l'effet disparaît.
Note: le script PuzzleCanal contient une autre solution : il utilise simplement GetHealth <10 pour déterminer que le joueur est "proche" de la mort et le "ressuscite" en lui redonnant toute sa santé – le joueur ne meurt donc jamais.

Script de démonstration : certaines personnes sont justes plus puissantes que d'autres …

```
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
```

Crime

(traduction par Stilgar)

Déterminer et changer le niveau de crime

Get/Mod/SetPCCrimeLevel (PJ seulement)

PCCrimeLevel gouverne la quantité d'or à payer pour être lavé de vos crimes ; influence la disposition des PNJs et la réaction des gardes. Voir aussi la fonction *PayFine*.

Mettre le PJ en prison

[no fix?] GotoJail

envoie le PJ à la prison la plus proche, plus exactement au *PrisonMarker* (un objet *Door*) et applique les pénalités de prison habituelles.
Script de démonstration :

Voici un petit script de B bien sympa, extrait du mod Modern Adventurer. Le pantalon des vacances maudites qui vous envoie en 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 "Le pantalon des vacances contient un enchantement puissant
qui déclenche un torrent de joie. Lorsque les gardes vous ont trouvé dansant le Can-Can tout
en haut de l'embarcadère de l'échassier des marais, ils n'étaient pas très souriants..", "ok"
              elseif ( message==1 )
                     MessageBox "Vous hurlez et criez de joie lorsque le pantalon des
vacances vous fait revivre les plus beaux moments de votre enfance. Les gardes qui vous
ramènent à vos sens contraste durement avec cette expérience. ", "ok"
              endif
       Player -> GoToJail
       endif
Set OnPCEquip to 0
endif

End holiday_script
```

Laver le PJ de ses crimes

[no fix] PayFine

La fonction *PayFine* enlève les objets volés de l'inventaire du joueur mais ne retire pas d'or. Appelée après avoir payé l'amende pour réinitialiser l'IA. Les mains du joueur sont aussi replacées dans leur position 'normale', c'est-à-dire que vous n'êtes ni prêt à lancer une attaque ou un sort.

[no fix] PayFineThief

Idem mais ne retire pas les objets volés. Appelée pour réinitialiser l'IA. Avant l'un des patchs, pouvait donner lieu à des retraits d'objets incorrects.
Pour des exemples, voir plus bas, sous "Variables globales utiles "

Détecter les crimes

[no fix] GetPCCrimeLevel (returns short)

Renvoie le niveau de crime courant du PJ. Peut être utilisé pour détecter n'importe quel crime que le PJ a commis en ayant été vu. Voir les scripts "Bill_MT_writxxxxx" pour des exemples d'utilisation.

Une autre solution rapportée par Nigedo:

OnPCHitMe

Si vous déclarez *OnPCHitMe* dans un script d'un PNJ, n'importe quel crime dont il pourrait avoir conscience provoquera un retour à vrai de cette fonction. Même si le crime ne se fait pas au détriment de ce PNJ, il suffit qu'il ait un Alarm Setting suffisamment élevé pour qu'il se préoccupe de son environnement proche ; le crime comptera comme une attaque de mêlée n'infligeant aucun dommage.
Dans ce cas, bien que *OnPCHitMe* soit moins fiable pour la détection d'attaque sur ce PNJ (j'ai dû utiliser une autre méthode pour le script sur lequel je travaillais), cela ouvre de nombreuses perspectives pour la détection des crimes.

Il est possible d'utiliser cette méthode pour détecter tous les crimes dans un seul script et ce, sans avoir besoin d'un PNJ pour les « rapporter », c'est-à-dire augmenter la prime sur le joueur, ou de vérifier/d'ajuster *PCCrimeLevel*.

Les paramètres d'alarme suivants devraient normalement renvoyer vrai pour les évènements correspondants :-

Evènement	**Alarme minimum**
N'importe quel vol	10
Attaque sur un PNJ	90
Meurtre d'un PNJ	10

[no fix] GetPCInJail (renvoie Booléen/short)
[no fix] GetPCTraveling (renvoie Booléen/short)

Bloodmoon rajoute deux fonctions qui permettent de vérifier que le PJ est en prison/en voyage. Elles renverront 1 si le PJ est en prison/voyage et 0 sinon. Utilisées dans le script de transformation en loup-garou afin d'empêcher la métamorphose du joueur.

Script de démonstration :

```
if ( PCWerewolf != 1 ) ; DON' RUN IF PLAYER ISN'T WEREWOLF
        return
endif

if ( GetPCinJail == 1 )
        return
endif

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

Variables globales utiles

CrimeGoldDiscount (globale de type short)
Contient la quantité d'or correspondant à l'amende réduite proposée dans les guildes des Voleurs.

CrimeGoldTurnIn (globale de type short)
Contient l'amende réduite à payer lorsque vous vous rendez aux gardes.
NdT : apparemment, pour un délit mineur où les gardes ne vous poursuivent pas

PCHasCrimeGold (globale de type short)
Utilisée dans les conditions de dialogues. Vaut 1 si le joueur a assez d'argent pour payer ses crimes.

PCHasGoldDiscount (globale de type short)
Utilisée dans les conditions de dialogue. Vaut 1 si le joueur a assez d'or pour s'offrir les services de la guilde des Voleurs concernant la « tête mise à prix ».

PCHasGoldTurnIn (globale de type short)
Utilisée dans les conditions de dialogue. Vaut 1 si le joueur a assez d'or pour payer l'amende lorsqu'il se rend aux gardes.

Exemple : un **champ *Result* d'un dialogue** pour le service « tête mise à prix » de la guilde des Voleurs :

```
Player->RemoveItem Gold_001 CrimeGoldDiscount
SetPCCrimeLevel 0
PayFineThief
```

Pour comparer, voici le champ *Result* que les gards ont lorsque votre tête est mise à prix et que vous vous rendez (vous le trouverez dans le topic *Greeting 0*):

```
Player->RemoveItem Gold_001 CrimeGoldTurnIn
SetPCCrimeLevel 0
PayFine
```

Ou si vous vous faites prendre et que vous devez payer l'amende compensatoire normale :

```
Player->RemoveItem Gold_001 GetPCCrimeLevel
SetPCCrimeLevel 0
PayFine
```

Magie

Limiter l'usage des téléportations

[no fix] DisableTeleporting
[no fix] EnableTeleporting

Pas besoin de beaucoup d'explications : ces fonctions active ou désactive la possibilité de se téléporter. Efficace pour empêcher les utilisateurs de magie de s'échapper de votre donjon ☺. Dans le jeu original, elle n'est utilisée que lorsque le joueur rencontre Dagoth Ur.

Je ne montrerais pas le script en entier (ce serait un sacré spoiler) mais en voici un extrait :

```
short teleportDisabled

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

on réinitialise plus tard dans le script EndGame.

Note : lorsque Tribunal est installé, cette fonction est effectivement buggée : un des scripts de démarrage de Tribunal annule toutes les autres commandes de téléportation et autorise celle-ci dans une zone spécifique de Longsanglot (merci à Slink et Riiak pour l'info). Voici le coupable :

```
Begin TribunalMain

;vérification pour la téléportation
if ( GetPCCell "Sotha Sil," == 1 )
	DisableTeleporting
else
	EnableTeleporting
endif

;vérfication pour la lévitation
if ( GetPCCell "Sotha Sil," == 1 )
	DisableLevitation
elseif ( GetPCCell "Longsanglot" == 1 )
	DisableLevitation
else
	EnableLevitation ; voici pourquoi la téléportation est tjs autorisée en dehors de
Longsanglot
endif


end
```

Le problème a été corrigé dans une mise à jour. La dernière version du script est la suivante (**NdT** : la version française ne semble pas bénéficier de cette correction) :

```
Begin TribunalMain

short disablestate ;désigne l'état courant pour teleport et levitate
short newstate     ;désigne les changements à faire

;par défaut, on autorise la téléportation et la lévitation
set newstate to 0

;on a seulement besoin de tester les cellules intérieures
if ( GetInterior )
	if ( GetPCCell "Sotha Sil," == 1 )
		;on ne permet pas la téléportation et la lévitation à cet endroit
		set newstate to 1
	elseif ( GetPCCell "Longsanglot" == 1 )
		;on n'autorise pas la lévitation à cet endroit
		set newstate to 2
	endif
endif
```

```
;si l'état devrait changer
if ( disablestate != newstate )
	if ( newstate == 1 )
		DisableTeleporting
		DisableLevitation
	elseif ( newstate == 2 )
		DisableLevitation
	elseif ( newstate == 0 )
		EnableTeleporting
		EnableLevitation
	endif
	set disablestate to newstate ;mise à jour de l'état courant
endif

end
```

Note : *DisableTeleporting* ne désactive pas les objets scriptés qui téléportent. DinkumThinkum a suggéré l'astuce suivante ; celle-ci utilise *GetPCCell* pour vérifier la position courante du joueur. Tant qu'il se trouve dans l'une des cellules du mod, rien ne se passe. Si ce n'est pas le cas, le script ramène le joueur dans la zone adéquate : au point d'entrée du mod par exemple. Cela ne fait pas exactement la même chose que le blocage des téléportations mais la zone devient totalement sans issue jusqu'à ce que les conditions pour en sortir aient été remplies correctement.

```
Begin DT_Test_BalmoraTrap

If ( GetPCCell, "Balmora" == 1 )
;si on se trouve dans Balmora, rien ne se passe
	Return
Endif

;sinon
MessageBox "En route pour Balmora!"
Player -> PositionCell, -21278, -17613, 534, 0, "Balmora (-3, -3)"

End DT_Test_BalmoraTrap
```

Limiter l'usage de la lévitation

[no fix] EnableLevitation
[no fix] DisableLevitation

Ces fonctions servent à autoriser ou bloquer les effets magiques de la lévitation. Lorsque *DisableLevitation* est appelé, tous les effets de lévitation existants sont supprimés. Lorsque le joueur tente d'utiliser un sort de lévitation et que c'est interdit, un message de notification stocké dans la variable *sLevitateDisabled* des ***Game Settings***. À la base, le texte dit "La lévitation ne fonctionne pas ici."

Scripts de démonstration : Ce script est placé sur un objet d'une pièce où la lévitation est interdite.

```
Begin clampstone

short pierre_desactivee ;la pierre a-t-elle été désactivée ? oui/non
short message_affiche   ;le message a-t-il été affiché ? oui/non

if ( pierre_desactivee == 0 ) ;pierre active
	DisableLevitation
	if ( message_affiche == 0 )
		MessageBox "Une pierre étrange au sommet de la pièce empêche la lévitation
ici."
		set message_affiche to 1
	Endif
else
	EnableLevitation
```

```
        if ( message_affiche == 1 )
                set message_affiche to 0
                MessageBox "La pierre a été désactivée. Vous pouvez maintenant léviter dans la
pièce."
        Endif
endif

if ( OnActivate == 1 )
        if ( pierre_desactivee == 0 )
                set pierre_desactivee to 1
        else
                set pierre_desactivee to 0
        endif
endif

End
```

Ce script est placé sur une porte permettant de sortir de la pièce.

```
Begin enable_lev_on_exit ;ce script autorise l'utilisation de la lévitation une fois sorti

if ( OnActivate == 1 )
        MessageBox "Vous quittez le champ d'influence de la pierre......"
        EnableLevitation
        Activate
endif

End
```

Vérifier et gérer les âmes et les gemmes spirituelles

*(**NdT** : le nom des créatures doit rester en anglais)*

HasSoulgem, "ID_créature"

```
        If ( Acteur -> HasSoulGem, "Golden Saint" )
```

Cette fonction vérifie que le joueur possède une gemme contenant l'âme de la créature spécifiée dans son inventaire. Une fonction peu utilisée qui pourrait apporter quelques quêtes intéressantes et de nouvelles utilités pour les gemmes spirituelles.

Exemple : Voici une partie du script *StrongSoulCheck* :

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

RemoveSoulgem, "ID_créature", nombre_enum

```
        If ( Acteur -> RemoveSoulGem, "Golden Saint", 1 )
```

Retire la gemme spirituelle contenant l'âme de la créature spécifiée de l'inventaire du joueur.
Exemple : voici la partie complémentaire du script ci-dessus, extrait de *RemoveStrongSoul* :

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

Note : le joueur ne sera pas très heureux si vous supprimez l'Etoile d'Azura par ce moyen. Voici une solution :

```
short StarCount ;Compteur d'étoile : il pourrait en avoir plus d'une je suppose.
```

```
if ( OnActivate )
	if ( Player->HasSoulGem "Golden Saint" > 0 ) ;si le joueur possède une âme de Sainte ☺
		set StarCount to ( Player->GetItemCount "Misc_Soulgem_Azura" )
		;on compte le nombre d'étoile
		Player->RemoveSoulGem "Golden Saint" 1
		;on enlève la gemme de la sainte dorée
		if ( ( Player->GetItemCount "Misc_Soulgem_Azura" ) < StarCount )
		;si on a supprimé une étoile d'Azura
			Player->AddItem "Misc_Soulgem_Azura" 1
		;on lui redonne
		endif
		Player->AddItem Gold_001, 10000 ;et on le paye
		MessageBox "Merci, revenez quand vous voulez."
	else
		MessageBox "Vous n'avez aucune âme de Sainte dorée."
	endif
endif
```

AddSoulGem "ID_créature", "ID_gemme_spirituelle"

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

AddSoulGem ajoute une gemme du type donné et contenant l'âme spécifiée à l'inventaire du joueur. Pas sûr s'il faut un argument nombre_enum *(**NdT** : apparement pas pris en compte s'il est spécifié)*.

DropSoulgem, "ID_créature"

```
	DropSoulGem "atronach_storm"
```

Je n'ai pas encore testé cette fonction – je suppose que l'objet appelant laisse tomber une gemme contenant l'âme donnée.

[no fix] OnPCSoulGemUse (est une variable de type short)

L'objet est une gemme spirituelle et elle a été utilisée pour recharger ou pour créer un objet.

Les gemmes spirituelles du jeu possèdent les identifiants suivants :
ID des gemmes spirituelles :

```
Misc_SoulGem_Azura	//Etoile d'Azura
Misc_SoulGem_Grand	//Puissante gemme spirituelle
Misc_SoulGem_Greater	//Grande gemme spirituelle
Misc_SoulGem_Common	//Gemme spirituelle ordinaire
Misc_SoulGem_Lesser	//Gemme spirituelle moindre
Misc_SoulGem_Petty	//Gemme spirituelle insignifiante
```

Cette fonction n'est pas utilisée dans le jeu d'origine.

Placer, enlever des sorts et malédiction

NdT : voir l'onglet Spellmaking de l'éditeur pour voir le noms des sorts

AddSpell, "ID_Sort"
RemoveSpell, "ID_Sort"

```
	"ID_Acteur" -> AddSpell "Absorb Speed"
```

La fonction *AddSpell* place le sort spécifié sur l'objet appelant. Cela signifie deux choses : les sorts normaux sont ajoutés à la liste des sorts du joueur. Les malédictions, maladies, etc, affecteront l'objet appelant. Idem pour la fonction *RemoveSpell* : les sorts normaux sont supprimés de la liste ; les effets des maladies et des malédictions sont enlevés.

Note : vous ne pouvez pas enlever les talents raciaux avec cette fonction (forum info).

Lancer des sorts

```
Cast, ID_Sort, "ID_Cible"

       ID_Objet -> Cast, "flame", Player
```

La fonction *Cast* force l'objet appelant à lancer le sort "ID_Sort" sur la cible "ID_Cible", et la cible subira ou bénéficiera des effets normalement.

Note: on a cru que *Cast* ne marchait que sur le PJ. À partir de Tribunal, (pas sûr pour les versions précédentes) vous pouvez utiliser *Cast* pour lancer un sort sur un Acteur à partir d'un Activateur – d'autres combinaisons fonctionnent probablement aussi.

Script de démonstration : la fonction *Cast* peut être utilisée pour les pièges, comme dans le script suivant attaché à un Container. Notez qu'il y a une condition Do Once ici pour que l'effet ne soit pas continuellement lancé sur le joueur.

```
Begin Trap_script

short done

if ( OnActivate == 1 )
       if ( done == 1 ) ; condition do-once
              Activate
              return
       else
              Cast, "flame", Player ;brûle le joueur
              set done to 1
              Activate
       endif
endif

End trap_script
```

L'exemple suivant utilise la fonction *AddSpell* :

```
begin Item_Cast
;lorsqu'on s'équipe de l'objet, le joueur subit une malédiction qui dure 25 s

short OnPCEquip
short CurseAdded
float Timer


if ( CurseAdded ) ; si le joueur est actuellement maudit
set Timer to ( Timer + GetSecondsPassed )  ; mise à jour du timer
	if ( Timer >= 25 ) ; après 25s, on enlève le sort ; 25 points de dommage infligés.
		set Timer to 0
		Player->RemoveSpell "ItemFlame" ;on enlève la malédiction
		set CurseAdded to 0 ;le joueur n'est plus maudit
	endif
endif

if ( OnPCEquip );lorsqu'on s'équipe de l'objet
	if ( MenuMode )
		return
	elseif ( CurseAdded == 0 ) ;la malédiction n'a pas encore été placée
		Player->AddSpell "ItemFlame" ; un sort de type malédiction (curse) !
					;un sort perso qui fait un point de dégâts / s
		set Timer to 0
		set CurseAdded to 1
	elseif ( CurseAdded ) ; n'ajoute le sort qu'une fois
		Player->RemoveSpell "ItemFlame"
		set CurseAdded to 0
endif

end Item_Cast
```

(script de Patrin, édité)

Le sort ajouté est personnalisé : il cause un point de dégâts de feu par seconde. Notez qu'il y a une condition DoOnce implicite dans ce script. **Ne pas avoir de condition do once peut faire crasher le jeu ! Il semble aussi que les créatures tuées par les effets d'un sort de malédiction lancé sur elles affecte toutes les créatures de ce type de la même malédiction. Cela peut être évité en utilisant '*RemoveSpell*' dans une section '*OnDeath*' du script. (Forum Info / Argent)**
Quelques infos sur les divers types de sorts : une capacité (*Ability*) réduisant la vie réduira la santé MAXIMUM, à l'inverse d'une malédiction (*Curse*). Cela ne semble pas être vrai pour les effets d'absorption ou de fortification où cela aurait été plus sensé (forum info / ManaUser).

Gérer et faire des tests sur les sorts

GetSpell, "ID_Sort" (renvoie Booléen/short)

```
	If ( Player -> GetSpell, "heal companion" == 1 )
```

Renvoie vrai si l'objet possède ID_Sort dans son inventaire. Les "Pouvoirs" et les autres sorts liés au signe de naissance ou à la race ne semblent pas être concernés par cette fonction ; seuls les sorts listés dans la partie principale de la fenêtre des sorts. Voir le script de démonstration plus bas.

GetSpellEffects, "ID_Sort" (renvoie Booléen/short)

```
	if ( Player -> GetSpellEffects, "flame" == 1 )
```

Renvoie vrai si ID_Sort affecte l'objet appelant. Les lignes suivantes pourraient être ajoutées dans "Trap_script" présenté à la section "Lancer des sorts" :

```
if ( Player -> GetSpellEffects, "flame" == 1 )
       MessageBox "Des flammes vous ont brûlées"
endif
```

C'est le moyen favori pour ajouter de nouveaux "effets de sorts". Un sort tout bête, qui provoque un effet minimal, est créé, par exemple augmenté la chance d'un point par seconde. La fonction *GetSpellEffects* est utilisée pour détecter que le sort a été lancé sur le joueur et le script se charge de tout le reste. Voir le script de démonstration ci-dessous. Semble aussi fonctionner pour les capacités (*Abilities*) et les maladies (*Diseases*). Les malédictions (*Curses*) et les maladies dues au Fléau (*Blight Diseases*) posent probablement problème pour cela mais je n'ai pas vérifié.

Gérer et faire des tests sur les effets des sorts

GetEffect, ID_Effet (renvoie short)

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

Cette fonction renvoie vrai si l'Acteur appelant est affecté par cet effet. Important : les effets ne sont pas les sorts, mais les éléments sur lesquels sont basés les sorts. Vous trouverez dans l'Appendice une liste de tous les effets des sorts.

RemoveEffects, ID_Effet#_enum

```
       Player -> RemoveEffects, 75
```

Retire tous les sorts lancés sur l'Acteur qui se basent sur cet effet. Pour cette fonction, vous aurez besoin du **numéro** de l'effet correspondant ; tout le contraire de la fonction *GetEffect* qui a besoin de l'identifiant de l'effet (Bravo, Bethesda!). Tous les deux peuvent être trouvés dans l'Appendice.

Important : les effets ne sont pas les sorts, mais les éléments sur lesquels sont basés les sorts. Vous trouverez dans l'Appendice une liste de tous les effets des sorts et le numéro correspondant.

Script de démonstration : voici un script qui vous permet de vérifier si un sort est présent dans l'inventaire, s'il est actif sur le joueur, si l'effet qu'il cause affecte le joueur et de le retirer cet effet par la suite. Vous pouvez le tester en tapant "*StartScript Magictest*" dans la console.

```
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  ;effacez la ligne pour voir ce qui se passe normalement
```

```
        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
```

Tester les maladies

GetBlightDisease **(renvoie Booléen/short)**
GetCommonDisease **(renvoie Booléen/short)**

```
        If ( Acteur -> GetBlightDisease == 1 )
```

Les deux fonctions renvoient 1 si l'Acteur appelant a contracté la maladie correspondante (commune ou due au Fléau), 0 sinon. Elles sont utilisées dans les scripts qui donnent la maladie aux créatures infectées.

Script de démonstration :

```
Begin diseaseBlackHeart

DontSaveObject

if ( CellChanged == 0 )
        return
endif

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

End
```

Explosion

ExplodeSpell "Nom du sort"

```
   ExplodeSpell "proj_trap_spell"
```

La fonction *ExplodeSpell* force une référence à lancer un sort à effet de zone sur elle-même. Si un sort à effet de zone est utilisé, la référence pourra "exploser". Voir le *TrapProjScript*, dans la section Trucs et Astuces concernant le piège à flèche (*Arrow trap*).

Les fonctions d'effets magiques Get/Mod/Set :

La plupart de ces fonctions semble faire référence à certains bonus que vous n'obtenez normalement qu'avec des sorts. Avec elles, vous pouvez apparemment rendre ces bonus permanents ou les modifier. Elles utilisent normalement des valeurs comprises entre -100 et 100 (%) mais elles accepteront n'importe quel nombre, en dehors des flags (0 ou 1). Ainsi une créature pourra supprimer le bonus de résistance au Fléau (*ResistBlight*) – ne serait-ce pas une magnifique surprise pour notre Nérévarine?

***Get/Mod/Set*ResistMagicka**
***Get/Mod/Set*ResistFire**

***Get/Mod/Set*ResistFrost**
***Get/Mod/Set*ResistShock**
***Get/Mod/Set*ResistDisease**
***Get/Mod/Set*ResistBlight**
***Get/Mod/Set*ResistCorprus**
***Get/Mod/Set*ResistPoison**
***Get/Mod/Set*ResistParalysis)**
***Get/Mod/Set*Chameleon**
***Get/Mod/Set*ResistNormalWeapons**
***Get/Mod/Set*WaterBreathing**
Placée à 1, permet la respiration aquatique

***Get/Mod/Set*WaterWalking**
Placée à 1, permet la marche sur l'eau

***Get/Mod/Set*SwimSpeed**
***Get/Mod/Set*SuperJump**
Celles-ci correspondent aux effets de Nage Rapide et de Saut, la plage de valeurs est donc normallement de 0 à 100, mais des valeurs plus hautes ou négatives fonctionnent aussi.

***Get/Mod/Set*Flying**
J'ai trouvé l'info suivante sur l'UESP : cela positionne le mode de vol du joueur. Pour que ce cheat marche, entrez la commande à la console et lancez un sort de lévitation. L'effet perdurera jusqu'à ce que vous désactiviez le vol à la console (merci Dave Humphrey).

***Get/Mod/Set*ArmorBonus**
Correspond aux effets de bouclier

***Get/Mod/Set*CastPenalty**
correspond à l'effet "Son" ? (<0 rend l'incantation plus difficile, >0, plus facile)

***Get/Mod/Set*Silence**
***Get/Mod/Set*Blindness**
***Get/Mod/Set*Paralysis**
Chaque effet de paralysie que vous appliquez à un acteur incrémente le nombre de 1. Chaque effet que vous enlevez décrémente ce nombre de 1. Vous pouvez aussi utiliser les fonctions Set et Mod pour le changer. Quand il est à 0, l'acteur peut se mouvoir normalement. Appelez la console, cliquez sur quelqu'un et tapez **`setparalysis 1`**. C'est bien pratique lorsque vous avez plusieurs acteurs ayant le même identifiant : cela permet d'immobiliser un seul individu, comme le ferait un sort. Si vous utilisez une capacité de paralysie (*paralysis ability*), cela paralyserait tous les acteurs/créatures ayant cet identifiant dans le cas où vous devriez sortir puis réentrer dans une cellule. L'effet de *SetParalysis* perdure jusqu'à ce que vous le replaciez à 0 ou que le joueur reste en dehors de la cellule pendant 3 jours et ce même s'il ne s'agit pas de PNJs 'auto-calculés'. (Forum info / Cortex)

***Get/Mod/Set*Invisibile**
(MW Original : sic! Pas invisible!)
***Get/Mod/Set*Invisible**
(les versions suivantes de MW, apparemment l'orthographe a été corrigée (Forum info / Cortex)

***Get/Mod/Set*AttackBonus**
Correspond à l'effet d'attaque fortifiée

***Get/Mod/Set*DefendBonus**
correspond à l'effet de sanctuaire

Son

Faire parler les Acteurs à l'aide d'un fichier audio

Say, "nom du fichier", "texte"

```
Actor -> say, "vo\Misc\CharGenBoat1.wav", "C'est là qu'on a besoin de vous."
```

Force le sujet à "prononcer" le discours audio, ne marche que pour les objets animés (*animating objects*). Les fichiers sonores des voix se trouvent dans le répertoire "Data files\Sound\Vo\" et sont ordonnés dans des sous-répertoires selon la race et le sexe. Vous pouvez aussi consulter la plupart d'entre eux dans la fenêtre *dialogue/voice*. Texte correspond à ce qui est affiché en sous-titre lorsque le fichier est joué.

SayDone
Renvoie vrai si l'objet appelant ne dit rien.

Script de démonstration : extrait de la génération du personnage

```
begin CharGenBoatNPC
;correspond au garde du bateau qui vous demande de quitter le navire

short state
float timer

if ( menumode == 1)
	return
endif

if ( GetDisabled == 1 )
	return
endif

if ( OnActivate == 1 )
	return
endif

if ( GetDistance, Player < 180 )
	if ( SayDone == 1)
		;première rencontre
		if ( state == 0 )
			if ( timer == 0 )	;utilisation d'un timer pour qu'il ne parle pas
TOUT le temps
				say, "vo\Misc\CharGenBoat1.wav", "C'est là que l'on a besoin de
vous. Allez jusqu'au quai et on vous indiquera où se trouve le Bureau de Recensement."
				set state to 10
			endif
		;pour toutes les autres rencontres
		else
			set timer to timer + GetSecondsPassed
			if ( timer > 6 )
				set timer to 0
				say, "vo\Misc\CharGenBoat2.wav", "Allons-y. En route."
			endif
		endif
	endif
endif

end CharGenBoatNPC
```

Musique

[no fix] StreamMusic, "nom_de_fichier.extension"

Joue le fichier son "nom_de_fichier.extension", normalement un fichier mp3. Le fichier audio doit se trouver par défaut dans le répertoire data files/music/ . La fonction peut aussi lire les fichiers MIDI (JOG).

Note: légèrement buggée : l'appel à *StreamMusic* place automatiquement le volume à 100 et le laisse à cette valeur une fois que la musique s'est terminée. Comme il n'y a aucune fonction pour modifier le volume, l'utilisateur est obligé de réinitialiser manuellement la valeur souhaitée via le menu Options.

Sons

```
[no fix] PlaySound, "ID_son"

[no fix] PlaySoundVP,"ID_son", volume_enum, pitch_enum

PlaySound3D, "ID_son"

PlaySound3DVP, "ID_son", volume_enum, pitch_enum

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

La fonction *PlaySound* lit le son sans aucune modification. L'objet auquel le script est attaché n'a pas d'importance, le son sera toujours entendu à plein volume, directement dans les oreilles du joueur comme qui dirait.

La fonction *PlaySound3D* simule l'émission d'un son depuis une source, l'objet auquel le script contenant la fonction est attaché.

Les variantes "VP" de ces deux fonctions vous permettent de modifier le volume et la balance pour le son joué. Bethesda ne s'en sert pas beaucoup, il semble que cela soit toujours utilisé avec la valeur 1.0 pour les deux arguments ; il s'agit vraisemblablement de la valeur standard. Mes propres essais ont montré que le son n'était pas joué si je plaçais le volume dans une variable, mais ce n'est pas une certitude.
Le "ID_son" est un identifiant de la liste de la fenêtre *Sound*, accessible via ***Gameplay – sounds menu***. Vous pouvez y ajouter des sons (placez les fichiers .wav quelque part dans Data files/sounds). http://www.findsounds.com/ est une bonne ressource sur le web. Les sons doivent être d'un certain format (voir plus bas) ; vous aurez donc peut-être à changer ce format grâce à un programme approprié si vous n'entendez pas le son dans le jeu.

Contrôler le son

```
StopSound, "ID_son"

        ID_objet -> StopSound "Lava Layer"
```

stoppe le son "ID_son" s'il est actuellement joué par l'objet appelant.

```
GetSoundPlaying, "ID_son"     (renvoie Booléen/short)

        if ( GetSoundPlaying "lava layer" == 0 )
```

Renvoie 1 lorsque le son spécifié est actuellement joué. Les identifiants des sons se trouvent dans le menu ***Gameplay /sounds et /sound gen***, où vous pouvez également placer les vôtres. Cette fonction est utilisée pour contrôler les sons mais aussi pour obtenir des informations car certains sons sont liés à des évènements du jeu, par exemple les coups critiques (son "Critical Damage") ou le désarmement des pièges (son "Disarm trap").

Script de démonstration : ce script s'assure que la lave émet toujours son grondement :

```
begin lava

if ( menumode == 1 )
	return
endif

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

end lava
```

Formats des fichiers son

Note: certains sons ne sont pas correctement joués dans le jeu (bien qu'ils le soient tous dans l'éditeur). Pour en être certain, utilisez les formats utilisés par Bethesda (merci à random name) :

Répertoire "Cr" et "Fx"
Windows PCM (.wav)
22050 kHz, 16-bit, Mono

De plus basses qualités fonctionnent aussi, par exemple 8,000 kHz; 8 Bit; Mono. Utilisé notamment dans mon mod "The Regulars" pour la musique des tavernes.

Répertoire "Vo"
MPEG Layer-3, 64 Kbps
44100 kHz, 16-bit, Mono

Garder la trace du temps

Il y a de nombreuses possibilités pour suivre l'écoulement du temps dans les scripts, dont certaines ne sont pas ou peu documentées dans le fichier d'aide.

Timer

[no fix] GetSecondsPassed (renvoie un float)

Un timer peut être scripté avec la fonction *GetSecondsPassed*. Elle renvoie le nombre de secondes écoulées *depuis la dernière frame*. Pour utiliser un timer, servez-vous de quelque chose comme ça :

```
Begin TimerScript

Float timer

Set timer to (timer + GetSecondsPassed)

If (timer > 10)
       MessageBox "Affiché toutes les 10 secondes"
       Set timer to 0
Endif

End TimerScript
```

Variables globales liées au temps de Morrowind

GameHour (variable globale de type float)
Day (variable globale de type short)
Month (variable globale de type short)
Year (variable globale de type short)

Ces variables globales sont positionnées par le jeu et contiennent la date et l'heure courante.

Le calendrier de MW est un petit peu buggé (merci à samois pour l'info): MW commence au Jour 16, Mois 7, Année 427. (16 Vifazur)
Les mois dont les suivants, avec le nombre de jours par mois.

(Primétoile	/	Morning Star	???)
Clairciel	/	Suns Dawn	31
Semailles	/	First Seed	28
Ondepluie	/	Rain's Hand	31
Plantaisons	/	Second Seed	30
Mi-l'an	/	Mid Year	31
Hautzénith	/	Sun's Height	30
Vifazur	/	Last Seed	31
Âtrefeu	/	Heart Fire	31
Soufflegivre	/	Frost Fall	30
Sombreciel	/	Suns Dusk	31
Soirétoile	/	Evening Star	30

Alors il n'y a que 334 jours dans une année en Morrowind !?! en fait, il semble que Bethesda se soit emmêlé les pinceaux et ait perdu un mois, Primétoile / Janvier… il semble qu'après Soirétoile, on passe directement à Clairciel. Si vous placez manuellement la variable Month à 0, Primétoile s'affichera correctement dans le menu de repos. Il y a donc une possibilité de scripter cela.

Script de démonstraion : Vérifier l'heure du jour avec la fonction GameHour:

```
Begin AfternoonTea ;l'heure du thé

If ( GameHour >= 17 )
        If ( GameHour <= 19 )
        ;entre 17 et 19 heures
                "Cup of Tea" -> Enable ;'autorise' la tasse de thé
        endif
elseif ( GameHour < 17 )
        if ( GameHour >19 )
;sans doute une erreur ici; l'heure ne peut pas prendre les valeurs 0-16 et 20-23 en même
temps
                "Cup of Tea" -> Disable
        endif
endif


End AfternoonTea
```

Garder la trace des jours écoulés

Day (variable globale de type short)

Utilisez la variable globale "Day". Elle contient le "jour du mois" courant – pour le "17, Vifazur", c'est 17. On peut garder la trace des jours passés comme ceci :

```
Short localdaysPassed ;nbre de jours passés
Short currentDay      ;jour courant

if ( currentDay != Day ) ;à chaque fois que Day change (vraisemblablement incrémentée)…
        set currentDay to Day ;on place la var à la valeur du jour courant
        set localdaysPassed to localdaysPassed + 1 ;ajoute 1 au compteur
endif
```

on utilisera cette méthode pour les quêtes limitées en temps dans un script global afin d'être sûr que l'écoulement du temps est correctement mesuré. On s'en servira également pour déclencher des évènements, par exemple le joueur est en possession d'un objet depuis un certain temps, etc.

DaysPassed (variable globale de type short)

Contient le nombre de jours depuis le commencement du jeu. Pour fonctionner, *DaysPassed* doit être déclaré en tant que variable globale de type short. La déclaration est présente dans Tribunal.esm, mais pas dans Bloodmoon.esm. Ainsi, pour les mods qui veulent s'en servir et qui ne dépendent pas de Tribunal, *DaysPassed* DOIT être explicitement déclaré. Cela peut être l'une des raisons pour lesquelles certains ont indiqué qu'elle était buggée dans Bloodmoon, et pour d'autres non – cela dépend du fait que Tribunal.esm soit coché ou non (Forum info / Erstam).

Phases lunaires

Indispensable pour les mods potentiels de loup-garous.

```
[no fix] GetMasserPhase              (renvoie short)
[no fix] GetSecundaPhase      (renvoie short)

        If (GetMasserPhase == 4)
                [autoriser la présence de loup-garous]
        endif
```

Note: le fichier d'aide liste *GetSecundusPhase*, mais la syntaxe ci-dessus *GetSecundaPhase* est celle qu'il faut utiliser. Notez aussi que *GetMasserPhase* et *GetSecundaPhase* renvoient la valeur de la phase lunaire pour la dernière cellule extérieure que vous avez visitée. (Forum Info / Elim).

Je n’ai fait qu’un test rapide, mais elles semblent fonctionner.
Les deux fonctions renvoient un short avec ces valeurs :
0 = MOON_PHASE_NEW (par défaut) : nouvelle lune
1 = MOON_PHASE_WAXING_CRESCENT or MOON_PHASE_WANING_CRESCENT: premier ou dernier croissant
2 = MOON_PHASE_WAXING_HALF or MOON_PHASE_WANING_HALF: demi-lune montante ou descendante
3 = MOON_PHASE_WAXING_GIBBOUS or MOON_PHASE_WANING_GIBBOUS: lune gibbeuse montante ou descendante
4 = MOON_PHASE_FULL : pleine lune

Climat

Changer le temps

[no fix] ChangeWeather, "ID_region", short_Type_Enum

```
ChangeWeather, "Région de la Faille de l'Ouest", 4
```

Cette fonction change le temps dans la région indiquée suivant la valeur de TypeEnum ; le temps changera à nouveau suivant les paramètres de la région après une durée défini par le jeu (je suppose que le jeu fait référence à la valeur définie dans le fichier Morrowind.ini, dans la section *Weather*. Pour le mien, l'entrée indique :

```
Hours Between Weather Changes=20 ;soit le nombre d'heures entre les
changements de temps
```

Les valeurs climatiques pour TypeEnum sont :

0	Clair
1	Nuageux
2	Brumeux
3	Couvert
4	Pluvieux
5	Orageux
6	Cendre
7	Fléau

Changer les paramètres climatiques pour une région

[no fix] ModRegion, "ID_region", clair_enum, nuageux_enum, brumeux_enum, couvert_enum, pluvieux_enum, orageux_enum, cendre_enum, fléau_enum

```
ModRegion, "Région de la Faille de l'Ouest", 10, 20, 10, 5, 5, 40, 10, 0
```

Change la probabilité du climat pour l'identifiant de la région. Utilisé pour supprimer ou ajouter de façon permanente un climat à une zone. La somme des valeurs doit être égale à 100 ou vous obtiendrez des résultats bizarres.

Déterminer le temps qu'il fait

[no fix] GetCurrentWeather (renvoie short)

```
If ( GetCurrentWeather == 1)
        ;[faire quelque chose si c'est nuageux]
endif
```

renvoie une des valeurs TypeEnum listées ci-dessus.
Script de démonstration : Bethesda utilise ceci pour faire bouger les bannières en fonction du type de temps :

```
begin OutsideBanner

;ce script est pour les objets bannières situés à l'extérieur afin qu'ils s'animent lorsqu'il
;y a du vent.
;Idle correspond à calme, Idle2 à une petite brise et Idle3 à un gros coup de vent

short ran

if ( MenuMode == 0 )
        set ran to random 100
        if ( ran < 30 )        ;30% de chances que la bannière bouge différemment
                ;ceci vérifiera le climat dans le futur
                if ( GetCurrentWeather  >= 5 )              ;orage, cendre ou fléau
                        LoopGroup, Idle3, 5
                endif
```

```
                ;la dernière anim appelée par ce script sera celle qui sera jouée
                if ( ran <= 10 )
                        PlayGroup, Idle

                elseif ( GetCurrentWeather < 5 )
                        PlayGroup, Idle2

                endif

        endif

endif
```

Déterminer la vitesse du vent

Pas de documentation:

[no fix] GetWindSpeed (renvoie float)

Je n'ai testé cette fonction que brièvement ; valeurs de retour : 0 pour les intérieurs, des floats pour les extérieurs (varie rapidement, pour un temps nuageux, les valeurs semblent converger vers 2). (Merci à XPCagey pour avoir trouvé cela)

Contrôles du joueur

Repos du joueur

[no fix] ShowRestMenu

Affiche le menu de repos et permet au joueur de dormir. Utilisé par exemple pour les lits situés dans les cellules où il est interdit de dormir.

Script de démonstration : **voici le script standard pour les lits :**

```
begin Bed_Standard

;utilisé pour les lits standards que le joueur peut activer et dans lesquelles il peut dormir

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

end
```

[no fix] GetPCSleep (renvoie Booléen/short)

```
        if ( GetPCSleep == 0 )
```

Renvoie vrai (1) si le joueur est en train de dormir. **Note:** Le compteur que vous voyez pendant que vous dormez est un menu. Soyez prudent si vous utilisez cette fonction dans le même script que la fonction *MenuMode* !
L'**exemple** semble venir d'un objet assez inutile mais montre l'utilisation...

```
Begin pillowScript ;le script des oreillers

short comfy

if ( PCVampire == 1 )
        return
endif

if ( comfy == -1 )
        if ( player->GetItemCount "misc_uni_pillow_unique" > 0 )
                if ( GetPCSleep == 0 )
                        set comfy to 0
                        return
                endif
        endif
endif


if ( comfy == 0 )
        if ( player->GetItemCount "misc_uni_pillow_unique" > 0 )
                if ( GetPCSleep == 1 )
                        MessageBox "Votre sommeil est très réparateur grâce à votre oreiller
super-confortable"
                        set comfy to -1
                        return
                endif
        endif
endif


End pillowScript
```

[no fix] WakeUpPC

Force le réveil du PJ avant l'heure sélectionnée. Crée parfois un monstre si le joueur dort à l'extérieur. Cela arrive toujours s'il essaie de ne dormir qu'une heure ; pour les durées plus longues, cela peut ne pas arriver (Merci à Manauser pour cette info). *WakeUpPC* interrompt le repos seulement lorsque vous *dormez*. Cela n'affecte pas les flâneries dans les endroits où dormir est interdit. (Forum info / Kir).

Script de démonstration : voici un petit extrait édité du très long script "sleepers" de Bethesda. C'est lui qui est responsable des rêves de Dagoth Ur qui affecte le joueur pendant la quête principale. Il montre comment utiliser *GetPCSleep* et *WakeUpPC* :

```
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
        ;pour être sûr que vous avez quitté la ville et êtes revenu afin qu'une nouvelle
attaque se produise
        return
endif

AddTopic "Disturbing Dreams"
;ajoute le topic, pas grave si c'est fait plusieurs fois

;LE PREMIER REVE...

if ( GetJournalIndex A1_2_AntabolisInformant >= 10 )
        if ( GetJournalIndex A1_Dreams < 1 )
                WakeUpPC
                MessageBox "Vous avez fait un rêve bizarre. Bla bla bla", "Ok"
                Journal A1_Dreams 1
                return
        endif
endif
```

Activer et désactiver l'interface et les contrôles du joueur

Fonctions de désactivation des contrôles du joueur

Toutes ces fonctions désactivent une partie de l'interface utilisateur, réduisant ainsi les actions du joueur.

[no fix] DisablePlayerControls

Le joueur peut regarder autour de lui avec la souris ou utiliser le menu Options, c'est tout ; les autres menus disparaissent.

[no fix] DisablePlayerFighting
[no fix] DisablePlayerMagic

Ces deux fonctions ne semblent pas très sûres à utiliser d'après les informations recueillies sur les forums : si le joueur tient une arme ou est prêt à lancer un sort, il peut continuer à l'utiliser ainsi que les raccourcis armes et sorts. Je ne vois actuellement pas de solution pour palier à ce problème.

[no fix] DisablePlayerJumping
[no fix] DisablePlayerLooking
[no fix] DisablePlayerViewSwitch
[no fix] DisableVanityMode

Fonctions d'activation des contrôles du joueur

Une fois qu'une fonction désactivante a été utilisée, la fonction d'activation correspondante peut être utilisée pour redonner le contrôle.

```
[no fix] EnableLevelUpMenu
[no fix] EnablePlayerControls        (Autorise les contrôles et les menus.)
[no fix] EnablePlayerJumping
[no fix] EnablePlayerFighting
[no fix] EnablePlayerLooking
[no fix] EnablePlayerMagic
[no fix] EnablePlayerViewSwitch
[no fix] EnableRest
[no fix] EnableVanityMode
```

Vérifier le statut des contrôles du joueur

Toutes ces fonctions renvoient 1 si la fonction correspondante a été appelée et est active, 0 si le joueur a le contrôle.

```
[no fix] GetPlayerControlsDisabled
[no fix] GetPlayerFightingDisabled
[no fix] GetPlayerJumpingDisabled
[no fix] GetPlayerMagicDisabled
[no fix] GetPlayerLookingDisabled
GetPlayerViewSwitch
```

(**Buggée**, cette fonction ne marche pas ; utilisez à la place :)

```
[no fix] GetVanityModeDisabled
```

Forcer la vue à la 1ère ou à la 3e personne

```
[no fix] PCGet3rdPerson       (renvoie Booléen/short)
```

renvoie 1 si on est en vue à la 3e personne

```
[no fix] PCForce3rdPerson
```

place le changement à la 3e personne dans la file d'attente (on peut en effet avoir à attendre la fin de l'animation)

```
[no fix] PCForce1stPerson
```

pareil qu'au-dessus mais pour la 1ère personne.

(Voir aussi la commande "ToggleVanityMode" (TVM) à la console).

Fonctions pour les menus de création du personnage

Ces fonctions, non documentées, sont utilisées pendant la phase de génération du personnage. Elles activent tous les menus qui sont utilisés pendant le processus et les fonctionnalités de base telles que l'inventaire, le menu magie, la fenêtre des stats et la carte :

Menus apparaissant à la génération du personnage :

```
[no fix] EnableBirthMenu
[no fix] EnableClassMenu
[no fix] EnableRaceMenu
[no fix] EnableNameMenu
```

Il n'y a pas de commande de désactivation pour celles-ci. Elles disparaissent en sélectionnant ok dans le menu.

Autoriser les menus in-game :

```
[no fix] EnableMagicMenu
[no fix] EnableMapMenu
```

```
[no fix] EnableInventoryMenu
[no fix] EnableStatsMenu
```

Pas de commandes de désactivation non plus. Elles auraient pu être utiles.

Les noms de ces fonctions devraient être assez explicites. On peut s'en servir pour modifier son apparence ou d'autres choses pendant l'exécution du jeu (bien que des problèmes peuvent apparaître en faisant cela). Elles peuvent être (et ont été) utilisées pour créer plusieurs méthodes de création du personnage. Soyez prudent, parfois votre niveau est ramené à 1.

Script de démonstration : voici l'un des nombreux scripts CharGen qui guident le joueur à travers la création du personnage. Celui-ci sécurise le processus pour le cas où le joueur se précipite vers la porte de sortie sans déclencher les petits tutoriaux.

```
Begin CharGenDoorExit

;il s'agit de la porte qui permet de quitter la première partie du bâtiment du recensement
;vérification de sûreté pour que tous les menus soient actifs

short done

if (done == 1)
        return
endif

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

End
```

Déterminer si le joueur a ouvert les menus

```
[no fix] MenuMode

        If ( MenuMode == 1 )
```

MenuMode renvoie 1 si le joueur a activé les menus (l'écran d'inventaire). Il est commun de trouver les lignes suivantes au début d'un script, afin d'éviter que celui-ci s'exécute lorsque le joueur ouvre son inventaire. Les dialogues, le contrôle du repos, la console et probablement presque tous les menus l'activeront aussi.

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

Utiliser MenuTest pour ouvrir et fermer des menus

Pas de documentation :

```
[no fix] MenuTest, short_enum
```

MenuTest ne renvoie rien, cependant, lorsqu'elle est appelée, elle ferme certains types d'inventaire, dont celui du PJ, des PNJs et des containers. Ne fonctionne pas pour les dialogues, les menus enchantement, alchimie, armurerie ou achats de sorts. (Forum Info / JOG, Jilin).

menutest ou menutest 0 pour fermer le menu
menutest 3 ouvre le menu stats ou le met au premier plan
menutest 4 ouvre le menu inventaire ou le met au premier plan
menutest 5 ouvre le menu sorts ou le met au premier au premier plan
menutest 6 ouvre le menu map ou le met au premier plan

pour menutest 3,4,5,6, c'est comme si vous cliquiez sur le bouton en haut à droite du menu

Script d'exemple :

```
if ( OnPCEquip == 1 )
   set OnPCEquip to 0
   coc Balmora
   MenuTest ; ferme le menu une fois la téléportation effectuée
endif
```

Variables et fonctions diverses

Sortir de l'exécution d'un script

[no fix] Return

Return indique au moteur du jeu de terminer l'exécution du script pour cette frame. Tout le code situé sous cette ligne sera ignoré. À la frame suivante, le script est de nouveau exécuté à partir du début.

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

Attention : rien de ce qui est situé sous la fonction *return* ne sera exécuté, même si certains branchements *if* devraient être évalués à vrai ! à utiliser avec précaution.

Contrôler les scripts globaux

[no fix?] ScriptRunning, "Nom_du_Script" (renvoie Booléen/short)

StartScript, "Nom_du_Script"

StopScript, "Nom_du_Script"

Ces fonctions permettent de contrôler les scripts globaux. Les scripts globaux se démarrent avec la fonction *StartScript* (ou bien à partir d'un autre script, local ou global, ou bien à partir du champ ***Result*** d'un dialogue) et un script s'exécutant se termine avec la fonction *StopScript*.
Note : si vous utilisez *StopScript* dans le script global que vous voulez terminer, celui-ci ne s'arrête pas immédiatement : le script continue son exécution jusqu'au *End* final et ensuite se termine. Vous aurez donc besoin de la commande *Return* si vous ne voulez pas que le reste du script soit exécuté.
Les *Start Scripts* (scripts de démarrage) de Tribunal d'un plug-in commencent leur exécution chaque fois que le jeu est chargé. Si vous mettez fin à l'un de ces scripts avec *StopScript*, celui-ci redémarrera au prochain chargement – voir la section Trucs et Astuces : "Détecter lorsqu'un joueur effectue un chargement d'une sauvegarde ". (Forum info / DinkumThinkum).
À ma connaissance, ces fonctions ne marchent pas avec des scripts locaux attachés à des objets ; vous pouvez cependant démarrer et stopper des "*targeted scripts*" en utilisant la notation pointée d'un objet : ID_objet -> StartScript (pour plus d'info, voir la section Trucs et Astuces sur les *targeted scripts*).

La fonction *ScriptRunning* renvoie 1 si un script est en train de s'exécuter, 0 sinon :

```
if ( ScriptRunning, CharGen == 0 )   ;si le script ne tourne pas
        StartScript CharGen          ;le lancer
Endif
```

StopScript peut aussi être utilisé pour construire des conditions do-once dans les scripts globaux d'une manière très nette, le script se terminant de lui-même :

```
Begin do-once_script

[…]; faire des trucs ici
StopScript do-once_script
End
```

Appliquer un effet de fading à l'écran

```
[no fix] FadeIn temps_float_enum
[no fix] FadeOut temps_float_enum

[no fix] FadeTo alpha_enum temps_float_enum

        FadeTo 50 2.0 ;(effet de fading de 50% en 2 secondes)
```

FadeIn et *Fadeout* obscurcissent l'écran (pas un objet) en fonction du temps spécifié (en secondes). Le temps est une valeur positive inférieure ou égale à 10. La fonction *FadeTo* n'applique qu'un certain pourcentage de l'effet : 0 correspond à totalement transparent et 100 correspond à l'obscurité la plus complète.

Ajouter un lieu à la carte

```
[no fix] ShowMap "ID cellule"

        ShowMap "Gnisis"
```

Cette fonction mettra en évidence les cellules indiquées. L'identifiant de cellule peut être complet ou partiel, c'est-à-dire que toutes les cellules qui commencent par la chaîne donnée seront indiquées sur la carte (par exemple *ShowMap "Vivec"* indiquera tous les quartiers).

Script de démonstration : en lisant ce livre, tous ces lieux seront indiqués sur votre carte :

```
Begin bookPilgrimsPath

if ( GetJournalIndex TT_PilgrimsPath >= 100 )
        Return
endif

if ( OnActivate == 1 )
        Journal TT_PilgrimsPath 100
        ShowMap "Gnisis"
        ShowMap "Vivec"
        ShowMap "Porte des Ames"
        ShowMap "Entrée de la caverne de Koal"
        ShowMap "Champs de Kummu"
        Activate
endif

End
```

Voir aussi la commande "*FillMap*" à la console.

Assigner des valeurs aléatoires à des variables

```
[no fix] Random, valeur_enum

        Set ma_variable to Random, 50
```

Introduire un peu d'imprévu dans les effets d'un script est une bonne idée et peut être réalisé en utilisant la fonction *Random*. *Random* renvoie une valeur comprise entre 0 et la valeur définie–1. Pour l'exemple ci-dessus, ma_variable prendra une valeur comprise entre 0 et 49.
Notez que la variable globale de type short *Random100* est positionnée à chaque frame par le script *Main*, à des valeurs comprises entre 0 et 100 ; vous pouvez donc vous servir de cela aussi.

Note: pour tout appel à *Random* avec des valeurs supérieures à 100, la plage des valeurs retournées devient vraiment pauvre... jusqu'à *Random, 255* où vous n'obtenez que 0 ou 1...et tous les multiples de 256 provoque un crash sur le bureau (CTD) (Morrowind et Tribunal). Dans Bloodmoon, il semble qu'ils aient corrigé l'étendue de la plage des valeurs de retour... on obtient des nombres qui sont mieux distribués, même pour une plage supérieure à 100.

Mais les CTD à 256, 512, etc arrivent toujours (Info by Neko). De plus, on a découvert que parfois, le résultat du *Random* est plus élevé que le nombre donné. Donner le résultat d'un *Random* à une variable avec l'une des valeurs suivantes produit quelque chose d'étrange, la valeur la plus élevée sera en fait proche de 1100: 65, 66, 68, 70, 71, 76, 77, 79, 82, 83, 84

Jouer une vidéo

```
[no fix] PlayBink "nom de fichier" flag_enum
```

Met le jeu en pause et joue la vidéo. Placez le flag à vrai pour donner la possibilité de passer le film. La vidéo doit être au format Bink et placée dans le répertoire Datafiles/Videos. MW charge les vidéos par défaut à partir du CD ; je ne suis donc pas sûr que cela fonctionne. Il est possible (pas testé) qu'en plaçant "TryArchiveFirst=-1" dans le fichier Morrowind.ini puisse avoir une influence là-dessus (-1 utiliser les données du CD, 0 utiliser les plus récentes, 1 utiliser les Archives seulement). Sinon vous devrez avoir recours à un crack no-CD pour pouvoir jouer vos propres vidéos.

Fonctions pour les listes de niveaux (Leveled Item et Creature)

```
[no fix] AddToLevCreature     "nomliste levcrea" "ID_créature" niveau_enum
[no fix] AddToLevItem "nomliste levitem" "ID_objet" niveau_enum
[no fix] RemoveFromLevCreature "nomliste levcrea" "ID_créature" niveau_enum
[no fix] RemoveFromLevItem "nomliste levitem" "ID_objet" niveau_enum
```

Ces fonctions permettent de manipuler les listes de *Leveled Item* et *Leveled Creature* à l'exécution. Les listes de niveaux consistent en des couples objet/niveau où le niveau est le niveau que le PJ doit avoir atteint pour rencontrer l'objet. Les fonctions *AddTo* ajouteront le couple objet/niveau dans la liste de niveau spécifiée tant que cette liste ne contient un couple similaire. Les fonctions *RemoveFrom* supprimeront toutes les occurrences du couple objet/niveau de la liste de niveau. **De plus, si la fonction *RemoveFrom* fournit un couple dont le niveau est -1, tous les couples contenant l'objet spécifié sont supprimés.**

Note: les fonctions *RemoveFrom* ne supprime pas les objets existants dans le monde. Si une instance de *Leveled Creature* a déjà effectué son calcul pour créer une certaine créature, retirer cette créature de la liste des *Leveled Creature* ne vous débarrassera pas de la créature existante. Cependant cela empêchera cette instance de *Leveled Creature* de créer une nouvelle fois cette créature.

Script de démonstration :

Lorsque ce script est placé sur un objet, son activation déclenche la suppression des rats dans la liste des *Leveled Creature* et enlève la viande de rat des *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 "plus aucun rat."
        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 "les rats reviennent."
        Endif
endif

end
```

Racine carrée

[no fix] GetSquareRoot, nombre (float)

```
        set var_1 to GetSquareRoot var_2
```

La fonction *GetSquareRoot* renvoie la racine carrée du nombre donné. Cela peut être utile dans les calculs de distance ou de vecteurs (vous vous souvenez de Pythagore ?).

Fonctions de niveau de l'eau

[no fix] GetWaterLevel (float)
[no fix] SetWaterLevel nouveau_niveau_d_eau_float
[no fix] ModWaterLevel changement_du_niveau_de_l_eau_float

Une bonne opportunité pour des pièges cruels … ces fonctions sont utilisées pour déterminer et pour modifier le niveau d'eau de la cellule intérieure courante. Lorsqu'un Acteur se trouve subitement sous l'eau, il attendra jusqu'à la moitié de sa barre de souffle pour entamer sa remontée vers la surface selon une ligne droite. Les corps flottants suivent les mouvements de l'eau sans gérer les collisions.

Scripts de démonstration :

Ce script va avec une manivelle qui permet d'augmenter ou de baisser le niveau d'eau de la pièce.

```
Begin crank

short changelevel     ;changement de niveau à effectuer : oui ou non
float direction       ;égale 1 si l'eau monte ; -1 sinon
float waterlift       ;valeur de la montée ou de la descente d'eau
short crankturn       ;pour effectuer la rotation de la manivelle
short currcrank       ;position courante de la manivelle
float newwaterlevel   ;nouvelle valeur du niveau de l'eau

if ( MenuMode )
        return
endif

if ( OnActivate == 1 )
        if ( changelevel == 0 ) ;si on n'est pas en train d'effectuer un chgmt
                if ( direction == 1 )
                        set direction to -1
                else
                        set direction to 1
                endif
                set changelevel to 1
        endif
endif
if ( changelevel == 0 ) ;pas de changement à faire -> on sort
        return
endif
```

```
;rotation de la manivelle
set crankturn to 360 * GetSecondsPassed
set crankturn to crankturn * direction
set currcrank to GetAngle X
set crankturn to currcrank + crankturn
SetAngle X crankturn

;calcul de l'augmentation ou de la diminution du niveau de l'eau
set waterlift to 120 * GetSecondsPassed
set waterlift to waterlift * direction
ModWaterLevel waterlift

set newwaterlevel to GetWaterLevel
if ( direction == 1 )                ;si on augmente le niveau d'eau
       if ( newwaterlevel >= 600 )   ;niveau d'eau >=600
              SetWaterLevel 600      ; c'est le maximum défini pour cette pièce
              set changelevel to 0   ;on a fini de changer le niveau
       endif
else                                 ;si on baisse le niveau
       if ( newwaterlevel <= 0 )     ;le niveau est au minimum nul
              SetWaterLevel 0
              set changelevel to 0   ;on a fini de changer le niveau
       endif
endif

end crank
```

Ce script, basé sur le script "Float", est placé sur n'importe quel objet avec un pivot central qui le fait flotter à la surface de l'eau sans se soucier du niveau. L'objet s'arrêtera également de balancer si le PJ se tient dessus.

```
Begin NewFloat

float timer
float swingTime       ;temps d'oscillation
float startAngle      ;angle de départ
float currangle       ;angle courant
short reset           ;réinitialisation : oui ou non

float xvalue
float zvalue
float zoffset         ; ?
float tmpoffset       ; ?
float weightoffset    ; ?

set startAngle to GetStartingAngle, x

if ( MenuMode == 0 )

   if ( timer == 0 )                          ;initialisation du timer
      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 )                       ;si le PJ se tient sur l'objet
      set zoffset to -30
      SetAngle X 0                            ;objet // à l'axe des x
   else
      ;rotate up
      if ( timer < swingTime )                ;ici si timer<1

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

```
      ;rotate down
      elseif ( timer < (swingTime * 3) )       ;ici si 1<timer<3

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

      ;up again
      elseif (timer < (swingTime * 4 ) )    ;ici si 3<timer<4

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

      ;> à 4 secondes (4*swingTimer), on réinitialise
      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
```

Trucs et astuces

Quelques aides : recherche de texte, copier-coller

Une bonne fonctionnalité à utiliser pour un débutant est la fonction de recherche de texte dans *edit* du menu principal du TESCS. Vous pouvez l'utiliser sur les scripts, par exemple pour chercher des scripts représentatifs d'une fonction spécifique que vous voudriez utiliser.

Vous pouvez aussi utiliser n'importe quel éditeur de texte ou l'un des éditeurs alternatifs pour les scripts listés ci-dessous ; il ne restera plus qu'à utiliser le copier-coller vers le TESCS en utilisant ctrl-c / ctrl-v.

Pour faire une copie d'un script que vous voudriez modifier, ne changez pas simplement le nom, cela effacera l'ancien script. Au lieu de ça, copier le script original (ctrl-a, ctrlc), ouvrez un nouveau script et copiez-y le précédent (ctrl-v) ; renommez maintenant le script et modifiez ce que vous voulez.

Autres éditeurs de scripts

1) MentalElf a créé un mode *Elder Scrolls Scripting* pour le vénérable éditeur **EMACS**, incluant la tabulation automatique des blocs if et la coloration syntaxique :
http://www.aloha.net/~frann/rsg/
EMACS est un logiciel libre sous licence GNU (le lien se trouve sur la page de MentalElf).

2) Dave Humphrey de chez UESP a créé **MWEdit,** un autre Construction Set avec un support plus important des scripts (c'est une version bêta. Je ne l'ai testé que brièvement mais cela semblait prometteur et stable): http://mwedit.sourceforge.net/

A propos des scripts, Dave propose les caractéristiques suivantes :
- Coloration syntaxique du code. Utilise par défaut le blanc et le bleu mais vous pouvez définir n'importe quelle couleur pour identifier les différents types de mots. Peut aussi être désactivé (*voir le script de trigonométrie ci-dessous pour une illustration de la coloration syntaxique*).
- Sélectionne la police utilisé dans la fenêtre de script.
- Les nouveaux compilateurs détectent beaucoup plus d'erreurs, aussi bien potentielles que réelles.
- Trois niveaux de messages d'avertissements/d'erreurs (faible, défaut, fort) vous permettant de définir combien de messages seront enregistrés.
- Le compilateur ajoute des espaces là où ils sont requis ou attendus (pour les conditions if pa r exemple).
- Les types d'objets utilisés dans les fonctions sont vérifiés plus rigoureusement. Si la fonction attend l'ID d'un PNJ, vous recevrez un message d'avertissement/d'erreur si vous lui passez un autre type.
- Compile automatiquement le script à la sauvegarde (aucun message d'avertissement ou d'erreur).
- Exporter et importer des scripts à partir ou vers des fichiers textes.
- Consulter une aide détaillée sur toutes les fonctions.
- Tous les messages sont affichés dans un panneau à part situé sous la fenêtre de script. Double-cliquez sur un message pour voir la ligne incriminée.
- Consulter des informations détaillées sur les messages du compilateur et afficher l'aide se rapportant à la fonction en question.

- Le compilateur n'autorise pas l'utilisation des mots réservés (end, X, Y, etc...) pour le nom des variables locales.
- Un message s'affichera pour les fonctions que l'on sait buggées.
- Des astuces peuvent être consultées pour scripter plus rapidement.

Scripter avec style pour un script plus sûr

C'est sûrement discutable puisqu'il s'agit plus de style personnel que d'un fait. Et ça sonne vraiment prétentieux ☺. Néanmoins, cela devrait être utile pour le nouveau venu ; voilà donc quelques commentaires à propos de mon opinion personnelle sur la façon d'écrire un script :

- Utilisez les commentaires. Pour les scripts les plus courts, cela peut sembler inutile, mais même là, vous pourriez indiquer quel(le) mod/quête/objet est concerné par ce script, quel est son objectif principal, etc... Pour les scripts plus longs, cela devient indispensable : pour vous-même si vous vous arrêtez de scripter pendant quelques jours ; et pour les autres s'ils veulent tirer des enseignements de votre script. Explicitez vos variables, mettez des headers (en-tête) sur la partie principale de votre script, commentez les lignes de code les plus importantes, etc...
- Utilisez les variables d'état avec style. Dues à la nature "exécuté une fois par frame ", elles sont le principal moyen de structurer votre script pour les évènements séquentiels. Voici quelques choses à faire :
  A) Limitez-vous au nombre minimal de variables nécessaires à votre script.
  B) Utilisez les *elseif* pour chaîner les différents états d'une variable ensemble et pas des blocs *if* séparés – cela doit être l'élément structurel principal de votre script (ce n'est pas toujours possible ou nécessaire, mais si cela l'est, cela vous évitera beaucoup d'ennuis).
  C) Vérifiez que les états dans les *elseif* s'organisent du plus petit au plus grand et suivent l'ordre logique des évènements – cela vous aidera pour tout organiser et ainsi éviter les bugs. Faire des sauts au hasard entre plusieurs états d'une même variable peut s'assimiler à une utilisation irréfléchie des GOTO du bon vieux BASIC.
  D) Utilisez-les à fond : avancez à petits pas. Je ne peux pas dire combien de scripts buggés se sont mis à fonctionner simplement parce que j'ai déplacé quelques fonctions vers un nouveau "bloc d'état". Parfois ce n'est absolument pas logique – mais si vous pouvez entamer un nouveau pas en toute sécurité, faites-le.
- Décidez-vous pour un style de programmation. Le script des TES est peu regardant quant à la syntaxe. Vous pouvez écrire les fonctions en minuscules, majuscules ou tel qu'on le fait dans ce manuel : *(if SomeFunction == 1)* ou *if (somefunction)*. Quoi que vous choisissiez, tenez-vous y.
- Utilisez des noms de variables parlants. Un nom qui reflète le rôle de cette variable rend le script plus lisible. Si vous utilisez des variables globales, donnez-leur un nom unique, par exemple ajoutez vos initiales au début du nom – essayez simplement de réduire les chances qu'un autre mod ait eu la même idée de nom pour une global – sinon cela va mettre un bazar monstre !
- Gardez à l'esprit vos utilisations de la fonction Return. La fonction Return est par essence dangereuse – souvenez-vous qu'elle stoppera l'exécution de toutes les lignes situées en-dessous d'elle. Vous pouvez l'utiliser mais avec modération. Si vous avez le sentiment que vous allez beaucoup l'utiliser dans un script, vous devriez plutôt introduire une variable d'état.

Nettoyer votre mod

Lorsque vous travaillez sur un mod, vous voudrez probablement examiner d'autres choses comme des références ou simplement copier-coller des éléments pour vos propres desseins. Le problème, c'est que le TESCS se souviendra que vous avez regardé ces choses si vous les bougez un tant soit peu ou que vous cliquez sur un bouton "OK", même si vous n'avez rien modifié. Avant de diffuser votre mod, vous devriez vérifier qu'il ne possède pas de tels références et les supprimer si nécessaire.
Dans le menu *File*, sélectionnez *Data Files*. Puis choisissez votre mod comme vous le feriez normalement et cliquez sur le bouton *Details*. Regardez la liste des caractéristiques : il s'agit d'une liste de tout ce que votre mod change ou ajoute. Recherchez les entrées que vous ne voulez pas modifier, sélectionnez-les et tapez sur la touche [del] ou [suppr] de votre clavier. La caractéristique est maintenant ignorée. Le chargement puis la sauvegarde du mod supprimera les caractéristiques ignorées du mod.
Une autre solution, plus facile à utiliser, est l'outil TESAME (TES Advanced Mod Editor), disponible sur plusieurs sites, par exemple :
http://theseventhrealm.com/portal/tools.html
NdT : et bien sûr sur wiwiland : www.kalendaar.com/morrowmodules/ dans la section Utilitaires

Un outil plus récent qui m'est devenu indispensable est Morrowind Enchanted Editor, disponible à cette adresse :
http://tfo.rh.rit.edu/esforum/secretmasters/EnchantedSetup0.91c.exe.
Il n'y a pas beaucoup de documentation, mais il offre une interface utilisateur plus sympa que celle de TESAME et il est très puissant.

Pour les scripts, essayez de supprimer toutes les variables globales ou les scripts tout entier que vous auriez pu créés pendant le développement de votre mod et qui ne vous servent plus dans la version finale. Ils alourdissent inutilement l'esp, gâchent probablement de la mémoire ; au mieux cela paraîtra simplement moche.

Voici une liste des indicateurs de changement qu'on trouve dans le tableau *details* de

TESAME:

DIAL – un nouveau topic et/ou modifié

INFO – une réponse d'un dialogue ou une entrée de journal

REFR – une référence d'un objet placé dans le monde du jeu tandis que MISC, CONT, etc... décrivent l'objet réel (même s'il n'y a aucune instance de celui-ci placé dans le monde)

SOUN – un son

NPC_ – un nouveau type de PNJ ou un PNJ modifié

CREA – une nouvelle créature ou un changement appliqué à une créature

LIGH – une nouvelle lumière ou un changement appliqué à une lumière

LTEX – l'application d'une texture de paysage

PGRD – un changement sur une *AI grid*

CELL – évident : signifie qu'une cellule intérieure ou extérieure a été modifiée – dans le cas où votre mod ne touche pas à cette cellule, vous devriez vous en débarasser. Cela affecte automatiquement tous les changements effectué dans cette cellule, du moins je pense que c'est automatique

SCRPT – un Script – beaucoup de gens laisse trainer des scripts de tests dans leur mod – ce n'est pas "propre" selon moi.

MISC – un nouvel objet de type *miscellaneous* ou un changement appliqué à un objet de ce type – vérifiez les noms et effacez si le changement n'est pas provoqué par votre plugin

ACTI – un activateur – voir ci-dessus

CONT – un container – très sensible : soyez certain que vous n'avez modifié qu'un nouvel identifiant (la copie d'un container) et pas le container original – ou alors ils seront tous modifiés.

STAT – un objet de type static – voir ci-dessus

Soyez prudent en nettoyant les dialogues : lorsque vous entrez un nouveau dialogue dans un topic, cela modifie aussi les topics au-dessus et en-dessous de celui dans lequel vous avez fait une insertion – c'est parce que les réponses dans les topics sont organisées en liste chaînées : chacune d'entre elles contient une information sur la ligne suivante, ce qui permet de rendre plus rapide le processus de recherche de la réponse correcte dans les topics. Ne nettoyez pas les topics qui seraient affichés à cause de ce phénomène – cela créerait une erreur vous indiquant la ligne suivante est différente pour une réponse spécifique dans un topic (**NdT** : en anglais,ça donne : *an error telling you that the next line is different for a specific response in a topic*). Vous pouvez recréer le lien en déplaçant le topic en question en haut et en bas pour ré-enregistrer l'ordre des topics avec le TES CS.

Références persistantes (On References Persist)

La fenêtre objet du TESCS a une *checkbox* nommée "*References persist*". Cocher cette *checkbox* assure qu'une référence (une instance d'un objet dans le monde du jeu) est toujours disponible pour être référencée par un script, même si le joueur se trouve dans une cellule différente ou qu'il n'a pas encore 'rencontré' l'objet.
Si un script utilise une référence spécifique à un objet tel que

```
Objet_référencé -> Enable
```

Alors "Objet_référencé" devra avoir *References persist* coché.
Le référencement indirect tel que :
PlaceAtMe mon_objet 1 1 1
Player->AddItem mon_objet 1
ne requiert normalement pas de référence persistante. Certaines fonctions font exception à la règle.
Par exemple

```
GetDistance, Objet_référencé
```

requiert que Objet_référencé ait été placé dans le monde et *references persist* coché.
Les Acteurs (PNJs et Créatures) sont toujours persistants.
(Merci à Nigedo pour les infos supplémentaires)

Limitations de l'éditeur de Script

Nombre de caractères : il existe une limite au nombre de caractères maximum par script. C'est quelque part près de 30000 caractères (la vraie limite est sûrement 32767, ce qui correspond à la valeur maximale pour un entier signé codé sur 16-bits, soit la longueur d'un script telle qu'elle est stockée dans l'.esp – merci à Horatio pour cette info). Si cela arrive, vous ne pourrez plus taper dans l'éditeur. Pour gagner des caractères, essayez les choses suivantes :

- Supprimer des caractères
- Utiliser des noms de variables plus courts
- Découper le script en plusieurs parties gérées par un script global ou attachées à un objet séparé en tant que script séparé.

Nombre de lignes : il y a aussi un nombre de lignes maximum. Cela semble variable : les rapports sur les forums parlent d'une plage de 900 à 1500 lignes de code ; c'est probablement plus la limite pour le script compilé que le nombre total de lignes : les espaces et les commentaires ne comptent pas. Vous serez averti par un message d'erreur à la sauvegarde du script.

Nombre limite de If-elseif : il y a une limite sur le nombre maximum de conditions if-elseif qui peuvent être utilisées dans un script. Je ne suis pas sûr du nombre absolu (j'ai entendu 127 et 256). Il existe aussi une "profondeur" maximum de if imbriqués – ce serait 10 (merci Riiak) – et un nombre de lignes maximum à l'intérieur d'un bloc if - endif.

Gagner du temps CPU/machine

Si vous développez un mod avec plein de scripts ou si les scripts sont longs et complexes, vous voudrez sans doute éviter le gâchis de CPU. Il y a un certain nombre de choses à faire :
Si le script n'a pas besoin d'être exécuté à chaque frame, **placez un petit compteur :**

```
Begin Mon_script_super_long

Short compteur_de_frame
If ( compteur_de_frame < 10 )
       set compteur_de_frame to ( compteur_de_frame + 1 )
       Return
Endif
set compteur_de_frame to 0

[le script super long vient ici]
End
```

Ce petit morceau de code, devant être placé au début du script, permettra à celui-ci (ou plutôt au script principal, celui qui est gourmand en temps CPU) de n'être exécuté qu'une fois toutes les 10 frames. Vous pouvez faire la même chose avec un timer et exécuter le script toutes les 3 secondes ou toutes les minutes.
Exécuter un script si le joueur est suffisament proche. Si vous avez scripté une balle magique rebondissante ou tout autre effet visuel, le script n'a aucune raison de tourner si le joueur ne voit pas cet effet. Placez donc quelque chose comme ça au début du script :

```
If ( GetDistance, player < 5000 )
       Return
Endif
```

Abréger les scripts qui ne sont plus nécessaire. Si vous avez des scripts locaux qui n'ont plus besoin de tourner une fois un certain point atteint, par exemple à la mort d'un Acteur, autant réduire leur appétit en CPU : placez les quelques lignes suivantes au début du script :

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

If ( GetHealth <= 0 )
       Return
Endif
```

Terminer les scripts globaux. Souvenez-vous que les scripts globaux continuent de tourner tout le temps jusqu'à ce que vous les stoppiez avec *StopScript*. Vous pouvez faire des scripts globaux do-once en plaçant la commande *StopScript* à la fin :

```
Begin do_once_global_script

[votre code ici]

StopScript "do_once_global_script"

End
```

Préférez l'utilisation de scripts locaux à celle de scripts globaux. Avec les locaux, vous êtes certain qu'ils ne s'exécutent que lorsque vous êtes dans les environs. Réfléchissez bien avant de faire un script global : il est possible qu'un script équivalent local existe.
Soyez prudent avec les boucles while, GetDetected, GetLOS et autres fonctions coûteuses. Utilisez les méthodes décrites ci-dessus (compteur ou timer) pour être sûr qu'elles ne sont pas appelées trop souvent.
Stopper l'exécution du script dans le mode menu. À moins que vous n'ayez une excellente raison, placez toujours au début de votre script les lignes suivantes pour éviter des ralentissements de curseur et d'autres effets non recherchés.

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

Targeted scripts : scripts ciblés : faire s'exécuter des scripts "globaux" attachés à un objet

```
"ID_objet" -> StartScript "ID_Script"
```

Il est possible d'utiliser la fonction *StartScript* pour faire s'exécuter des scripts globaux liés à un objet ou un Acteur. Ces scripts ressemblent à la fois à des scripts locaux (les fonctions appelées font référence par défaut à l'objet ou l'Acteur que le script cible) et à la fois à des scripts globaux (ils s'exécutent continuellement). Voici un extrait du post de FreshFish qui a découvert cette technique incroyable :

"Voici ce que dit le fichier d'aide sur la fonction *StartScript*:

Cette fonction démarre l'exécution d'un script. C'est un script global. Il n'est rattaché à aucun objet : les fonctions de déplacement, de rotation, de vérification des distances et autres n'ont aucune raison d'être dans un script global.

Cela n'a pas de sens, vous n'avez jamais essayer de démarrer un script à partir d'un dialogue ? ou à partir du script attaché à un PNJ ? Eh bien, cela fonctionne parfaitement, les fonctions AiTravel ou PositionCell du script appelé s'appliquent à l'objet appelant la commande StartScript. J'appelle ces scripts des scripts 'ciblés' ou *'targeted' scripts*. Les scripts ciblés s'exécutent continuellement tout comme les globaux : vos PNJs peuvent donc vaquer à leurs affaires lorsque vous êtes dans d'autres cellules. Et si vous démarrez un autre script à partir d'un *targeted script* alors ce script hérite de la même cible : vous pouvez donc les chaîner à votre convenance.

Il semble donc que la limitation d'un seul petit script par objet n'a plus lieu d'être.

Je tiens à préciser qu'un script ciblé est différent d'un script 'local' : en effet, du point de vue du dialogue ou des autres scripts, les variables définies dans un script ciblé ne sont pas considérées comme locales à l'objet.
Si un script est stoppé puis redémarré, toutes les variables (état) sont préservées, ce qui inclue les variables de type DoOnce ; soyez donc vigilant si vous souhaitez utiliser votre script plusieurs fois : vous aurez peut-être besoin de réinitialiser vos variables manuellement. Je ne sais pas ce qui se passerait si vous tentiez de démarrer un scipt ciblé sur plus d'une cible en même temps mais je doute que cela soit très joli."

>Peut-on démarrer un script sur un objet, puis après qu'il ait stoppé, en démarrer un second, différent, sur le même objet ?
Vous pouvez démarrer un script différent sur le même objet à n'importe quel moment, pas besoin d'attendre que le premier se termine. Il peut par contre y avoir un conflit en fonction du contenu de vos scripts : autant éviter d'aller dans deux directions à la fois.

>Qu'est-ce qui se passe avec les objets qui possèdent déjà un script local ?
>Y-a-t-il des conflits ?
Pas de problèmes là non plus : j'ai un PNJ qui enfile une robe dès qu'il pleut grâce à un script local et j'utilise aussi un script 'ciblé' pour la faire suivre un autre PNJ.

Quelques informations supplémentaires postées par Riiak sur les forums Morrowind Mods :

a) Les variables se trouvant dans les scripts ciblés ne sont PAS considérées comme locales et sont donc inutilisables dans les dialogues. Cela vient du fait que le script est global et non pas local.
b) Un script ciblé peut être démarré à partir d'un dialogue générique par un PNJ générique. Ainsi vous pouvez écrire un dialogue pour parler de "Croisés" et démarrer le script à partir de là, quelque soit l'identifiant du PNJ. (cela autorise de nombreuses situations intéressantes). Dans la même veine, voici une suggestion de Cortex : vous pouvez utiliser les extraits de voix génériques (comme les paroles que prononcent les Acteurs lorsque vous vous approchez d'eux, lorsque vous initiez un combat ou lorsqu'ils sont touchés ; vous pouvez démarrer ces scripts à partir d'eux car ils ont comme tous les dialogues un champ résultat (*result field*). Cela ouvre beaucoup de possibilités !
c) Les scripts ciblés vous permettent d'accéder à l'inventaire des PNJs (avec les mêmes limitations qu'un accès normal : vous devez connaître les identifiants d'objets, etc...)
d) (Pour autant que je puisse en dire) Ne peuvent être utilisés pour le partage d'inventaire des compagnons de Tribunal, mais peuvent servir pour les simples scripts d'escorte.
e) (Pour autant que je puisse en dire) On peut avoir des copies multiples du même script ciblant le même PNJ (cela peut aider, mais cela causera sûrement des problèmes)

Il est préférable que le script se termine de lui-même : c'est le moyen le plus 'propre' et cela évite d'avoir plusieurs copies qui s'exécutent en même temps.

Utilisés dans les dialogues, les scripts ciblés s'attachent apparemment toujours au PNJ donnant la réponse et pas à un objet référencé : ID_Objet -> StartScript "nom_du_script" ne marchera pas, le script sera toujours attaché au PNJ qui vous parle. (Forum info / Argent).

Attention en utilisant des scripts ciblés avec des objets non uniques. Cela marchera à la première exécution, mais après le chargement d'une sauvegarde, le script s'attachera à la première instance de l'objet trouvée dans la base de données (c'est similaire à ce qui a été dit pour *GetDistance*), au lieu de s'attacher à la référence qui a appelé le script la première fois (forum info, MentalElf).

Détecter le chargement d'une sauvegarde :

Voici quelques raisons pour vouloir tester le chargement d'une sauvegarde :
1) continuer de jouer une musique spécifique (les musiques mp3 se réinitialisent au chargement).
2) pour qu'un PNJ bascule de nouveau en mode course ou discrétion.
3) pour redonner à un objet sa bonne dimension (s'il est en dehors de l'échelle 0.5 à 2.0).

Il y a plusieurs façons de faire:

JOG propose d'utiliser SetJournalIndex:

```
if ( ( getjournalindex "toto" ) != 100 )
   Messagebox "vous venez de recharger, tricheur!!!"
   setjournalindex "toto" 100
endif
```

"toto" correspond à n'importe quel topic du journal qui n'a pas de texte pour l'index 100.
SetJournalIndex placera l'index à cette nouvelle valeur, qu'une entrée pour cette valeur existe ou pas (très utile pour des flags qui n'ont pas besoin de leur propre entrée dans le journal) mais au chargement, l'index sera réinitialisé à la plus grande valeur trouvée correspondant à l'entrée existante dans le journal.
En d'autres termes, au chargement, l'index (pour un topic) prend la valeur de la dernière entrée trouvée ; si on a auparavant placé cet index à une valeur qui ne correspond à aucune entrée, il suffit de vérifier que cette valeur a changé pour détecter un chargement.

Selon MentalElf, *GetForceRun*, *GetForceSneak*, *GetScale* peuvent aussi être utilisées pour détecter un chargement. Cela vient du fait que *ForceRun* et *ForceSneak* sont réinitialisées au chargement et que l'échelle d'un objet (*scale*) est placée entre 0.5 et 2.0. Selon moi, *ForceRun* est mieux puisque *ForceSneak* place le PNJ en position accroupie.

```
; (objet PNJ)
if ( GetForceRun == 0 )
; le joueur vient juste de charger une suauvegarde
; traitement à développer ici
ForceRun ;remettre le PNJ en course forcée pour le prochain chargement
Endif
```

Une option différente est d'utiliser les scripts de démarrage (*start scripts*) de Tribunal et Bloodmoon. Voici deux exemples de Dinkum Thinkum :

```
begin DT_DoOnce_TribStartScript02
; ce script montre que le script de démarrage ne s'exécute qu'une fois lorsqu'une sauvegarde
; est chargée avec le mod activé
; par DinkumThinkum

; le script exécute la section de code 'Do Once' une seule fois
; à chaque fois que le mod est activé pour une sauvegarde donnée

; début de ka section 'Do Once' qui doit s'exécuter à chaque chargement d'une sauvegarde
; nécessitant l'activation de ce mod

MessageBox, "Vous verrez ce message à chaque fois que vous chargez une sauvegarde nécessitant
d'activer ce mod.", "OK"

; fin de la section 'Do Once'.

StopScript DT_DoOnce_TribStartScript02

end DT_DoOnce_TribStartScript02
```

```
begin DT_DoOnce_TribStartScript01

; le script de démarrage de Tribunal ne s'exécutera qu'une fois
; par DinkumThinkum

; le script exécute la section 'Do Once', lorsque le mod est activé mais que vous chargez une
; sauvegarde qui ne nécessite pas l'activation de ce mod (ou que vous commencez une nouvelle
; partie)

; le script exécute la section 'Reload', à chaque fois qu'une sauvegarde réalisée avec le mod
; activé est chargée, le mod étant actif (coché)

; l'une ou l'autre des sections de code peut être omise mais il faudra toujours les structures
de contrôles

;Note: LES DEUX 'StopScripts' sont nécessaires. Faites-moi confiance...

; DT_DoOnce_TSS01 - variable globale, ne sera pas réinitialisée par StopScript
; (initialisée à 0)

if ( DT_DoOnce_TSS01 == 1 )

; début de la section 'Reload', ne s'exécute qu'une fois à chaque fois que le mod est rechargé

MessageBox, "Vous avez chargé ce mod avec une sauvegarde faite lorsque ce mod était activé.",
"OK"

; fin de la section 'Reload'

StopScript DT_DoOnce_TribStartScript01
Return
endif


; début de la section 'Do Once', ne s"exécute que lorsque le mod est chargé pour la première
; fois

MessageBox, "Vous avez chargé ce mod avec une sauvegarde faite sans que le mod ait été
activé", "OK"

;fin de la section 'Do Once'


set DT_DoOnce_TSS01 to 1
StopScript DT_DoOnce_TribStartScript01

end DT_DoOnce_TribStartScript01
```

Utilisation de la variable CharGenState – Désactiver menus et sauvegardes

L'option sauvegarde du menu principal, ainsi que la touche de sauvegarde rapide, sont disponible selon la valeur de la variable globlale *CharGenState*. Placez-là à une valeur différente de -1 (99 par exemple) pour que le joueur n'ait plus la possibilité de sauvegarder sa partie. Réinitialisez-la à -1 pour que tout revienne à la normale.
Ce procédé entraîne quelques effets secondaires. Tout d'abord, vous n'aurez plus accès à l'écran des menus. Cela peut se corriger en utilisant la commande *EnableStatsMenu* – TOUS les menus seront de nouveau accessibles, et pas seulement la fenêtre des stats. Cependant, le journal, les racourcis et le menu des raccourcis (F1) restent désactivés et vous ne pourrez pas fouiller les corps. Je n'ai pas trouvé de moyen pour dépasser ces limitations.
Le procédé peut entraîner de nombreux conflits ; il est préférable de prendre quelques précautions : vérifier que le joueur ne se trouve pas dans la séquence de génération du personnage et vérifier que la valeur de la variable est bien réinitialisée après utilisation.
(Forum info / Erstam)

Détecter l'utilisation de parchemins et de livres

Voilà une tâche étonnament difficile ! en effet, *OnActivate* et *OnPCEquip* sont toutes les deux nécessaires ET ne fonctionnent pas comme prévu. Kir a trouvé une solution qu'il montre dans son script 'invoquant' une lettre de crédit bancaire.

```
Begin BankLetter10

short button
short messageOn
short invoke
short gone
short goneway
;short testdist
short PCSkipEquip
short OnPCEquip

set PCSkipEquip to 1
; désactiver si utilisé (l'objet n'est utilisable qu'une seule fois)
if ( gone == 1 )
	if ( goneway == 1 )	; activé en tant qu'objet externe
		Disable
	else				; équipé à partir de l'inventaire
		startscript BankLetter10Remove
	endif
	set gone to 0
	return
endif

if ( OnActivate == 1 )
	Set messageOn to 2
	set goneway to 1
endif

If ( OnPCEquip == 1)
	Set messageOn to 2
	Set OnPCEquip to 0
	set goneway to 2
endif

if ( messageOn == 0 )
	return
endif

if ( messageOn == 2 )
	MessageBox "Voulez-vous invoquer la Lettre de Crédit?" "Oui" "Non"
	Set messageOn to 1
	return
endif



if ( messageOn == 1 )
	set button to GetButtonPressed
	if ( button == 0 )
		Set invoke to 1
		Set messageOn to 0
;		return
	endif
	if ( button == 1 )
		Activate
		Set messageOn to 0
		return
	endif
endif

if ( invoke == 1 )
	PlaySound "Item Gold Up"
	Player->AddItem, Gold_001, 10000
	set gone to 1
	set invoke to 0
endif
End
```

Erstam a posté un script bénéficiant de caractéristiques encore meilleures : on constate qu'il existe un problème intéressant avec les variables *OnPCEquip* / *SkipEquip*:
"Inspiré par le script BankLetter de MSFD 7, j'ai découvert un moyen d'exécuter un script personnalisé soit lorsque un livre/parchemin est "équipé" à partir de l'inventaire du joueur soit lorsque qu'il est activé dans l'environnement du jeu, *tout en affichant normalement le livre/parchemin*. Etonnamment, c'est la variable *PCSkipEquip* qui est positionnée à 1 lorsqu'on dépose un livre sur le portrait du joueur, au lieu de la variable *OnPCEquip*. Voici le code que j'ai utilisé : "

```
Begin activateBook

short OnPCEquip
short PCSkipEquip
short doOnce
short actionFlag ;variable qui simule l'activation, que ce soit dans l'inventaire ou en dehors

if ( actionFlag == 1 )
	if ( doOnce == 0 )
		; insérez votre code ici
		set doOnce to 1
	endif
	set actionFlag to 0
endif

; PCSkipEquip est placée à 1 à chaque fois qu'on équipe un livre à partir de l'inventaire
if ( PCSkipEquip == 1 )
	set PCSkipEquip to 0
	set actionFlag to 1
	return
endif

; les lignes suivantes sont importantes, sinon le livre ne peut pas être ramassé
if ( MenuMode == 1 )
	return
endif

; pour l'activation du livre lorsqu'il est placé dans l'environnement de jeu
if ( OnActivate == 1 )
	set actionFlag to 1
	Activate
endif

End
```

Cela devrait marcher sans la condition doOnce, au cas où vous voudriez exécuter une action à chaque fois que vous vous équiperez du livre. Cependant je n'ai pas fait de tests.

Forcer les Acteurs à changer d'armes

Comme la fonction *Equip* ne fonctionne pas, la seule façon de procéder consiste à retirer les objets de l'inventaire de l'Acteur ou alors de modifier ses talents pendant l'exécution.
Voici un exemple que j'ai utilisé pour forcer un garde à passer de l'épée à l'arc et vice-versa :

```
Begin HBCaravanGuardAI

; ce script rend le garde plus dangereux en le faisant passer de l'arc à l'épée lorsque le
joueur s'approche
short currentarrows ; compteur de flèches : nombre courant de flèches
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
elseif ( GetDistance, Player >= 120 )
        if ( currentarrows < storearrows )
                        AddItem "arrow of wasting flame", 1
        else
                set doonce to 0
        endif
endif

End
```

L'exemple suivant vient de Bethesda : il fait la même chose mais en utilisant la méthode de modification des talents (plus élégante que la mienne ☺):

```
begin marksmanToggle

short counter         ;compteur de frame : pour réduire la fréquence d'exécution du script
short myMarksman      ;valeur locale du talent

if ( MenuMode == 1 )
        return
endif

if ( counter < 20 )
        Set counter to counter + 1
        Return
endif
if ( myMarksman == 0 ) ;récupère la valeur du talent
        set myMarksman to GetMarksman
endif
if ( GetMarksman > 0 ) ;si c'est un bon archer
        if ( GetDistance Player < 400 ) ; et que le joueur est suffisament proche
                SetMarksman 0  ;alors il devient un très mauvais archer
        endif
else
        if ( GetDistance Player > 600 ) ;le joueur est assez loin
                SetMarksman myMarksman   ; restaure la valeur du talent
        endif
endif
;Notes originales de Bethesda (non traduites) équivalentes aux commentaires ci-dessus
;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
```

Piège à flèche ou magique (Arrow- or magical traps)

J'avais tout d'abord posté ce script pour montrer l'usage de *SetDelete*, mais je pense que sa place est ici.

Lorsque ce script est placé sur un objet, dès qu'il apparaît dans le jeu (ou dès qu'il est 'rencontré' s'il a été placé dans l'éditeur), il calculera la position du joueur et se déplacera dans sa direction à un taux très rapide (déterminé avec *GetSquareRoot*). Lorsque ce point est atteint ou que l'objet est à portée du joueur, il explose (grâce à la fonction *ExplodeSpell*) et se désactive (avec *Disable*). Après cela, on attend quelques frames pour que le sort disparaisse et on l'efface (avec *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
```

```
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
```

Téléportation à l'aide de scripts

La téléportation vers des intérieurs et vers des extérieurs en particulier n'est pas si banal : il peut y avoir des problèmes avec les cellules environnantes : chargement incomplet (certaines parties du paysage peuvent être absentes) ou crashs. Une solution, suggérée par Aftershock_81:

```
COE 0 0 ;fonction CenterOnExterior

Player->SetPos x xpos
Player->SetPos x ypos
Player->SetPos x zpos

FixMe
```

où *FixMe* est censé recharger la cellule de destination dans le cas où *SetPos* aurait effectué un chargement incorrect.Nigedo a rapporté des problèmes avec cette approche et a proposé une autre solution:

*"*Parfois le chargement de la cellule échoue malgré *FixMe*. Les autres fois, on aura droit à des messages d'erreurs bizarres ou à des freezes du jeu. Avec l'aide de Grumpy et Mode_Locrian (merci les gars), j'ai écrit un script qui marche relativement bien en utilisant des coordonnées différentes pour les tests. Le joueur est déplacé par étapes (en fait par cellules), ce qui permet un chargement correct 'par approche' de la cellule de destination.

Utilisez FadeOut au départ (ou au déclenchement du script) et FadeIn à l'arrivée pour obtenir un rendu plus esthétique.

Il y a deux étapes. Les coordonnées de départ extérieures du joueur sont tout d'abord stockées dans 3 variables globales lorsqu'il est transporté dans un intérieur. Ensuite, le script suivant, appelé par *StartScript* lorsque le joueur demande à retourner aux coordonnées extérieures de départ.

J'ai essayé d'optimiser ce script en diminuant successivement la valeur des 'sauts' des *SetPos* empilés (j'avais commencé avec 131072gu – NdT : gu = Game units, voir page ...) et ce afin de réduire le nombre de cellules chargées. Malheureusement cela échouait de temps en temps : lorsque la distance entre 0,0 et la destination était supérieur à 8192gu, la cellule n'était pas toujours chargée et donc le but de cette approche n'était pas atteint.

Cette méthode est un peu lourde, mais il semble qu'on puisse compter dessus."

```
Begin script_placePC

;Global Long Start_PCX
;Global Long Start_PCY
;Global Long Start_PCZ
Float xpos
Float ypos
Float zpos
Long higher ;ces deux variables permettent de déterminer si on est au-dessus ou en-dessous
Long lower  ; de la variable de destination
Short step ;détermine l'étape d'avancement du script

If ( step == 0 ) ;centre le joueur sur la carte, ie le place au coordonnées 0,0
       Player->COE 0 0
       Set step to 1
       Return
Endif

Set xpos to ( Player->GetPos x )
Set ypos to ( Player->GetPos y )

If ( step == 1 ) ;premier pas on cherche à atteindre la coordonnée x
       Set higher to ( Start_PCX + 8192 )
       Set lower to ( Start_PCX - 8192 )
       If ( xpos > higher )
              Set xpos to ( xpos - 8192 )
              Player->SetPos x xpos
       Elseif ( xpos < lower )
              Set xpos to ( xpos + 8192 )
```

```
                Player->SetPos x xpos
        Else ;lorsqu'on a atteint la coordonnée x souhaitée
                Set step to 2
        Endif
        Return
Elseif ( step == 2 ) ;deuxième pas, on cherche à atteindre la coordonnée y
        Set higher to ( Start_PCY + 8192 )
        Set lower to ( Start_PCY - 8192 )
        If ( ypos > higher )
                Set ypos to ( ypos - 8192 )
                Player->SetPos y ypos
        Elseif ( ypos < lower )
                Set ypos to ( ypos + 8192 )
                Player->SetPos y ypos
        Else    ;lorsqu'on a atteint la coordonnée y souhaitée
                Set step to 3
        Endif
        Return
Elseif ( step == 3 ) ;place le joueur définitivement
        Set xpos to Start_PCX
        Set ypos to Start_PCY
        Set zpos to Start_PCZ
        Player->SetPos x xpos
        Player->SetPos y ypos
        Player->SetPos z zpos
        Set step to 0
        StopScript script_placePC
Endif

End
```

Notes: Parfois, mon PJ est arrivé à destination mais mort. Hé hé...je ne peux que supposer qu'il avait 'dérangé' certaines créatures en chemin. C'est donc une bonne idée d'encapsuler tout le script à l'intérieur d'une condition '*ToggleGodMode*'. Pour l'esthétique, utilisez *FadeOut* et *FadeIn* comme je l'ai dit avant. On peut aussi améliorer les performances en appelant *ToggleMenus* après le *COE* du premier pas, ce qui a pour effet de cacher le HUD et par conséquent tous les chargements aux yeux du joueur. Les menus/HUD peuvent être réactivés à la fin du script avec la combinaison de fonctions suivante : *ToggleMenus*, *MenuTest*, *MenuTest*.

Tester la présence d'un autre mod

(par Ragnar_GD, traduction par Flox)
Certaines personnes veulent savoir, à partir d'un script, si un autre plugin spécifique est chargé. C'est possible dans certaines conditions.

C'est possible si le plugin à tester

a) possède une variable globale unique
b) positionne la valeur de cette variable globale à partir d'un script

Si vous connaissez une telle variable, implantez-la dans votre propre script. Puis vérifiez pendant l'exécution que la valeur de la variable est bien modifiée ; par exemple, supposons que l'autre mod possède une variable globale nommée « Variable_exterieure » et que vous vous servez de la variable de cette façon :

```
If (Variable_exterieure != 0)
       MessageBox "j'ai détecté le plugin !"
endif
```

Cette méthode ne fonctionne pas, si « Variable_extérieure » a une valeur préréglée (puisqu'avec cette méthode, la valeur sera remplacée) ou le plugin étranger n'a pas encore changé la valeur en une autre valeur autre que « 0 » au moment où ce test est lancé.

Avertissement : ne **jamais** implanter une variable globale dans votre plugin avec une valeur préréglée, mais la laisser avec la valeur « 0 » par défaut, et écrire le script plus tard (c'est-à-dire avec un script de démarrage). Car si quelqu'un utilise un nom similaire pour la variable globale, votre plugin pourra planter.

Le fait d'écraser une variable globale d'un autre plugin ne vous avertira pas de l'apparition de bugs sérieux !

(Note de GBG: – une raison de plus pour que les gens essayent d'utiliser des noms uniques quand c'est possible dans leurs mods – ne nommez pas votre globale "check" ou « verif » mais appelez-la plutôt <vos initiales_check : "YI_check" – Tout ceci peut vous permettre d'éviter un grand nombre de problèmes.)

Démarrer des scripts globaux de manière sure – ne pas recourir au script main

Avec l'extension, il n'y a plus aucun problème – ajoutez juste votre script à la liste des "*Start Scripts*" en sélectionnant ***Edit Start Scripts*** du menu ***Gameplay***. Ce script sera maintenant automatiquement démarré au chargement d'une partie, tout comme le script main. Pour ceux qui n'ont pas Tribunal/Bloodmoon:

Beaucoup de moddeurs ajoute une ligne au script main (le seul script qui est exécuté par défaut lorsqu'on commence une nouvelle partie et qui est toujours en train de s'exécuter) pour être certain qu'un script global essentiel à leur mod démarre. Par exemple :

```
StartScript "Mon_Script"
```

Bien que cela fonctionne, cela peut facilement provoquer avec des mods qui utilisent la même approche – en effet, seuls les changements du dernier plugin chargé seront actifs dans le jeu. De plus, il existe une alternative : en mettant un activateur (invisible) dans le bureau des taxes et du recensement de Seyda Nihyn avec le script suivant attaché, vous êtes sûr que le script est démarré pendant la phase de création du personnage.

```
Begin Lanceur_de_script
If  ( ScriptRunning Mon_Script == 0 )
       StartScript Mon_Script
       ; MessageBox "Mon_Script a été activé" ; dites-le au joueur si vous voulez
endif
End
```

Pour être certain qu'un script global démarre dans un jeu déjà entamé, vous pouvez utiliser une méthode similaire en plaçant des activateurs dans des lieux qu'on visite souvent. Si vous en avez un à Balmora, Vivec, Sadrith Mora, Dagon Fel, et Caldéra, voire même dans la forteresse du joueur, je suis sûr que cela ne prendra pas longtemps avant que le script ne démarre. Vous pouvez aussi utiliser un objet requis dans votre mod pour le démarrer. Par exemple, pour l'excellent *Bank mod* de Indestructible, je me suis fait une version qui attache le script ci-dessus à une bannière d'une banque et qui démarre le script "*interest*" (les intérêts). Comme cela, je suis sûr que le script tourne avant même que le PJ ne soit entré dans la banque.

Utiliser les sons pour détecter les évènements

Comme cela me semble être une excellente idée (merci à BalorNG), je le mentionne ici encore, bien que j'en ai déjà parlé dans la section son. Vous pouvez utiliser la fonction *GetSoundPlaying* pour déterminer certains évènements du jeu qui ne sont pas accessibles autrement. Regardez les sons du menu ***Gameplay/sounds***, cela pourra vous donner quelques idées : savoir si quelqu'un vient de tomber, s'il y a un monstre tout près, si quelqu'un vient de recevoir un coup etc.
Voici quelques informations à ce sujet (merci Horatio):
GetSoundPlaying est une fonction très puissante qui peut être utilisée pour détecter si le PJ (et je suppose tous les acteurs) est en train de faire une certaine action comme lancer un sort ou manier une arme. J'ai utilisé cela dans mon mod spellcasting pour savoir si le PJ lance un sort et de quelle école il s'agit. Le format est le suivant :

```
if ( player->GetSoundPlaying, "ID Son" == 1 )
;faire quelque chose de cool ici
endif
```

regardez dans le menu des sons du TESCS pour trouver quelle ID Son correspond à une action spécifique. Par exemple, "illusion cast" correspond au lancement d'un sort d'illusion. Vous aurez probablement à faire quelques essais.
Note: pour une raison inconnue, l'ID son "drink" provoque une erreur ; donc ne vérifiez pas si le joueur est en train de boire une potion.

Batailles de grande envergure

(par Horatio)
La façon la plus facile d'organiser une scène de bataille entre deux groupes de PNJs est d'utiliser les commandes d'IA. Voyons un exemple : le PJ est allié avec une poignée de gardes impériaux : ils vont se battre avec des assassins de la Confrérie Noire. Tout d'abord placez le taux d'attaque (bouton AI) des PNJs de la Confrérie à 100 pour qu'ils attaquent le PJ à vue. Après modifiez l'IA des légionnaires avec :

```
AIFollow, player, 0,0,0,0,0
```

Vous pouvez faire cela dans des scripts attachés aux PNJs de la Légion ou avec un script externe. Le comportement par défaut de la commande *AIFollow* est d'attaquer tout ce qui s'en prend à la personne qu'ils suivent. Dès que la Confrérie attaquera le PJ, les gardes commenceront à riposter. Bataille grandiose garantie !
J'ai utilisé une variation de ceci dans le mod GIANTS pour convaincre les gardes d'attaquer les monstres qui sont en fait des PNJs (vampires, ombres, géants, gorgones, etc).

Ridable objects : Un guide pour créer des objets 'montables'

(Par MadMax_001, traduit par Aqualonne)

MadMax partage avec nous ses connaissances sur la façon de créer des 'ridable objects' (tels que des bateaux), et traitera plus particulièrement des problèmes soulevés et de la façon de les résoudre ; les nombreuses informations dont nous allons parler explorent en profondeur le thème. Vous pouvez regarder ses scripts dans les mods de l'académie de pêche (c.f. 'Fishing *Academy*') et le tapis magique ('*Magic Carpet'*) ; ne les utilisez pas sans son accord !

Sélection des objets

Pratiquement tous les objets (les statiques et ceux qui activent) peuvent être utilisés. Ceci dit, il est important de sélectionner le bon type d'objet type : votre script n'en sera que plus efficace ! A présent, quels types d'objets conviennent ? La préférence est donnée aux hauteurs minimales (épaisseur). L'autre facteur important est le point central qui est aussi celui où votre personnage sera. Cela vous épargnera aussi du travail par la suite… Si le point central de l'objet n'est pas celui que vous voulez, vous pouvez résoudre ce problème en important l'objet dans 3D Studio et bouger l'axe vers le point où vous escomptez que soit votre personnage. Je ne vais pas expliquer en détail comment faire dans 3D Studio, il doit déjà y avoir quelques tutoriaux sur la question...

Créer et supprimer des objets

Je suis sûr qu'un bon nombre de moddeurs savent qu'il est uniquement possible de bouger un objet à travers les cellules extérieures dans une certaine distance ; en effet, le jeu fera seulement la mise à jour et les actions pour l'objet et le script dans une certaine distance. Par exemple, si votre personnage dépasse ce paramètre, vous pouvez avoir des problèmes comme le gel de l'objet, ou l'impression pour le joueur qu'il a disparu dans les airs ! Mais, si vous vous reportez à la cellule d'origine, l'objet réapparaîtra.

A présent, pour bouger des objets partout dans les cellules extérieures, l'astuce à utiliser est de créer un nouvel objet. A chaque fois que vous allez dans une nouvelle cellule, la fonction *CellChanged* (**NdT** : indique qu'il y a eu un changement de cellule) aura comme valeur *true* pour un cadre (**NdT** : true signifie que le changement est confirmé). C'est le meilleur moment où remplacer un objet existant par un nouveau. Ceci dit, il y a un gros problème que j'ai découvert : ne jamais créer un objet avec un script d'objet ! En conséquence, il vous faudra utiliser un script global. En même temps, n'oubliez pas de détruire l'ancien objet ou vous aurez tous ces objets dans des cellules différentes, ce qui vous causera des problèmes ultérieurement. Gardez un objet à un moment. Voici un exemple que vous pouvez utiliser :

```
;----------------
; Script d'Objet
;----------------
if ( player->CellChanged == 1 )
Startscript, "Create_obj_script" ; voici le script global
set obj_count to ( obj_count - 1 ) ; parametre global qui compte combien d'objets existent
Disable
SetDelete, 1
endif
```

```
;----------------
; Script Global
;----------------
PlaceAtPC "objectname", 1, 0, 0 ; vous pouvez aussi utiliser la fonction PlaceItem
set obj_count to ( obj_count + 1 )
Stopscript "create_obj_script"
```

Théoriquement, le script précédent devrait fonctionner à merveille, mais en réalité il va vous causer de sérieux problèmes. Détruire un objet après avoir changé la cellule peut parfois entraîner des crashs. C'est particulièrement le cas quand vous avez le chargement d'une grosse zone. Pour résoudre cela, différez la suppression en utilisant un compteur de temps (timer) ; en fixant ce temps à 1,5 secondes il n'y a pas l'air d'y avoir de problèmes. Voici maintenant à quoi votre script d'objet doit ressembler :

```
if ( player->CellChanged == 1 )
Startscript, "Create_obj_script" ; le script global
Disable
set timer_flag to 1
endif

if ( timer_flag == 1 )
set timer to ( timer + GetSecondsPassed )
if ( timer > 1.5 ) ; l'utilisation de notre compteur de temps, le timer
set obj_count to ( obj_count - 1 )
SetDelete, 1
else
return ; arrêter toutes les autres tâches en cours
endif
endif
```

Si vous comptez bouger votre objet à une vitesse très élevée, il est possible que l'objet rencontre un autre changement de cellule pendant notre temps de 1,5 secondes. Vous devez alors vous assurer que l'objet soit détruit avant qu'il soit hors d'atteinte de notre script, sinon ce sera un objet fantôme qui reviendra. Par conséquent, on apporte une modification :

```
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
```

Chuter à partir des objets

Il suffit de supprimer l'effet de gravitation qui s'applique sur le personnage. Vous pouvez le faire en utilisant la lévitation ou la capacité de voler du personnage (**Note** de GBG : vous pouvez aussi utiliser *SetPOS)*. Ceci semble parfait, mais il deviendra impossible de détecter l'utilisation des capacités de course (run) ou de discrétion (sneak). La dernière solution vous permet de détecter les mouvements, mais cela peut aussi engendrer une chute. Et puis d'abord, vous devez sûrement vous demander pourquoi votre personnage chute… La raison est simple : à chaque fois que vous créez un objet, son paramètre de son collision n'est pas mis à jour ; il le sera lorsque vous changerez de cellule. Vous découvrirez que votre personnage semble flotter au travers de l'objet. La bonne nouvelle est qu'on peut remédier à ce problème : en désactivant puis en activant à nouveau l'objet, il sera mis à jour. Afin d'être certain que l'objet est toujours de type solide, utilisez la désactivation et l'activation au moins une fois dans chaque frame du script de l'objet.

Détecter les collisions

C'est là que commencent les ennuis ! Des objets ne peuvent pas entrer en collision avec d'autres objets. Seuls des personnages (joueurs ou non) et des créatures peuvent entrer en collision avec des objets. Précisons que le terme « objet » recouvre donc les statics, activators, et le terrain. La seule façon de détecter les collisions est lorsque votre personnage touche l'objet ; c'est pourquoi j'ai mentionné plus tôt que si vous sélectionnez un gros objet à monter, l'affichage ne sera pas merveilleux jusqu'à ce que votre personnage touche quelque chose. Si cela ne vous dérange pas, tant mieux…

La méthode la plus simple pour détecter les collisions est de comparer les coordonnées de votre personnage avec ceux de l'objet (celui que vous montez) ; bien que cela ne soit pas complètement prouvé, utiliser la fonction GetSquareRoot (la racine carré) dans le script de l'objet peut entraîner des crashs. Vous devez faire attention à deux plans : par exemple, le tapis volant utilise la détection pour les plans horizontaux (axes x et y) et verticaux (axe z). Vous pouvez vous référer à mon script sur la façon dont c'est réalisé ; si quelqu'un a une meilleure idée, qu'il n'hésite pas à nous la faire connaître.

Concernant les sauvegardes

Lorsque vous sauvegardez votre partie en bougeant, les coordonnées de l'objet mis à jour dans votre sauvegarde sont ceux résultant du changement de cellule. Afin de positionner correctement l'objet, il est toujours mieux de garder les paramètres globaux des coordonnées de l'objet. Quand vous chargez la partie, pensez tout d'abord à faire une détection sur les coordonnées de l'objet pour comparer les coordonnées existant à celles globales. Si la différence est importante, mettez l'objet à ses coordonnées globales. De cette façon, lorsque vous chargerez la partie, l'objet sera précisément à la même position lorsque vous la sauvegarderez.

Script de trigonométrie – sinus et cosinus

J’ai demandé à JDGBOLT de partager avec les lecteurs de MSFD la dernière version de son magnifique script de trigonométrie : je suis très heureux de pouvoir le présenter ici. Bien qu’il soit très long, je pense qu’il s’agit d’une ressource inestimable pour quiconque réalise des scripts incorporant des éléments de trigonométrie, par exemple les scripts réalisant des déplacements d’objets. J’ai édité et commenté le script original pour le rendre clair à n’importe quel moddeur. Pensez à remercier JDGBOLT dans vos crédits si vous vous en servez!

Le script calcule le sinus et le cosinus de 3 angles. Ce seront vraisemblablement les angles correspondants aux 3 axes d’un objet, obtenus avec *GetAngle*. Stockez ces angles dans des variables globales

```
float Z_input_variable
float X_input_variable
float _input_variable
```

vous pouvez par exemple le faire à partir du script de l’objet que vous déplacez. Exécuter ensuite le script : Startscript jdtrigscript. Dans cette version, le script se termine de lui-même et place les résultats dans les variables globales suivantes :

```
Float Z_sin
Float Z_cos
Float X_sin
Float X_cos
Float Y_sin
Float Y_cos
```

Vous aurez donc besoin de créer ces variables globales avant de pouvoir compiler ce script. Le script est très rapide et les résultats très précis. Les résultats seront accessibles après une frame.

(remerciements supplémentaires à JDGBOLT pour nous avoir fait part de son script ! il s’agit de la coloration syntaxique utilisée dans MWEdit, voir la section Autres éditeurs de scripts.)

```
begin jdtrigscript
; parJDGBOLT
; édité et commenté pour MSFD par GhanBuriGhan
; calcule à la fois le sinus et le cosinus de 3 angles (par exemple, 3 différents axes)

; pour les résultats intermédiaires de sinus et cosinus
float ax1
float ax2
float ay1
float ay2

; variable principale d’angle
float angle

;pour le stockage temporaire de l’angle :
float angle_source ;contient l’”angle (chargé à partir de la variable globale)
float angle_temp
float angle_temp2

short angleshort ; variable  short pour reprendre l’andgle - for tab values
float angledec ; pour la partie décimale de l’angle

short anglequad ;quadrant

float axfinal ;  porur le résultat final de sinus
float ayfinal ; pour le résultat final de cosinus

float angconvx1 ; pour calculer la partie décimale de sinus et cosinus
float angconvy1
float angconvx2
float angconvy2

short objnum ;compteur pour calculer les 3 angles

;-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
while ( objnum <= 3 ) ; ce script calcule le sinus et le cosinus des 3 angles en une frame
```

```
;exportation des résultats
if ( objnum == 1 )
    set Z_sin to axfinal ; assigne le résultat à la variable globale Z_sin afin de libérer axfinal pour l'axe suivant
    set Z_cos to ayfinal ; idem
endif
if ( objnum == 2 )
    set X_sin to axfinal ;idem
    set X_cos to ayfinal ;idem
endif
if ( objnum == 3 )
    set Y_sin to axfinal ;idem
    set Y_cos to ayfinal ;idem
    ;MessageBox "sine of Z: %.3f  cosine of Z: %.3f", Z_sin, Z_cos
    ;MessageBox "sine of X: %.3f  cosine of X: %.3f", X_sin, X_cos
    ;MessageBox "sine of Y: %.3f  cosine of Y: %.3f", Y_sin, Y_cos
    Set objnum to 0
    StopScript jdtrigscript
    Return
endif
;-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
set objnum to ( objnum + 1 ) ;incrémente le compteur pour faire chaque angle un par un
;-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
; récupère les angles des variables globales
if ( objnum == 1 )
    set angle_source to Z_input_angle
endif
if ( objnum == 2 )
    set angle_source to X_input_angle
endif
if ( objnum == 3 )
    set angle_source to Y_input_angle
endif

;GetAngle renvoie des valeurs comprises entre -180 et +180, on doit donc compenser pour calculer à partir de valeurs
;comprises entre 0 et 360°:
if ( angle_source < 0 )
    set angle_source to ( 360 + angle_source )
endif

set angle to angle_source

;-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
;détermine le quadrant. De cette façon, on a seulement besoin de calculer le sinus et le cosinus pour 0-90°
if ( angle <= 90 )
    set angle_temp2 to angle
    set anglequad to 1
elseif ( angle <= 180 )
    set angle_temp2 to ( 180 - angle )
    set anglequad to 2
elseif ( angle <= 270 )
    set angle_temp2 to ( angle - 180 )
    set anglequad to 3
elseif ( angle <= 360 )
    set angle_temp2 to ( 360 - angle )
    set anglequad to 4
endif

; récupère la partie décimale de l'angle
set angleshort to angle_temp2
set angledec to ( angle_temp2 - angleshort ) ; la différence entre le float et le short correspond à la partie décimale
set angle_temp to angleshort
;-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
; les valeurs qui suivent correspondent aux valeurs pré-calculées de sinus et cosinus:
if ( angle_temp <= 10 ) ; on arrange les angles en paquets de 10 pour la vitesse (et la limite des blocs if ?)
    if ( angle_temp == 0 )
        set ax1 to 0.000000 ; sinus
        set ay1 to 1.000000 ;cosinus
        set ax2 to 0.017452 ; sinus de angletemp+1
        set ay2 to 0.999847 ;cosinus de  angletemp+1
    elseif ( angle_temp == 1 )
        set ax1 to 0.017452
        set ay1 to 0.999847
        set ax2 to 0.034899
        set ay2 to 0.999390
    elseif ( angle_temp == 2 )
        set ax1 to 0.034899
        set ay1 to 0.999390
```

```
            set ax2 to 0.052333
            set ay2 to 0.998629
        elseif ( angle_temp == 3 )
            set ax1 to 0.052333
            set ay1 to 0.998629
            set ax2 to 0.069756
            set ay2 to 0.997564
        elseif ( angle_temp == 4 )
            set ax1 to 0.069756
            set ay1 to 0.997564
            set ax2 to 0.087155
            set ay2 to 0.996194
        elseif ( angle_temp == 5 )
            set ax1 to 0.087155
            set ay1 to 0.996194
            set ax2 to 0.104528
            set ay2 to 0.994521
        elseif ( angle_temp == 6 )
            set ax1 to 0.104528
            set ay1 to 0.994521
            set ax2 to 0.121869
            set ay2 to 0.992546
        elseif ( angle_temp == 7 )
            set ax1 to 0.121869
            set ay1 to 0.992546
            set ax2 to 0.139173
            set ay2 to 0.990268
        elseif ( angle_temp == 8 )
            set ax1 to 0.139173
            set ay1 to 0.990268
            set ax2 to 0.156434
            set ay2 to 0.987688
        elseif ( angle_temp == 9 )
            set ax1 to 0.156434
            set ay1 to 0.987688
            set ax2 to 0.173648
            set ay2 to 0.984807
        elseif ( angle_temp == 10 )
            set ax1 to 0.173648
            set ay1 to 0.984807
            set ax2 to 0.190808
            set ay2 to 0.981627
        endif
elseif ( angle_temp <= 20 ) ; etc.....
if ( angle == 11 )
set ax1 to 0.190808
set ay1 to 0.981627
set ax2 to 0.207911
set ay2 to 0.978147
elseif ( angle_temp == 12 )
set ax1 to 0.207911
set ay1 to 0.978147
set ax2 to 0.224951
set ay2 to 0.974370
elseif ( angle_temp == 13 )
set ax1 to 0.224951
set ay1 to 0.974370
set ax2 to 0.241921
set ay2 to 0.970295
elseif ( angle_temp == 14 )
set ax1 to 0.241921
set ay1 to 0.970295
set ax2 to 0.258819
set ay2 to 0.965925
elseif ( angle_temp == 15 )
set ax1 to 0.258819
set ay1 to 0.965925
set ax2 to 0.275637
set ay2 to 0.961261
elseif ( angle_temp == 16 )
set ax1 to 0.275637
set ay1 to 0.961261
set ax2 to 0.292371
set ay2 to 0.956304
elseif ( angle_temp == 17 )
set ax1 to 0.292371
set ay1 to 0.956304
```

```
set ax2 to 0.309016
set ay2 to 0.951056
elseif ( angle_temp == 18 )
set ax1 to 0.309016
set ay1 to 0.951056
set ax2 to 0.325568
set ay2 to 0.945518
elseif ( angle_temp == 19 )
set ax1 to 0.325568
set ay1 to 0.945518
set ax2 to 0.342020
set ay2 to 0.939692
elseif ( angle_temp == 20 )
set ax1 to 0.342020
set ay1 to 0.939692
set ax2 to 0.358367
set ay2 to 0.933580
endif
elseif ( angle_temp <= 30 )
if ( angle_temp == 21 )
set ax1 to 0.358367
set ay1 to 0.933580
set ax2 to 0.374606
set ay2 to 0.927183
elseif ( angle_temp == 22 )
set ax1 to 0.374606
set ay1 to 0.927183
set ax2 to 0.390731
set ay2 to 0.920504
elseif ( angle_temp == 23 )
set ax1 to 0.390731
set ay1 to 0.920504
set ax2 to 0.406736
set ay2 to 0.913545
elseif ( angle_temp == 24 )
set ax1 to 0.406736
set ay1 to 0.913545
set ax2 to 0.422618
set ay2 to 0.906307
elseif ( angle_temp == 25 )
set ax1 to 0.422618
set ay1 to 0.906307
set ax2 to 0.438371
set ay2 to 0.898794
elseif ( angle_temp == 26 )
set ax1 to 0.438371
set ay1 to 0.898794
set ax2 to 0.453990
set ay2 to 0.891006
elseif ( angle_temp == 27 )
set ax1 to 0.453990
set ay1 to 0.891006
set ax2 to 0.469471
set ay2 to 0.882947
elseif ( angle_temp == 28 )
set ax1 to 0.469471
set ay1 to 0.882947
set ax2 to 0.484809
set ay2 to 0.874619
elseif ( angle_temp == 29 )
set ax1 to 0.484809
set ay1 to 0.874619
set ax2 to 0.500000
set ay2 to 0.866025
elseif ( angle_temp == 30 )
set ax1 to 0.500000
set ay1 to 0.866025
set ax2 to 0.515038
set ay2 to 0.857167
endif
elseif ( angle_temp <= 40 )
if ( angle_temp == 31 )
set ax1 to 0.515038
set ay1 to 0.857167
set ax2 to 0.529919
set ay2 to 0.848048
elseif ( angle_temp == 32 )
```

```
set ax1 to 0.529919
set ay1 to 0.848048
set ax2 to 0.544639
set ay2 to 0.838670
elseif ( angle_temp == 33 )
set ax1 to 0.544639
set ay1 to 0.838670
set ax2 to 0.559192
set ay2 to 0.829037
elseif ( angle_temp == 34 )
set ax1 to 0.559192
set ay1 to 0.829037
set ax2 to 0.573576
set ay2 to 0.819152
elseif ( angle_temp == 35 )
set ax1 to 0.573576
set ay1 to 0.819152
set ax2 to 0.587785
set ay2 to 0.809016
elseif ( angle_temp == 36 )
set ax1 to 0.587785
set ay1 to 0.809016
set ax2 to 0.601815
set ay2 to 0.798635
elseif ( angle_temp == 37 )
set ax1 to 0.601815
set ay1 to 0.798635
set ax2 to 0.615661
set ay2 to 0.788010
elseif ( angle_temp == 38 )
set ax1 to 0.615661
set ay1 to 0.788010
set ax2 to 0.629320
set ay2 to 0.777145
elseif ( angle_temp == 39 )
set ax1 to 0.629320
set ay1 to 0.777145
set ax2 to 0.642787
set ay2 to 0.766044
elseif ( angle_temp == 40 )
set ax1 to 0.642787
set ay1 to 0.766044
set ax2 to 0.656059
set ay2 to 0.754709
endif
elseif ( angle_temp <= 50 )
if ( angle_temp == 41 )
set ax1 to 0.656059
set ay1 to 0.754709
set ax2 to 0.669130
set ay2 to 0.743144
elseif ( angle_temp == 42 )
set ax1 to 0.669130
set ay1 to 0.743144
set ax2 to 0.681998
set ay2 to 0.731353
elseif ( angle_temp == 43 )
set ax1 to 0.681998
set ay1 to 0.731353
set ax2 to 0.694658
set ay2 to 0.719339
elseif ( angle_temp == 44 )
set ax1 to 0.694658
set ay1 to 0.719339
set ax2 to 0.707106
set ay2 to 0.707106
elseif ( angle_temp == 45 )
set ax1 to 0.707106
set ay1 to 0.707106
set ax2 to 0.719339
set ay2 to 0.694658
elseif ( angle_temp == 46 )
set ax1 to 0.719339
set ay1 to 0.694658
set ax2 to 0.731353
set ay2 to 0.681998
elseif ( angle_temp == 47 )
```

```
set ax1 to 0.731353
set ay1 to 0.681998
set ax2 to 0.743144
set ay2 to 0.669130
elseif ( angle_temp == 48 )
set ax1 to 0.743144
set ay1 to 0.669130
set ax2 to 0.754709
set ay2 to 0.656059
elseif ( angle_temp == 49 )
set ax1 to 0.754709
set ay1 to 0.656059
set ax2 to 0.766044
set ay2 to 0.642787
elseif ( angle_temp == 50 )
set ax1 to 0.766044
set ay1 to 0.642787
set ax2 to 0.777145
set ay2 to 0.629320
endif
elseif ( angle_temp <= 60 )
if ( angle == 51 )
set ax1 to 0.777145
set ay1 to 0.629320
set ax2 to 0.788010
set ay2 to 0.615661
elseif ( angle_temp == 52 )
set ax1 to 0.788010
set ay1 to 0.615661
set ax2 to 0.798635
set ay2 to 0.601815
elseif ( angle_temp == 53 )
set ax1 to 0.798635
set ay1 to 0.601815
set ax2 to 0.809016
set ay2 to 0.587785
elseif ( angle_temp == 54 )
set ax1 to 0.809016
set ay1 to 0.587785
set ax2 to 0.819152
set ay2 to 0.573576
elseif ( angle_temp == 55 )
set ax1 to 0.819152
set ay1 to 0.573576
set ax2 to 0.829037
set ay2 to 0.559192
elseif ( angle_temp == 56 )
set ax1 to 0.829037
set ay1 to 0.559192
set ax2 to 0.838670
set ay2 to 0.544639
elseif ( angle_temp == 57 )
set ax1 to 0.838670
set ay1 to 0.544639
set ax2 to 0.848048
set ay2 to 0.529919
elseif ( angle_temp == 58 )
set ax1 to 0.848048
set ay1 to 0.529919
set ax2 to 0.857167
set ay2 to 0.515038
elseif ( angle_temp == 59 )
set ax1 to 0.857167
set ay1 to 0.515038
set ax2 to 0.866025
set ay2 to 0.500000
elseif ( angle_temp == 60 )
set ax1 to 0.866025
set ay1 to 0.500000
set ax2 to 0.874619
set ay2 to 0.484809
endif
elseif ( angle_temp <= 70 )
if ( angle_temp == 61 )
set ax1 to 0.874619
set ay1 to 0.484809
set ax2 to 0.882947
```

```
set ay2 to 0.469471
elseif ( angle_temp == 62 )
set ax1 to 0.882947
set ay1 to 0.469471
set ax2 to 0.891006
set ay2 to 0.453990
elseif ( angle_temp == 63 )
set ax1 to 0.891006
set ay1 to 0.453990
set ax2 to 0.898794
set ay2 to 0.438371
elseif ( angle_temp == 64 )
set ax1 to 0.898794
set ay1 to 0.438371
set ax2 to 0.906307
set ay2 to 0.422618
elseif ( angle_temp == 65 )
set ax1 to 0.906307
set ay1 to 0.422618
set ax2 to 0.913545
set ay2 to 0.406736
elseif ( angle_temp == 66 )
set ax1 to 0.913545
set ay1 to 0.406736
set ax2 to 0.920504
set ay2 to 0.390731
elseif ( angle_temp == 67 )
set ax1 to 0.920504
set ay1 to 0.390731
set ax2 to 0.927183
set ay2 to 0.374606
elseif ( angle_temp == 68 )
set ax1 to 0.927183
set ay1 to 0.374606
set ax2 to 0.933580
set ay2 to 0.358367
elseif ( angle_temp == 69 )
set ax1 to 0.933580
set ay1 to 0.358367
set ax2 to 0.939692
set ay2 to 0.342020
elseif ( angle_temp == 70 )
set ax1 to 0.939692
set ay1 to 0.342020
set ax2 to 0.945518
set ay2 to 0.325568
endif
elseif ( angle_temp <= 80 )
if ( angle_temp == 71 )
set ax1 to 0.945518
set ay1 to 0.325568
set ax2 to 0.951056
set ay2 to 0.309016
elseif ( angle_temp == 72 )
set ax1 to 0.951056
set ay1 to 0.309016
set ax2 to 0.956304
set ay2 to 0.292371
elseif ( angle_temp == 73 )
set ax1 to 0.956304
set ay1 to 0.292371
set ax2 to 0.961261
set ay2 to 0.275637
elseif ( angle_temp == 74 )
set ax1 to 0.961261
set ay1 to 0.275637
set ax2 to 0.965925
set ay2 to 0.258819
elseif ( angle_temp == 75 )
set ax1 to 0.965925
set ay1 to 0.258819
set ax2 to 0.970295
set ay2 to 0.241921
elseif ( angle_temp == 76 )
set ax1 to 0.970295
set ay1 to 0.241921
set ax2 to 0.974370
```

```
set ay2 to 0.224951
elseif ( angle_temp == 77 )
set ax1 to 0.974370
set ay1 to 0.224951
set ax2 to 0.978147
set ay2 to 0.207911
elseif ( angle_temp == 78 )
set ax1 to 0.978147
set ay1 to 0.207911
set ax2 to 0.981627
set ay2 to 0.190808
elseif ( angle_temp == 79 )
set ax1 to 0.981627
set ay1 to 0.190808
set ax2 to 0.984807
set ay2 to 0.173648
elseif ( angle_temp == 80 )
set ax1 to 0.984807
set ay1 to 0.173648
set ax2 to 0.987688
set ay2 to 0.156434
endif
elseif ( angle_temp <= 90 )
if ( angle_temp == 81 )
set ax1 to 0.987688
set ay1 to 0.156434
set ax2 to 0.990268
set ay2 to 0.139173
elseif ( angle_temp == 82 )
set ax1 to 0.990268
set ay1 to 0.139173
set ax2 to 0.992546
set ay2 to 0.121869
elseif ( angle_temp == 83 )
set ax1 to 0.992546
set ay1 to 0.121869
set ax2 to 0.994521
set ay2 to 0.104528
elseif ( angle_temp == 84 )
set ax1 to 0.994521
set ay1 to 0.104528
set ax2 to 0.996194
set ay2 to 0.087155
elseif ( angle_temp == 85 )
set ax1 to 0.996194
set ay1 to 0.087155
set ax2 to 0.997564
set ay2 to 0.069756
elseif ( angle_temp == 86 )
set ax1 to 0.997564
set ay1 to 0.069756
set ax2 to 0.998629
set ay2 to 0.052335
elseif ( angle_temp == 87 )
set ax1 to 0.998629
set ay1 to 0.052335
set ax2 to 0.999390
set ay2 to 0.034899
elseif ( angle_temp == 88 )
set ax1 to 0.999390
set ay1 to 0.034899
set ax2 to 0.999847
set ay2 to 0.017452
elseif ( angle_temp == 89 )
set ax1 to 0.999847
set ay1 to 0.017452
set ax2 to 1.000000
set ay2 to 0.000000
elseif ( angle_temp == 90 )
set ax1 to 1.000000
set ay1 to 0.000000
set ax2 to 0.999847
set ay2 to 0.017452
endif
endif

;on utilise l'extrapolation linéaire simple pour approximer la valeur décimale de sinus et cosinus :
```

```
set angconvx1 to ( ax2 - ax1 ) ; calcule la distance entre sin(angle) et sin(angle+1)
set angconvy1 to ( ay2 - ay1 ) ; idem pour  cos
set angconvx2 to ( angconvx1 * angledec ) ; calcule la fraction par le  reste décimal (calculate a fraction by the decimal residue)
set angconvy2 to ( angconvy1 * angledec ) ; idem pour cosinus
set axfinal to ( ax1 + angconvx2 ) ; résultat final égale résultatt+ fraction
set ayfinal to ( ay1 + angconvy2 ) ; résultat final égale résultat + fraction

;correction pour le  quadrant:
if ( anglequad == 1 )
	set axfinal to ( axfinal * 1 ) ; stocke le sinus final
	set ayfinal to ( ayfinal * 1 ) ; stocke le cosinus final
elseif ( anglequad == 2 )
	set axfinal to ( axfinal * 1 ) ; stocke le sinus final
	set ayfinal to ( ayfinal * -1 ) ; stocke le cosinus final
elseif ( anglequad == 3 )
	set axfinal to ( axfinal * -1 ) ; stocke le sinus final
	set ayfinal to ( ayfinal * -1 ) ; stocke le cosinus final
elseif ( anglequad == 4 )
	set axfinal to ( axfinal * -1 ) ; stocke le sinus final
	set ayfinal to ( ayfinal * 1 ) ; stocke le cosinus final
endif

endwhile ; fin de la boucle pour les 3 angles

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
end jdtrigscript
```

Mannequins

Ils sont très populaires car c'est un bon moyen d'exposer les collections d'armures – ils sont présents dans beaucoup de mods rajoutant des maisons – ce script montre comment s'y prendre (merci beaucoup à Stephen Kent c'est-à-dire Riiak Shi Nal pour avoir partagé ce script). C*et exemple est un script de"prochaine génération" qui utilise les fonctions de Tribunal pour empêcher que le PJ ne déplace le mannequin s'il est équipé d'armes* (Riiak n'a pas encore trouvé le moyen pour qu'un mannequin tienne une arme à la main) *et/ou de pièces d'armures. Dupliquez votre script pour tenir compte des versions féminines et masculines des mannequins. Les changements apportées n'empêche pas le joueur de déplacer le mannequin s'il possède des objets divers ; ces objets seront irrémédiablement perdus.*

J'ai ajouté quelques commentaires supplémentaires en plus de ceux de Riiak. Le Mannequin est en réalité un PNJ normal avec 0 de santé (positionnée dans le TESCS). Dans cette version, vous donnez des objets au mannequin en l'activant et il s'équipera des pièces d'armures.

```
Begin rsn_mannequin_f_script

short button          ;pour récupérer la réponse du joueur
short questionState   ;pour savoir dans quel état de la question on se trouve
short nEquipType      ;quel type d'objet est équipé ?
short nStillEquipped  ;armure équipée ?
float fDeleteTimer    ;timer pour supprimer l'objet

SkipAnim ;GBG: essentiel, cela force le mannequin, qui est un PNJ, à rester immobile

if ( menumode == 1 )
       return
endif

if ( GetDisabled == 1 )
; si le mannequin a été désactivé, on attend encore un peu avant d'effacer cette référence
       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 )
  ; section divisée en deux parties à cause de la limitation du nombre de if imbriqués
  ; du langage.
  Set nStillEquipped to 0
  ; ici on vérifie qu'une arme est équipée (à mettre surtout si quelqu'un découvre comment
  ; faire pour qu'un PNJ exhibe une arme)
  Set nEquipType to ( GetWeaponType )
  if ( nEquipType == -1 )
    ; ici on vérifie qu'une pièce d'armure est équipé (NOTE: on a 11 pièces d'armures
    ; différentes ; il faut donc tester pour chacune d'elle séparément)
    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: placée à 1 si on a toujours certaines pièces
d'armures équipées
              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
  else
    Set nStillEquipped to 1
  endif

if ( nStillEquipped != 1 ) ;on n'avance que si on n'a pas trouvé d'armure équipée pour les 6
                           ; types précédents
  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 )
            ;n'affiche cette question que si aucune arme ou armure n'est équipée.
            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
  else
    Set nStillEquipped to 1
  endif
endif

; Etape suivante de l'exécution : on attend le choix de l'utilisateur ou on active tout de
suite
set questionState to 20
endif

if ( questionState == 20 )
if ( nStillEquipped != 1 )
set button to GetButtonPressed
else
; le Mannequin a toujours des armes ou des armures équipées, mais l'utilisateur veut l'activer
; donc on le prévient au lieu de ramasser directement l'objet
              MessageBox "You haven't removed your equipment."
              Set button to 1 ;indique qu'il y a toujours des objets sur le Mannequin
       endif
```

```
if ( button == 0 )
                set questionState to 0
; désactive le mannequin courant et en crée un nouveau sans que l'on ait à s'inquiéter des
; pertes d'objets
; GBG: si on transporte beaucoup les Mannequins
; la fonction SetDelete peut être une bonne idée
                Disable
; c'est l'objet qui contient ce script qui génère un nouveau mannequin lorsqu'on le sort de
; l'inventaire.
; GBG: pour un mannequin "femme" cet objet ne serait pas le même
; ajout dans l'inventaire de l'objet : cet objet possède 1 script attaché pour recréer le
; mannequin quand on le sort de l'inventaire
player->addItem, "_rsn_man_f_holder", 1
playSound "Item Misc Up"
elseif ( button == 1 )
; il y a toujours des objets sur le mannequin (après vérification ou suite à la réponse de
l'utilisateur)
set questionState to 0
Activate
endif
endif

end
```

Le script suivant va sur l'objet ajouté à votre inventaire lorsqu'on déplace le mannequin. Lorsque que vous le sortez de l'inventaire, un nouveau mannequin apparaît à vos pieds, d'où la nécessité de récupérer toutes les pièces d'armures. Sans cela, vous les perdriez tous :

```
; prévoir l'équivalent de ce script pour gérer le mannequin de sexe opposé.

Begin rsn_man_f_holder_script

short OnPCDrop
float fDeleteTimer

if ( GetDisabled == 1 )
; si l'objet a été désactivé, on attend encore un peu avant d'effacer cette référence
        Set fDeleteTimer to ( fDeleteTimer + GetSecondsPassed )
        if ( fDeleteTimer > 5 )
                SetDelete 1
        endif
        return
endif

if ( OnPCDrop == 1 )
        Disable
        ; voici le PNJ avec 0 points de santé : il s'agit vraiment d'un corps sans vie
        PlaceAtPC, "_rsn_mannequin_female", 1, 0, 0
        Set OnPCDrop to 0
endif
End
```

Serait-elle en train de me regarder ?

Voici un superbe script d' Horatio qui permet de détecter si un Acteur regarde le joueur :

```
Begin PCLookAtMe

float fPCX     ;coordonées du PJ
float fPCY
float fPCAngle
float fdx      ;distance entre l'Acteur et le PJ
float fdy
float fRatio

short sPCLookAtMe

set sPCLookAtMe to 1

; vous pouvez aussi ajouter une vérification avec GetLOS ici
; cependant comme je n'ai jamais réussi à faire fonctionner un GetLOS correctement, je n'ai
pas insisté

;si le PJ est vraiment loin
if ( GetDistance, Player > 8000 )
        set sPCLookAtMe to 0
else

;de la trigonométrie, et oui !
;calcule la direction relative du PJ par rapport à l'Acteur
;n'utilise que des 'parties' de 45 degrés


        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

if ( sPCLookAtMe == 0 )
;faire quelque chose pendant que le joueur ne regarde pas
endif

End
```

Séquence cinématique

Voici une approche très intelligente de gianluca (Morrowind Summit forums) pour réaliser une séquence cinématique avec le moteur du jeu. On enlève les contrôles du joueur, on le place sur un objet invisible "*CollisionWall*" puis on le bouge, lui ainsi que la caméra. On ne peut pas avoir de séquences cinématiques impliquant le joueur mais cela reste tout de même magnifique.

```
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
```

Résolution des problèmes (*Troubleshooting*)

Conseils généraux

- Un bon moyen de débugger est d'insérer des *MessageBox* aux points clefs de votre script.
- Si vous obtenez des messages d'erreurs et que vous n'en trouvez pas la cause, essayez de supprimer les lignes suspectes du code une à une (utilisez ; pour les mettre en commentaires) pour trouver précisément la ligne causant l'erreur.
- Soyez attentifs aux codes d'erreurs que l'éditeur et le jeu vous transmettent : ils permettent d'identifier la source du problème jusqu'à un certain point.

La Console

Utilisation de la Console pour vérifier les variables:

Pendant le jeu, vous pouvez utiliser la console pour vérifier l'état de vos variables. Appelez la console (la touche normale est ² ou n'importe quelle touche située à gauche du "1" si vous utilisez un clavier standard) et tapez "sv" – vous obtiendrez la liste des variables globales avec leur valeur. Maintenant trouvez-vous un objet possédant un script attaché. Rappelez la console, cliquez-gauche sur l'objet – le titre de la fenêtre changera. Tapez "sv" à nouveau – maintenant, ce sont les variables locales du script qui seront listées. Pour vérifier les variables globales qui ne sont pas listées par "sv", utilisez la commande "show *nom_de_la_variable*".

Utilisation de la Console pour tester rapidement les scripts:

La console peut vous aider à tester vos scripts : pour les objets que vous n'avez pas besoin de placer dans l'éditeur, il vous suffit de l'utiliser. Ecrivez juste l'ID de votre objet sur un morceau de papier, chargez n'importe quelle sauvegarde puis tapez dans la console

```
PlaceatPC "Mon_Objet" 1,1,1
```

L'objet apparaîtra alors à vos pieds.

```
Player -> AddItem "Mon_Objet", 1
```

L'objet apparaîtra dans votre inventaire.

Pour aller dans un lieu spécifique, utilisez

```
coc "nom_de_la_cellule "
```

pour les cellules intérieures ou

```
coe -1,-7
```

pour les cellules extérieures (écrivez les coordonnées de la cellule telles qu'elles apparaissent dans l'éditeur) ; *coc* fonctionne aussi pour les cellules extérieures mais comme certaines d'entre elles ne possèdent pas un nom unique, vous pourriez atterrir ailleurs. Cela peut tout de même être utile, par exemple `coc balmora` vous déposera quelque part dans Balmora.

```
tcl
```

Toggles collision – vous permet de traverser les murs et donc d'atteindre facilement des lieux innaccessibles.

```
Tgm
```

Toggle God mode – plus besoin de s'inquiéter des gros monstres !

Une autre commande inestimable pour faire basculer le jeu en mode "debug" et qui permet de visualiser les nombres se rapportant aux effets des sorts, des chances de frappe, etc...(merci Wakim) :

1) pressez la touche "²" pour appeler la console.

2) tapez "tcs" dans la console et tapez entrée.

3) cliquez (droite ou gauche, je ne me souviens plus) n'importe où sur l'écran en dehors de la fenêtre de la console afin de continuer à jouer tout en ayant la console affichée et active.

Et voilà. Il y a d'autres variantes pour la commande "tcs", comme (et ce n'est pas exhaustif) "tks" et "tms" qui correspondent à "toggle xxxxxx statistics" où xxxxx est : c = combat, m = magic, k = kill.

Messages d'erreurs, mauvais fonctionnements et causes habituelles

Dans le jeu, lorsque le script s'exécute :

Dans l'éditeur

L'éditeur indique généralement la ligne à laquelle il a rencontré un problème ; les erreurs reportées ici sont donc souvent faciles à résoudre.

"Mismatched If/else/endif starting on line..."
une ou plusieurs conditions n'ont pas été 'fermées'. Utilisez les tabulations pour trouver plus facilement l'oubli. Cela arrive aussi lorsque des noms incorrects sont utilisés pour les variables de fonctions ou des erreurs de frappe.

"Function reference object "Foobject" not found"
Cette erreur indique que l'objet en question, par exemple

```
Foobject -> GetDistance Player
```

N'existe pas. Lorsque vous écrivez un script puis que vous créez un objet pour qu'il soit référencé, vous obtiendrez aussi cette erreur. Fermez la fenêtre de l'éditeur puis rouvrez-la pour enregistrer l'objet avec le compilateur.

"Could not find variable or function "Foobject""
Une autre erreur de syntaxe, indiquant que vous avez utilisé une variable, fonction indéfinie ou un objet qui n'existe pas.

"Script command "foofunc" not found on line 3"
Une commande/fonction n'a pas été reconnue par le compilateur. Sans doute une faute de frappe.

"You need to end a script with script End scriptname"
Indique que le compilateur n'a pas trouvé la commande *End*. Cette erreur peut aussi se produire lorsqu'une erreur précédente interrompt le processus de compilation.

"Syntax error Line 20. Miss matched paranthesis"
Indique que vous n'avez pas fermé toutes les parenthèses ou au contraire que vous en avez fermé de trop.

"You need to enter a value on line 20"
Vous n'avez pas fourni assez d'arguments à une certaine fonction.

Messages d'erreurs in game:

"EXPRESSION in script..."
Suivi par

"RightEval ..."
Cette erreur indique que des variables n'ont pas été déclarées. Cela arrive souvent lorsque des variables du jeu sont quasiment utilisées comme des fonctions, par exemple *PCEquip*.

En général, EXPRESSION apparaît lorsque des variables n'ont pas été déclarées dans le script.

"LeftEval"
Cette erreur semble apparaître lorsque vous avez accidentellement déclaré une fonction comme une variable. Par exemple, les lignes suivantes produiront cette erreur :

```
short ScriptRunning
if (ScriptRunning "MonScript" == 1 )
```

Chacune de ces deux erreurs peut être causée par l'absence de caractères blancs aux endroits appropriés. Laissez toujours un espace entre une parenthèse et un appel de fonction, des variables, etc...
If (OnActivate == 1), et pas *If (OnActivate==1)*. Cela cause rarement des erreurs, mais ça peut arriver, croyez-moi.

"Infix to Postfix" error
Indique généralement une mauvaise syntaxe, concernant la commande *set* et la notation pointée:

```
set une_variable to ID_Acteur->GetHealth
```

devra plutôt être écrit :

```
set une_variable to ( ID_Acteur->GetHealth )
```

Une autre cause peut être un oubli de déclaration, variable ou variable-fonction, comme *OnPCEquip*, etc... (Merci Horatio et Ragnar_GD)

La commande AITravel ne fonctionne pas

Une cause habituelle est les coordonnées qui sont par erreur placées trop loin (par exemple en ayant oublié un "-" ou en ayant un chiffre de trop). Une autre très désagréable : deux "-" côte à côte qui ne semblait en faire qu'un, à peine plus long qu'un seul "-" dans l'éditeur.

Dédoublement de PNJs

Les dédoublements de PNJs ou d'autres objets semblent se produire de temps en temps lorsqu'une nouvelle version d'un plugin en écrase une ancienne et que les informations de la sauvegarde sont aussi gardées : un clone du PNJ est créé, même si vous n'avez modifié que son script. Essayez de charger le plugin dans une cellule située loin du PNJ. On peut éviter ce problème en gardant le nom de fichier du mod identique ; apparemment les sauvegardes enregistrent le nom du fichier dans la même partie que les PNJs (merci Raptormeat). Pour plus d'informations sur les dédoublements et comment les éviter, consultez l'excellente page de Shadowsong : http://www.angelfire.com/clone2/shadowsong/index.html

Crash sur le bureau à l'exécution d'un script

Il y a malheureusement un grand nombre de causes possibles. On peut en relier beaucoup avec l'absence de conditions "do once" (par exemple, en appelant certaines fonctions à chaque frame). D'autres problèmes connus : supprimer un objet avec son propre script alors que celui-ci est train de s'exécuter. Utiliser *Equip* sur des potions (corrigée avec Tribunal). Lancer des sorts ciblés à partir d'un activateur (corrigé avec Tribunal). Utiliser *AIActivate* sur des portes qui téléportent en dehors de la cellule active. Essayer d'utiliser *PlaceItem* avec le

même identifiant que l'objet sur lequel s'exécute le script. Utiliser *SetDelete* sur un objet qui ne peut pas être désactivé (*a non-disabled object*).

Crash sur le bureau (Crash To Desktop) au chargement d'un plugin

L'une des raisons rapportées sont les calculs extrêmes longs : il semble qu'il y ait un problème avec les très longues additions (par exemple en ajoutant plus de 20 variables en une seule ligne) qui provoque un CTD au chargement. Si cela devait arriver, séparez vos calculs via des résultats intermédiaires sur plusieurs lignes.

Appendice

Nouvelles fonctions apportées par TRIBUNAL

L'extension Tribunal a introduit un certain nombre de nouvelles fonctions, en a étendu ou corrigé d'anciennes. Notez que le créateur du mod ET son utilisateur doivent avoir installé Tribunal (ou Bloodmoon, ou la GOTY) pour pouvoir s'en servir ; marquez donc votre mod en conséquence. Les fonctions sont maintenant traitées avec le document principal, mais cette liste permet de voir facilement quelles sont les fonctions de Tribunal.
(merci beaucoup à Mike Lipari de Bethsoft pour avoir partagé ceci avec la communauté)

Modifications/Corrections aux scripts de Morrowind :

- *SetPos* accepte maintenant les variables (float) en arguments.
- *SetAngle* accepte maintenant les variables (float) en arguments.
- *Equip* fonctionne maintenant comme prévu et force les PNJs à s'équiper d'armure ou de vêtements.
- *AIActivate* a apparemment été corrigée

Index des nouvelles fonctions de Tribunal :

AddToLevCreature
AddToLevItem
ClearForceJump
ClearForceMoveJump
ClearForceRun
DaysPassed (variable)
DisableLevitation
EnableLevitation
ExplodeSpell
ForceJump
ForceMoveJump
ForceRun
GetArmorType
GetCollidingActor
GetCollidingPC
GetForceJump
GetForceMoveJump
GetForceRun
GetPCJumping
GetPCRunning
GetPCSneaking
GetScale
GetSpellReadied
GetSquareRoot
GetWaterLevel
GetWeaponDrawn
GetWeaponType
HasItemEquipped
HurtCollidingActor
ModScale
ModWaterLevel
PlaceItem
RemoveFromLevCreature
RemoveFromLevItem
SetDelete
SetJournalIndex (?)
SetScale
SetWaterLevel

Nouvelles fonctions apportées par BLOODMOON

 La seconde extension de Morrowind, Bloodmoon, introduit quelques fonctions. Notez que le créateur du mod ET son utilisateur doivent avoir installé Bloodmoon pour pouvoir s'en servir ; marquez donc votre mod en conséquence.

Index des nouvelles fonctions et variables de Bloodmoon :

AllowWereWolfForceGreeting
BecomeWerewolf
GetPCInJail
GetPCTraveling
GetWerewolfKills
IsWerewolf
PCKnownWerewolf PlaceAtMe
SetWerewolfAcrobatics
TurnMoonRed
TurnMoonWhite
UndoWerewolf

Fonctions non documentées

Ces fonctions n'étaient pas documentées dans le fichier d'aide original. J'en ai expliqué la majorité dans ce guide, mais il y en a toujours quelques-unes que je ne connais pas. J'ai pensé que cela pourrait utile d'en avoir une liste. La liste a été construite par quelqu'un qui a ouvert le TES-CS .exe avec un éditeur hexadécimal, il est donc probable qu'elle soit complète.
(merci à Soralis pour la liste et XPCagey et aux autres qui ont aidé à sa construction) :

Non documentée dans le fichier d'aide :

```
PayFineThief
EnableStatReviewMenu
GetFactionReaction
ShowMap
EnableBirthMenu
EnableClassMenu
EnableRaceMenu
EnableNameMenu
RemoveEffects
EnableMagicMenu
EnableMapMenu
EnableInventoryMenu
EnableStatsMenu
GetInterior
GetLineOfSight (alias for GetLOS?)
GetWindSpeed

GetCurrentTime
ResetActors
OutputRefInfo
MenuTest
```

Fonctions qui sont **écrites différemment que celles de la documentation (XPCagey):**

```
getHealthRatio -> getHealthGetRatio
getInvisible -> getInvisibile
setInvisible -> setInvisibile
modInvisible -> modInvisibile
getSecundusPhase -> getSecundaPhase
```

(syntaxe actuelle à droite)

Et celles qui ne marchent pas :

```
getPlayerViewSwitch -> (ne peut être
utilisée)
OnRepair -> absente dans les tables de
chaines de caractères (string tables)
UsedOnMe -> absente dans les tables de
chaines de caractères (string tables)
name -> absente dans les tables de chaines
de caractères (string tables)
```

Sachez aussi que la plupart des commandes console peuvent être compilées comme des fonctions par le Construction Set (voir la liste ci-dessous). Beaucoup ne sont pas listées dans le fichier d'aide.

Fonctions typées variables (Variable-type functions) :

Pour une consultation facile, la liste suivante vous montre toutes les fonctions/variables de type hybride que vous **devez déclarer** comme des variables si vous souhaitez les utiliser dans un script.

Variables locales positionnées par le jeu :

```
Short OnPCEquip
Short OnPCAdd
Short OnPCRepair
Short OnPCSoulGemUse
Short OnPCHitMe
Float minimumProfit          (Tribunal)
```

Variables locales que vous pouvez positionner comme un drapeau/flag:

```
Short Companion              (Tribunal)
Short PCSkipEquip
```

Globales spéciales

Certaines variables globales ont une signification particulière dont vous pouvez tirer parti dans votre propre script. Comme elles sont globales, pas besoin de les déclarer.

`Short NPCVoiceDistance (750)`	Utilisé comme une distance lorsqu'un PNJ qui vous suit vous appelle afin que vous l'attendiez (voir par exemple DandsaScript)
`Float GameHour`	Contient l'heure courante de la journée (0-23)
`Short Day`	Contient le jour courant du mois (1-30)
`Short Month`	Contient le mois courant de l'année (0-11)
`Short Year (427)`	Contient l'année
`Float TimeScale (30)`	Positionne le ratio temps réel/temps de jeu
`Short Random100`	Contient un nombre tiré au hasard entre 0 et 100 (positionné par le script principal)
`Short PCRace`	Contient la race du joueur (1=Argonien, 2=Bréton, 3=Elfe Noir, 4=Aldmer, 5=Impérial, 6= Khajiit, 7=Nord, 8=Orc, 9=Rougegarde, 10=Bosmer)
`Short PCVampire`	Statut vampirique : 0=Normal, 1=Vampire, -1= guéri
`Short VampClan`	Si le PJ devient un vampire, cela indique son clan. 1=Aundae, 2=Berne, 3=Quarra
`Short DaysPassed`	Tribunal: Contient le nombre de jours passés depuis le début du jeu. Rapportée buggée dans Bloodmoon (JOG) (Forum info / JOG).
`Short PCWerewolf`	Statut lycanthrope: 0=Normal, 1=Loup-Garou, -1=Guéri
`Float WerewolfClawMult (25.00)`	Augmenté pendant les quêtes des Loup-Garous afin que vos griffes fassent plus de dégâts.
`Short PCHasGoldDiscount`	Dialogues. Passe à 1 si le joueur a assez d'or pour payer le rabais offert par la guilde des Voleurs sur votre "tête mise à prix ".
`Short PCHAsTurnIn`	Dialogues. Passe à 1 si le joueur a assez d'or pour payer l'amende réduite lorsqu'il se rend.
`Short PCHasCrimeGold`	Dialogues. Passe à 1 si le joueur a assez d'argent pour payer l'amende compensatoire.
`Short CrimeGoldTurnIn`	Or nécessaire pour payer l'amende réduite
`CrimeGoldDiscount`	Or nécessaire pour payer le rabais de la guilde des Voleurs

Unités du jeu :

Système de mesure US :

1 unité de jeu = 0.56 inches/pouces
50 =28 inches/pouces
500 = 23.3 feet/pieds
5000 = 233.3 feet/pieds
8192 = 385 feet/pieds = 1 cellule du jeu

Système métrique :

1 unité de jeu = 1.42 cm
100 unités de jeu = 142 cm = 1.42 mètres
1000 unités de jeu = 14.2 mètres
8192 unités de jeu = 116.33 mètres = 1 cellule extérieure

L'île de Morrowind s'étend sur 5.00 km du nord au sud et sur 4.65 km d'ouest en est.

(merci à Iudas pour ces informations)

Liste des effets magiques

Pour la fonction *GetEffect*, vous aurez besoin de l'ID de la chaîne de caractère elle-même (par exemple *GetEffect sEffectWaterBreathing*). Pour la fonction *RemoveEffects function*, c'est le nombre correspondant à l'effet qui est nécessaire (par exemple *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
```

Liste des commandes utilisables en mode console

(traduction par Aqualonne)

La plupart de ces commandes présentent un intérêt uniquement pour débuguer ou tester, mais certaines (repérées par un astérisque *) peuvent être utilisées dans les scripts. Pour les fonctions qui envoient une sortie à la console : vous pouvez continuer à jouer avec la console ouverte en faisant un clic droit en dehors de la fenêtre de console.

Command	short	Description
*CenterOnCell, "*Cell_ID"*	COC	Place le personnage dans la cellule demandée, particulièrement utile pour tester des mods.
*CenterOnExterior, *X, Y*	COE	Place le personnage dans la cellule extérieure demandée.
CreateMaps *"Filename.esp"*		Fabrique un fichier à l'image de la carte dépendant de la valeur de "Create Maps Enable" dans Morrowind.Ini ; à 1 (XBox), le fichier Filename.esp.map est créée dans le répertoire DataFiles avec les données la carte (dans un format inconnu). A 2 (ExteriorCellMaps), si vous avez créée un répertoire Maps à la racine de Morrowind, la commande fera un fichier bitmap de 256*256 avec un bon nombre de couleurs pour chaque cellule d'extérieur du jeu. La commande prend un bon moment même sur les ordinateurs puissants comme chaque cellule du jeu est chargée…
	BC	Abréviation de Beta Comment : permet d'éditer le fichier Morrowind.ini pour lui donner un nom de fichier dans la ligne du mode Beta Comment. Par exemple, Beta Comment File=BetaComment.txt Vous pouvez alors utiliser la B.C. Command pour mettre une note à propos d'un objet dans le jeu ; vous ouvrez la console et cliquez quelque part, puis tapez votre commentaire, par exemple BC "Corde mal attachée." Alors, dans BetaComment.txt vous aurez : 6/20/2004 (21:02) Morrowind.esm 5/8/2003 (21:07) Paul ex_t_root_03 Tel Vos (10,14) 85078 118468 4111 "Corde mal attachée." Au moment où la commande a été exécuté, les informations suivantes sont récupérées : le fichier dont vient l'objet (ici, la corde), l'heure de modification du fichier, le jeu de l'utilisateur (probablement l'identifiant dans windows), la cellule, les coordonnées spatiales de l'objet, et le commentaire.
FillJournal		Ajoute toutes les entrées au journal, ce qui prend un bon moment.
*FillMap		Montre toutes les villes (uniquement celles ayant un nom) sur la carte. Prend juste quelque seconde. Peu recommandé pour les scripts.
*FixMe		Fait passer le joueur à 128 unités de l'endroit où il se trouve actuellement. Pratique pour être « décollé » en cas de problèmes.
GetFactionReaction, *"factionID", "factionID", amount*		L'identifiant de la faction n'est pas optionnel ! Cela marche uniquement dans la console. L'issue de la commande est assez incertaine…
Help		Montre les commandes disponibles et leurs abréviations.
MoveOneToOne	MOTO	Change la vitesse de tous les personnages. Avec MOTO actif, vos vitesses de marche et de course sont les mêmes, et la même animation sera utilisée pour la marche ou la course.
ObjectReferenceInfo	ORI	Donne les informations sur l'objet demandé, par exemple la cellule où il se trouve et le fichier esm ou esp dont il est originaire. Pratique pour connaître les objets ajoutés par un mod.
OutputObjCounts		Compte tous les objets de catégories variées. Le résultat s'affiche dans la console.
OutputRefCounts		Compte toutes les références dans des catégories variées. Le résultat s'affiche dans la console.
	PT	Abréviation de Purge Textures ; le moteur du jeu recharge alors l'ensemble de ses textures. Si vous jouez en mode fenêtre, vous pouvez utiliser cette commande pour tester les textures dans le jeu en même temps que vous les travaillez avec un autre logiciel.
Show *global_var*		Affiche la valeur de la variable globale demandée dans la console.
ShowVars	SV	Montre les variables globales et les variables dans les scripts génériques, ou les variables locales si vous cliquez sur un objet avec un script local. Le résultat s'affiche dans la console.
StopCellTest	SCT	Arrête le test de cellule, le joueur reste dans la cellule actuellement chargée.
TestCells		Charge toutes les cellules dans l'ordre alphabétique ; ceci dit, il semble que cela fonctionne uniquement pour les intérieurs. Différence avec TestInteriorCells ?...
TestInteriorCells		Charge toutes les cellules d'intérieurs dans l'ordre alphabétique.

TestThreadCells		Charge toutes les cellules d'extérieures dans un ordre peut-être aléatoire…
TestModels	T3D	Fait le test de tous les objets et fichiers .nif
*ToggleAI	TA	Arrête l'intelligence artificielle, dans les combats. Pratique pour tester une cellule sans être attaqué par ses occupants !
ToggleBorders	TB	Montre la bordure des cellules extérieures.
ToggleCombatStats	TCS	Vous permet de surveiller les statistiques de combat en temps réel. En enclenchant ceci, vous pourrez jouer avec la fenêtre de console ouverte. A noter : en testant, la sortie a aussi été ajoutée dans le fichier log.txt ; il n'est pas dit que ce soit par défaut.
*ToggleCollision	TCL	Active ou désactive les collisions, vous permet de passer à travers les murs. Attention, les autres personnages peuvent se retrouver à planer ou coincés dans un mur. Conseil : utilisez le si vous êtes bloqués contre un élément du décor, et désactivez dès que l'élément a été franchi.
ToggleCollisionBoxes	TCB	Montre les boites de collisions pour tous les modèles.
ToggleCollisionGrid	TCG	Affiche la matrice qui indique probablement l'état actuel de la collision ; le jeu est assez ralenti…
ToggleDebugText	TDT	Affiche quelques informations de débuggage à l'écran : les groupes d'animations du joueur, la vitesse, la direction, la position, le FPS, le nombre de mouvements par frame, et quelques compléments.
ToggleDialogueStats	TDS	Affiche le résultat des tentatives de persuasions dans la console ; il est possible que cela marche avec d'autres résultats de dialogues, cela n'a pas été testé.
*ToggleFogOfWar	TFOW	Vous permets de voir toute la carte locale.
ToggleFullHelp	TFH	Affiche les scripts et renseignements au passage de la souris pendant que vous êtes dans le mode console ou dans une boite d'information en jouant. Montre aussi dans la console des informations sur la manière dont le jeu utilise des listes (sur l'entrée dans une cellule ou l'ouverture d'un container).
*ToggleGodMode	TGM	Vous rend invulnerable.
ToggleGrid	TG	Affiche les coordonnées de la cellule actuelle, et une grille des cellules actuellement chargées à l'écran, indiquant les activités de la partie « chargement des tâches » du moteur de Morrowind (enregistrement des cellules).
ToggleKillStats	TKS	Quand un personnage est tué dans le jeu, affiche dans la console le nom du mort, le nombre total de tués, et avec Bloodmoon le nombre de tués en loup-garou.
ToggleLights		Aucune idée, n'ayant pas obtenu de sortie dans la console de commande…
ToggleLoadFade		Aucune idée à nouveau…
ToggleMagicStats	TMS	Affiche les informations sur le sort actif dans la console. Donne ses effets, son nom, et ses statistiques.
*ToggleMenus	TM	Arrête tous les menus, dont celui avec sauver/charger/options ! Les menus restent invisibles jusqu'à ce que la console soit réenclenchée (appuyez deux fois sur le bouton qui la charge). Pas reommandé à utiliser dans des scripts ; si le joueur utilise la console, votre script peut stopper définitivement les menus !
ToggleScripts		Arrête probablement les tâches au niveau de la gestion des scripts.
ToggleScriptOutput	TSO	Aucune idée, n'ayant pas obtenu de sortie dans la console de commande…
ToggleStats	TST	Active le débuggage pour les stats (ToggleCombatStats, ToggleMagicStats, ToggleDialogueStats, ToggleKillStats)
*ToggleSky	TS	Bascule pour l'affichage du ciel ; affecte aussi les éclairs. Quand le ciel est arrêté, c'est simplement la nuit.
ToggleTextureString	TTS	Aucune idée, n'ayant pas obtenu de sortie dans la console de commande…
ToggleWater		Aucune idée, n'ayant pas obtenu de sortie dans la console de commande…
*ToggleWorld	TW	Arrête le monde et tous les objets, conservant juste le ciel et l'eau. Cela n'affecte pas les paramètres comme la collision, l'intelligence artificielle, etc. Tout reste actif mais devient simplement invisible.
ToggleWireframe	TWF	Montre les grilles au lieu de procéder à un rendu complet.
TogglePathGrid	TPG	Arrête l'affichage des grilles pour l'intelligence artificielle.
*ToggleVanityMode	TVM	bascule dans le mode Vanity. Contrôlé, cela peut-être particulièrement utile pour des "scènes coupées".
	SA	Montre les statistiques pour l'animation, et les informations concernant l'objet sélectionné : sort actif, arme tirée, attaqué, informations sur le groupe…
ShowGroup	SG	Affiche dans la console les identifications du groupe d'objets sélectionné. Pour le joueur, c'est PlayerSaveGame.
	ST	Affiche dans la console les identifications et les références aux objets "cibles" sélectionnés ; par exemple, les adversaires dans un combat.
ShowScenegraph	SSG	Ouvre une nouvelle fenêtre qui affiche une vue hiérarchique de la structure de la scène actuellement dessinée. Utilisez la touche Tab pour avoir la fenêtre. Attention, Morrowind ne répondra pas tant que cette fenêtre est ouverte.

Paramètres de jeu

Le très long tableau qui suit est une liste des différents paramètres utilisés par le jeu. Tous les paramètres qui ont une valeur numérique sont listés ici. Les entrées en chaînes de caractères – celles qui sont utilisées pour l'affichage des messages texte, des menus, des noms de l'effet des sortilèges, etc... – ne sont pas listés. Mais comme elles sont assez explicites, cela ne devrait pas poser de problèmes. Ce qui n'est pas le cas des paramètres numériques. Merci à quatre membres du forum (maxpublic, Ldones, Wakim et Iudas) : grâce à eux, la signification d'un grand nombre de ces paramètres est maintenant connue et présentée dans le tableau ci-dessous. La liste est toujours à l'état d'ébauche : je ne l'ai pas éditée entièrement mais je suis sûr qu'elle pourra intéresser beaucoup de moddeurs.

Dans beaucoup de cas, vous verrez des paramètres de jeu Base et Mult : ils font partie d'une équation de fonction linéaire tout ce qu'il y a de plus classique y = mx + b, où m est le coefficient directeur, b l'ordonnée à l'origine et x un certain attribut, compétence ou une autre valeur.

Notez que les noms commencent par f ou i (pour floats et integers). Les entrées chaînes de caractères commencent par s (string). À ma connaissance, on ne peut pas les changer pendant le jeu ; c'est seulement possible dans le TES-CS.

Nr.	Nom	Valeur	Description et Commentaires
"465"	"fRepairMult"	1.0000	Correspond à l'efficacité générale de la compétence de réparation d'un personnage, via le marteau utilisé
"466"	"fRepairAmountMult"	3.0000	Indique au jeu combien de points de santé sont redonnés à l'objet lorsqu'on le répare Détermine le coût de réparation des objets (calculé à partir de Max Item Health ou de Item Cost, je ne suis pas sûr)
"467"	"fSpellValueMult"	10.0000	
"468"	"fSpellMakingValueMult"	7.0000	
"469"	"fEnchantmentValueMult"	1000.0000	Paramètre pour le prix à payer à un enchanteur pour enchanter un objet. Linéaire.
"470"	"fTravelMult"	4000.0000	Détermine le coût des voyages à dos d'Echassier des Marais et en bateau (je pense) Multiplie le coût des voyages – je ne sais pas pourquoi on a une valeur aussi élevée mais en l'augmentant, on paye plus cher
"471"	"fTravelTimeMult"	16000.0000	Indique au jeu la durée du temps qui s'écoule pendant un voyage
"472"	"fMagesGuildTravel"	10.0000	Positionne le prix d'un voyage par le Guide de Guilde
"947"	"fWortChanceValue"	15.0000	Iudas: Utilisé pour savoir si une plante "possède" des ingrédients. Wakim: est comparé avec la compétence d'alchimie pour déterminer quels sont les effets d'un ingrédient que vous pouvez voir.
"949"	"fMinWalkSpeed"	100.0000	Vitesse minimum de marche du PJ, indépendamment des stats, compétences ou encombrement
"950"	"fMaxWalkSpeed"	200.0000	Vitesse maximum de marche du PJ, indépendamment des stats, compétences ou encombrement La vitesse actuelle du PJ et des PNJs est déterminée en vérifiant plusieurs fActors (Vitesse, Athletisme, etc...) et basé là-dessus, le jeu assigne une valeur comprise entre fMinWalkSpeed et fMaxWalkSpeed – ces deux paramètres dictent l'éventail des vitesses de marche (Walk Speeds)
"951"	"fMinWalkSpeedCreature"	5.0000	Pareil que pour le PJ mais si vous encombrez sérieusement une créature, elle va se déplacer trèèèès lentement. J'ai pu le voir par accident.
"952"	"fMaxWalkSpeedCreature"	300.0000	Pareil qu'au-dessus, elles vont plus vite, elles couvrent donc l'éventail des vitesses plus rapidement
"953"	"fEncumberedMoveEffect"	0.3000	Indique dans quelle mesure l'encombrement affecte la marche et la course, dans les limites min/max positionnées par les autres valeurs.
"954"	"fBaseRunMultiplier"	1.7500	Changer cette valeur augmentera/baissera la vitesse de course de base. Indique de combien la course est plus rapide que la marche
"955"	"fAthleticsRunBonus"	1.0000	Indique dans quelle mesure Athlétisme affecte la vitesse de course.
"956"	"fJumpAcrobaticsBase"	128.0000	Indique la distance de saut de base pour le PJ.
"957"	"fJumpAcroMultiplier"	4.0000	Multiplicateur pour l'Acrobatie : c'est pourquoi vous pouvez vous jeter du haut de certains bâtiments sans en souffrir lorsque votre score d'Acrobatie est suffisamment élevé.

"958"	"fJumpEncumbranceBase"	0.5000	Indique dans quelle mesure l'encombrement affecte la capacité de saut, mais je ne sais pas comment
"959"	"fJumpEncumbranceMultiplier"	1.0000	Indique dans quelle mesure l'encombrement affecte la capacité de saut, mais je ne sais pas comment
"960"	"fJumpRunMultiplier"	1.0000	INCERTAIN – affecte probablement la distance de saut pendant une course (cela ne semble pas être énorme, mais je peux me tromper)
"961"	"fJumpMoveBase"	0.5000	
"962"	"fJumpMoveMult"	0.5000	
"963"	"fSwimWalkBase"	0.5000	Multiplié à votre vitesse de marche pour déterminer la vitesse de nage en mode 'marche'
"964"	"fSwimRunBase"	0.5000	Multiplié à votre vitesse de course pour déterminer la vitesse de nage en mode 'course'.
"965"	"fSwimWalkAthleticsMult"	0.0200	Indique de quelle façon Athlétisme affecte la vitesse de nage en mode marche. Ces petites valeurs vous empêche de filer à toute vitesse dans l'eau comme c'est le cas sur la terre ferme lorsque l'Athlétisme est élevé.
"966"	"fSwimRunAthleticsMult"	0.1000	Pareil qu'au-dessus mais pour le mode course.
"967"	"fSwimHeightScale"	0.9000	Détermine la distance par rapport à la surface de l'eau à laquelle vous devez vous trouver avant que l'indicateur de souffle ne disparaisse
"968"	"fHoldBreathTime"	20.0000	Nombre de secondes pendant lesquelles le PJ peut retenir son souffle avant d'être affecté par le manque d'oxygène
"969"	"fHoldBreathEndMult"	0.5000	Dans quelle mesure l'endurance affecte le temps pendant lequel vous pouvez retenir votre souffle. Je suppose qu'il s'agit d'un multiplicateur final en secondes, ajouté à HoldBreathTime.<ne semble pas fonctionner.>
"970"	"fSuffocationDamage"	3.0000	Lorsque vous êtes à bout de souffle, nombre de points de santé que vous perdez à chaque seconde
"971"	"fMinFlySpeed"	5.0000	Vitesse minimum lorsque vous volez (dans les airs ☺).
"972"	"fMaxFlySpeed"	300.0000	Vitesse maximum lorsque vous volez (dans les airs ☺).
"973"	"fStromWindSpeed"	0.7000	INCERTAIN - Détermine la vitesse de marche durant une tempête de cendres/fléau, mais je ne sais pas comment – ça pourrait très bien être une valeur complètement à part – peut déterminer la vitesse de déplacement des particules /sprites (poussière, etc...). Note intéressante : alors que je marchais sur l'eau, j'ai remarqué que je me déplaçais difficilement.
"974"	"fStromWalkMult"	0.2500	Détermine la vitesse de marche durant une tempête de cendres ou de Fléau, mais je ne sais pas comment la fonction GetWindSpeed est utilisée pour réduire la vitesse...
"975"	"fFallDamageDistanceMin"	400.0000	Distance minimum, correspondant à la hauteur d'une chute, pour laquelle vous vous blesserez. (Probablement en unités de jeu : chaque unité vaut 1,42 cm
"976"	"fFallDistanceBase"	0.0000	Augmente/diminue la hauteur d'une chute pour laquelle vous vous blesserez.
"977"	"fFallDistanceMult"	0.0700	Plus c'est haut, plus vous vous blesserez en tombant
"978"	"fFallAcroBase"	0.2500	Le talent Acrobatie augmente la hauteur de chute pour laquelle vous ne vous blesserez pas.
"979"	"fFallAcroMult"	0.0100	Dans quelle mesure le talent Acrobatie affecte la hauteur de chute et les dégâts mais je ne sais pas comment
"980"	"iMaxActivateDist"	192	Distance maximum à laquelle le joueur peut activer un objet – à peu près 9 pieds
"981"	"iMaxInfoDist"	192	Distance maximum pour l'affichage d'un message d'information(nom d'un objet/PNJ/créature)
"982"	"fVanityDelay"	30.0000	Secondes avant que le VanityMode ne commence – la caméra tourne autour du joueur lorsqu'aucune entrée (clavier/souris) n'est détectée.
"983"	"fMaxHeadTrackDistance"	400.0000	Si je me souviens bien, distance maximum entre 2 PNJs/créatures telle que la routine 'head follow' se déclenche. Par exemple, placez-vous dans la vue Vanity et vous verrez votre PJ regarder les passantes et suivre leurs mouvements des yeux pendant un certain temps.
"984"	"fInteriorHeadTrackMult"	0.5000	INCERTAIN – a quelque chose à voir avec la routine Head Follow en intérieur – réduit la distance de moitié ?
"985"	"iHelmWeight"	5	Ces valeurs sont utilisées pour déterminer dans quelle catégorie rentre une piève d'armure : légère, intermédiaire ou lourde. En modifiant ces valeurs, vous changerez les armures de *tout* type *partout* dans le jeu. Plutôt bien en fait ; je l'ai utilisé pour modifier tous mes types d'armures.
"986"	"iPauldronWeight"	10	
"987"	"iCuirassWeight"	30	
"988"	"iGauntletWeight"	5	
"989"	"iGreavesWeight"	15	
"990"	"iBootsWeight"	20	
"991"	"iShieldWeight"	15	
"992"	"fLightMaxMod"	0.6000	Valeurs utilisées en conjonction avec les poids d'armures pour déterminer la classe d'armure (Légère, Intermédiaire, Lourde).
"993"	"fMedMaxMod"	0.9000	
"994"	"fUnarmoredBase1"	0.1000	Ces deux valeurs dictent l'étendue du taux d'armure (Armor Rating, AR) pour les personnages sans armure, basées sur le talent Combat sans Armure – tels quels, ces paramètres produisent un AR maximum de 65 (avec 100 dans le talent Combat Sans Armure). Ce qui correspond comme vous pouvez le voir à une multiplication entre les deux paramètres – les valeurs semblent interchangeables : on obtiendra le même effet (à moins d'être passé à côté de quelque chose) – En changer un à 1.000 et conserver l'autre à 0.0650 entraîne une valeur d'armure maximum de 650 – le jeu multiplie les deux nombres puis multiplie le résultat par 1000 pour obtenir la valeur in-game – ne semble pas avoir d'effet sur la valeur minimum – la progression de l'AR (d'un talent bas à un talent élevé) est codé dans l'exécutable. Dès qu'un personnage porte une pièce d'armure d'un type quelconque, le talent Combat sans Armure ne s'applique plus du TOUT. (Forum Info / The other Felix)
"995"	"fUnarmoredBase2"	0.0650	

"996"	"iBaseArmorSkill"	30	Le niveau du talent auquel les pièces d'armure associées atteignent leur valeur d'armure de base (telle qu'elle a été définie dans l'éditeur) – Exemple : une armure de verre a une valeur d'armure de base de 50 – Au niveau 30 du talent Armure légère (comme indiqué à côté) vous verrez une valeur d'armure de 50 dans le jeu – Avant le niveau 30, la valeur d'armure est plus basse que celle de base et après elle est plus élevé – je n'ai pas encore trouvé le procédé du jeu pour déterminer le multiplicateur
"997"	"fBlockStillBonus"	1.2500	INCERTAIN – probablement le bonus correspondant à l'augmentation des chances de parade en restant immobile
"998"	"fDamageStrengthBase"	0.5000	La FORCE ajoute des dommages à ceux infligés par les armes. Détermine la quantité de dommages ajoutés
"999"	"fDamageStrengthMult"	0.1000	Affecte la quantité de dommage dus à la Force pendant un combat (je ne suis pas certain de la façon dont cette valeur modifie l'effet in-game)
"1000"	"fSwingBlockBase"	1.0000	
"1001"	"fSwingBlockMult"	1.0000	
"1002"	"fFatigueBase"	1.2500	Quantité de fatigue perdue en marchant. Cependant il semblerait que cela permette de sauter très haut sans être blessé. (Bug? – Forum Info / DinkumThinkum). 1002 – 1022 Tous les effets de 'Fatigue' in-game – pour des actions séparées/distinguées, même si tous n'ont pas un effet in-game (je n'ai jamais été capable de réduire la fatigue lors du lancement d'un sort) – ces paramètres sont assez explicites mais je ne les ai pas tous testés
"1003"	"fFatigueMult"	0.5000	Utilisé pour déterminer les chances de succès d'un sort si vous êtes fatigué
"1004"	"fFatigueReturnBase"	2.5000	Quantité de fatigue regagnée en 1 seconde. Cela explique pourquoi on ne perd pas de fatigue en marchant.
"1005"	"fFatigueReturnMult"	0.0200	Quantité de fatigue regagnée en 1 seconde lorsqu'on marche
"1006"	"fEndFatigueMult"	0.0400	
"1007"	"fFatigueAttackBase"	2.0000	Quantité de fatigue perdue à chaque attaque de mêlée
"1008"	"fFatigueAttackMult"	0.0000	
"1009"	"fWeaponFatigueMult"	0.2500	
"1010"	"fFatigueBlockBase"	4.0000	Quantité de fatigue perdue en parant une attaque.
"1011"	"fFatigueBlockMult"	0.0000	Augmentera la quantité de fatigue perdue en parant une attaque.
"1012"	"fWeaponFatigueBlockMult"	1.0000	
"1013"	"fFatigueRunBase"	5.0000	Quantité de fatigue perdue en courant.
"1014"	"fFatigueRunMult"	2.0000	Celui-ci fonctionne avec l'encombrement : plus vous êtes chargé, plus vous perdez de la fatigue en une seconde
"1015"	"fFatigueJumpBase"	5.0000	Quantité de fatigue perdue pour un saut.
"1016"	"fFatigueJumpMult"	0.0000	Modificateur pour la perte de fatigue pour un saut
"1017"	"fFatigueSwimWalkBase"	2.5000	Quantité de fatigue perdue en nageant en mode 'marche'
"1018"	"fFatigueSwimRunBase"	7.0000	Quantité de fatigue perdue en nageant en mode 'course'
"1019"	"fFatigueSwimWalkMult"	0.0000	Modificateur pour la perte de fatigue pour la nage en mode 'marche'
"1020"	"fFatigueSwimRunMult"	0.0000	Modificateur pour la perte de fatigue pour la nage en mode 'course'
"1021"	"fFatigueSneakBase"	1.5000	Niveau de base pour la perte de fatigue en mode 'discret'
"1022"	"fFatigueSneakMult"	1.5000	Multiplicateur pour ce niveau de base
"1023"	"fMinHandToHandMult"	0.1000	
"1024"	"fMaxHandToHandMult"	0.5000	
"1025"	"fHandtoHandHealthPer"	0.1000	
"1026"	"fCombatInvisoMult"	0.2000	Réduit les chances de toucher le PJ lorsqu'il est sous l'effet d'un sort de caméléon ou d'invisibilité
"1027"	"fCombatKODamageMult"	1.5000	
"1028"	"fCombatCriticalStrikeMult"	4.0000	Ne marche que si frappez quelqu'un qui ne vous a pas repéré lorsque vous êtes en mode 'discret'. Dommages x4 infligés pour une attaque sournoise réussie ; fonctionne aussi si vous êtes sous l'effet d'un sort de caméléon ou d'invisibilité
"1029"	"iBlockMinChance"	10	Chance minimum de parade
"1030"	"iBlockMaxChance"	50	Chance maximum de parade
"1031"	"fLevelUpHealthEndMult"	0.1000	Multiplié à l'END courante pour déterminer le nombre de points de santé gagnés à la montée de niveau.
"1032"	"fSoulGemMult"	3.0000	La valeur financière d'une gemme spirituelle est multipliée par cette valeur pour déterminer la capacité d'accueil d'une gemme. Les créatures dont la valeur spirituelle est inférieure ou égale à cette capacité peuvent "rentrer" dans la gemme.
"1033"	"fEffectCostMult"	0.5000	Paramètre pour tous les coûts magiques et pour tous les effets de sorts. En changeant cela, vous changerez tous les coûts des sorts et des enchantements. Tout. Changement linéaire. Par exemple, en doublant cette valeur, tous les sorts coûteront 2 fois plus de mana, tous les objets enchantés contiendront 2 fois plus de 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	Nombre minimum d'un ingrédient donné pour fabriquer une potion
"1045"	"fMagicItemCostMult"	1.0000	INUTILISES
"1046"	"fMagicItemPriceMult"	1.0000	
"1047"	"fMagicItemOnceMult"	1.0000	
"1048"	"fMagicItemUsedMult"	1.0000	
"1049"	"fMagicItemStrikeMult"	1.0000	
"1050"	"fMagicItemConstantMult"	1.0000	
"1051"	"fEnchantmentMult"	0.1000	Paramètre déterminant la puissance des enchantements qu'un objet peut contenir basé sur la valeur définie dans les propriétés de chaque objet. Linéaire, si un objet possède une valeur d'enchantement de 1200 dans le TESCS (par exemple, un anneau d'un goût exquis), en la multipliant par fEnchantmentMult, vous obtiendrez l'enchantement actuel tel qu'il apparaît dans la fenêtre de création d'un objet magique du jeu.
"1052"	"fEnchantmentChanceMult"	3.0000	Affecte les chances de réussite du PJ pour créer un objet magique
"1053"	"fPCbaseMagickaMult"	1.0000	Défini le multiplicateur de points de magie du PJ en fonction de l'INT (par exemple 1 x INT avec ce paramètre)
"1054"	"fNPCbaseMagickaMult"	2.0000	Idem pour les PNJs.
"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	Modificateur positif aux relations que vous entretenez avec les gens appartenant à la même faction que vous
"1067"	"iAutoRepLevMod"	0	On peut apparemment ajouter des points de réputation à chaque montée de niveau. Jamais essayé.
"1068"	"iMagicItemChargeOnce"	1	1068-1071 affecte la quantité auto-calculée de charges des objets magiques basé sur leur fonction – cette valeur est le nombre d'utilisations sur lequel le jeu se base pour calculer le nombre de maximum de charges d'un objet magique (marche partout, pour tous les objets magiques du jeu – 1068 : paramètre pour le nombre de charges qu'un objet à effet unique aura. La formule est BaseSpellEffectCost x iMagicItemChargeOnce. Linéaire. De cette façon, un tel objet aura exactement le nombre de charges nécessaires pour lancer cet effet. 1069 : pour les objets à effet constant 1070 : paramètre de multiplicateur de charges lorsqu'on utilise l'effet d'un objet enchanté. Voir l'explication ci-dessus. 1071 : pour les objets à "déclenchement automatique ". (les charges sont calculées pour atteindre X 'utilisations')
"1069"	"iMagicItemChargeConst"	10	
"1070"	"iMagicItemChargeUse"	5	
"1071"	"iMagicItemChargeStrike"	10	
"1072"	"iMonthsToRespawn"	4	Temps (en mois) requis pour que les coffres des guildes (Mages, Guerriers, etc...) se (re)remplissent, que les plantes redonnent des ingrédients. Le contenu créé dans les coffres de guildes est le même que pour tous les autres coffres.
"1073"	"fCorpseClearDelay"	72.0000	Nombre d'heures avant qu'un corps non persistent ne disparaisse.
"1074"	"fCorpseRespawnDelay"	72.0000	Nombre d'heures avant qu'une nouvelle créature ne remplace une créature tuée (notez que cela ne semble pas fonctionner correctement).
"1075"	"fBarterGoldResetDelay"	24.0000	Nombre avant qu'un commerçant ne revienne à sa quantité d'or initiale.
"1076"	"fEncumbranceStrMult"	5.0000	Multiplicateur appliqué à la Force pour connaître le poids maximum qu'un PJ/PNJ/créature peut transporter.
"1077"	"fPickLockMult"	-1.0000	De combien la difficulté de crochetage augmente en fonction du niveau de la serrure – plus cette valeur est basse, plus c'est difficile – des valeurs positives rendront des serrures de haut niveau plus faciles à crocheter
"1078"	"fTrapCostMult"	0.0000	Correspond à la difficulté de désamorçage du piège ; elle est basée sur le coût du sort que déclenchera le piège – plus cette valeur est basse, plus le désamorçage sera difficile Valeur multipliée au coût du sort puis ajouté à votre chance de le désamorcer. Comme c'est 0 par défaut, le coût du sort n'est pas intégré dans les probabilités. Ce paramètre est donc inutilisé.
"1079"	"fMessageTimePerChar"	0.1000	
"1080"	"fMagicItemRechargePerSecond"	0.0500	Paramètre qui gère la quantité de charges rendues à un objet magique par seconde de jeu. Linéaire : 0.05 x 20 secondes = 1 charge rendue.
"1081"	"i1stPersonSneakDelta"	10	
"1082"	"iBarterSuccessDisposition"	1	Si vous réussissez une négociation avec un marchand, sa disposition augmente de 1 ; elle baisse de 1 si vous échouez.
"1083"	"iBarterFailDisposition"	-1	
"1084"	"iLevelupTotal"	10	Nombre total d'augmentations de talents avant une montée de niveau.
"1085"	"iLevelupMajorMult"	1	Combien de points vaut une compétence primaire. Par exemple, placé à 2, pour un point gagné dans une compétence primaire, vous obtiendrez 2 points dans le total

			pris en compte pour la montée de niveau.
"1086"	"iLevelupMinorMult"	1	Idem pour les talents secondaires.
"1087"	"iLevelupMajorMultAttribute"	1	Je *pense* - pas sûr que je me souvienne correctement – mais je pense que ceux-ci fonctionnent comme les précédents, mais pour les talents dépendant de vos 2 caractéristiques primaires. Si vous avez Force et Agilité, et que vous placez 1087 à 2, alors n'importe quelle compétence primaire dépendant de l'une de ces deux caractéristiques qui augmentera d'1 point équivaudra à 2 points pour la montée de niveau.
"1088"	"iLevelupMinorMultAttribute"	1	
"1089"	"iLevelupMiscMultAttriubte"	1	
"1090"	"iLevelupSpecialization"	1	Le jeu garde la trace du nombre de points de compétences que vous avez gagné depuis la dernière montée de niveau. Si vous avez un total de 8 points gagnés dans des compétences dépendant de l'Agilité, alors, à la montée de niveau, vous aurez un multiplicateur de iLevelUp08Mult en face d'Agilité. Si cette valeur est 4, vous verrez un x4 à côté d'AGI et vous gagnerez 4 points en dépensant un point dans l'Agilité.
"1091"	"iLevelUp01Mult"	2	
"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	Valeur minimum de l'âme pour avoir accès à un effet constant dans la fenêtre de création d'enchantement.
"1102"	"fConstantEffectMult"	15.0000	INUTILISE
"1103"	"fEnchantmentConstantDurationMult"	100.0000	Multiplicateur pour le coût de lancement d'un effet constant comparé à celui d'un sort de durée 0. Pour une restauration de santé 2-2 pendant 0 secondes coûtera 0,50 en tant que sort 'normal', l'effet constant correspondant coûtera 0.5 x 100 = 50.
"1104"	"fEnchantmentConstantChanceMult"	0.5000	
"1105"	"fWeaponDamageMult"	0.1000	Dommages appliqués aux armes pendant un combat, lorsque l'arme s'abîme ne somme.
"1106"	"fSeriousWoundMult"	0.0000	INUTILISE
"1107"	"fKnockDownMult"	0.5000	Multiplicateur pour les dégâts infligés lorsqu'un acteur est à terre.
"1108"	"iKnockDownOddsBase"	50	Positionnent les chances de base pour un renversement lorsque la condition est atteinte
"1109"	"iKnockDownOddsMult"	50	
"1110"	"fCombatArmorMinMult"	0.2500	
"1111"	"fHandToHandReach"	1.0000	Portée des armes. Des valeurs inférieures à 1.0 n'ont pas de sens.
"1112"	"fVoiceIdleOdds"	10.0000	Contrôle la probabilité (likelihood) pour qu'un PNJ se mette à parler via un fichier son lorsqu'il folâtre (pas sûr des spécifications)
"1113"	"iVoiceAttackOdds"	10	Contrôle la probabilité (likelihood) pour qu'un PNJ se mette à parler via un fichier son lorsqu'il est attaqué (pas sûr des spécifications)
"1114"	"iVoiceHitOdds"	30	Contrôle la probabilité (likelihood) pour qu'un PNJ se mette à parler via un fichier son lorsqu'il est touché (pas sûr des spécifications)
"1115"	"fProjectileMinSpeed"	400.0000	Vitesse minimum des projectiles des armes
"1116"	"fProjectileMaxSpeed"	3000.0000	Vitesse minimum des flèches et des carreaux d'arbalète
"1117"	"fThrownWeaponMinSpeed"	300.0000	Vitesse minimum des armes de lancers
"1118"	"fThrownWeaponMaxSpeed"	1000.0000	Vitesse maximum des armes de lancers
"1119"	"fTargetSpellMaxSpeed"	1000.0000	Positionne la vitesse des sorts. En doublant cette valeur, vos sorts atteindront l'horizon en un éclair ! – la vitesse minimum est apparemment codé dans l'exécutable
"1120"	"fProjectileThrownStoreChance"	25.0000	Pour la récupération des flèches, carreaux, armes de lancer en fouillant un corps
"1121"	"iPickMinChance"	5	PAS SÛR – je ne sais pas s'il s'agit de pickpocket ou de crochetage (lock picking)
"1122"	"iPickMaxChance"	75	PAS SÛR – je ne sais pas s'il s'agit de pickpocket ou de crochetage (lock picking)
"1123"	"fDispRaceMod"	5.0000	Vos relations avec les membres de votre race sont meilleures.
"1124"	"fDispPersonalityMult"	0.5000	Déterminent dans quelle mesure la personnalité affecte la disposition des PNJs
"1125"	"fDispPersonalityBase"	50.0000	
"1126"	"fDispFactionMod"	3.0000	Déterminent dans quelle mesure votre rang dans une faction modifie vos relations avec ses membres. C'est pourquoi, dans une position élevée, tout le monde est très gentil avec vous.
"1127"	"fDispFactionRankBase"	1.0000	
"1128"	"fDispFactionRankMult"	0.5000	
"1129"	"fDispCrimeMod"	0.0000	Multiplié par le niveau de crime du joueur (prime) pour déterminer comment cette information affecte vos relations avec les PNJs
"1130"	"fDispDiseaseMod"	-10.0000	De combien la disposition diminue lorsque vous êtes malade.
"1131"	"iDispAttackMod"	-50	Pas tout à fait sûr – modificateur de disposition d'un PNJ si le PJ l'attaque
"1132"	"fDispWeaponDrawn"	-5.0000	De combien la disposition baisse si vous avez une arme à la main.
"1133"	"fDispBargainSuccessMod"	1.0000	Je ne me souviens pas si ceux-ci fonctionnent comme les paramètres de négociation décrits plus haut ou si ce sont des multiplicateurs. Affecte la disposition à long terme du marchand
"1134"	"fDispBargainFailMod"	-1.0000	
"1135"	"fDispPickPocketMod"	-25.0000	Modificateur de disposition du PNJ s'il surprend le PJ en flagrant délit de pickpocket sur sa personne
"1136"	"iDaysinPrisonMod"	100	Détermine le temps à passer en prison en fonction du niveau de crime.
"1137"	"fDispAttacking"	-10.0000	Pas sûr – je crois qu'il s'agit d'un modificateur de disposition d'un PNJ qui verrait le PJ attaquer quelqu'un d'autre et qui affecterait sa disposition non-combattante.
"1138"	"fDispStealing"	-0.5000	Pas sûr – même chose pour le vol
"1139"	"iDispTresspass"	-20	Même chose si le PNJ voie le PJ 'mourir' – pas sûr de ce que cela peut signifier dans le jeu
"1140"	"iDispKilling"	-50	Même chose si le PNJ voie le PJ tuer un PNJ innocent (je suppose …)

"1141"	"iTrainingMod"	10	Détermine le coût d'un entraînement. Plus c'est élevé, plus c'est cher. – pas sûr de la méthode de calcul
"1142"	"iAlchemyMod"	2	
"1143"	"fBargainOfferBase"	50.0000	Multiplié à la valeur de l'objet pour déterminer l'offre du marchand durant une vente. La valeur de base change aussi en fonction du niveau du PJ. Quantité d'or que les marchands offriront pour acheter vos objets, en pourcentage – je pense que cela marche dans les 2 sens, mais je ne sais pas comment c'est utilisé pour déterminer la valeur d'achat
"1144"	"fBargainOfferMulti"	-4.0000	Dans quelle mesure le commerçant baisse ses prix pendant une session de marchandage
"1145"	"fDispositionMod"	1.0000	
"1146"	"fPersonalityMod"	5.0000	
"1147"	"fLuckMod"	10.0000	pourcentage : multiplié à votre chance pour obtenir l'augmentation de base à toutes vos compétences.
"1148"	"fReputationMod"	1.0000	
"1149"	"fLevelMod"	5.0000	
"1150"	"fBribe10Mod"	35.0000	De combien la disposition d'un PNJ augmente si la tentative de corruption de 10 PO réussit. – Ne croyez pas qu'il s'agit de la valeur finale – il peut s'agir de pourcentages - (d'autres facteurs comme la race, le sexe ou la faction réduisent significativement la valeur finale)
"1151"	"fBribe100Mod"	75.0000	Idem pour une tentative de corruption de 100 PO.
"1152"	"fBribe1000Mod"	150.0000	Idem pour une tentative de corruption de 1000 PO.
"1153"	"fPerDieRollMult"	0.3000	
"1154"	"fPerTempMult"	1.0000	Utilisé dans presque tous les calculs modifiant la disposition.
"1155"	"iPerMinChance"	5	
"1156"	"iPerMinChange"	10	
"1157"	"fSpecialSkillBonus"	0.8000	Déterminent la vitesse de progression des compétences. Plus la valeur est faible, plus vous progressez vite. Ces valeurs sont multipliées par le taux que vous définissez pour chaque compétence.
"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	Lorsqu'un PNJ lance un cri d'alarme, ce paramètre correspond à la portée de base où les PNJs affiliés réagiront.
"1167"	"iCrimeKilling"	1000	Quantité d'or correspondant aux crimes. Je pense que fCrimeStealing est multiplié par le prix de l'objet volé.
"1168"	"iCrimeAttack"	40	
"1169"	"fCrimeStealing"	1.0000	
"1170"	"iCrimePickPocket"	25	
"1171"	"iCrimeTresspass"	5	
"1172"	"iCrimeThreshold"	1000	Valeur de la prime à laquelle les PNJs commenceront à réagir négativement
"1173"	"iCrimeThresholdMultiplier"	10	
"1174"	"fCrimeGoldDiscountMult"	0.5000	Remise de la guilde des Voleurs lorsque votre tête est mise à prix.
"1175"	"fCrimeGoldTurnInMult"	0.9000	Remise sur l'amende lorsque vous vous rendez.
"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	INUTILISE
"1187"	"iGreetDistanceMultiplier"	6	Utilisé pour ces ennuyeux greetings vocaux lorsque vous êtes trop proche d'un PNJ (si on en croit le Construction Set) multiplié par leur 'hello rating' (taux pour engager la conversation) pour obtenir la distance avant qu'il ne parle.
"1188"	"iGreetDuration"	4	
"1189"	"fGreetDistanceReset"	512.0000	De combien vous devez vous éloigner avant qu'un PNJ ne refasse un test pour les greetings locaux.
"1190"	"fIdleChanceMultiplier"	0.7500	Probablement un multiplicateur concernant les chances qu'un PNJ ne marmonne quelque chose alors que vous êtes à côté de lui
"1191"	"fSneakUseDist"	500.0000	Aide à savoir si vous pouvez passer en mode discrétion
"1192"	"fSneakUseDelay"	1.0000	Aide à déterminer le temps d'apparition de l'icône de discrétion
"1193"	"fSneakDistanceBase"	0.5000	Voir ci-dessus
"1194"	"fSneakDistanceMultiplier"	0.0020	Voir ci-dessus
"1195"	"fSneakSpeedMultiplier"	0.7500	Multiplié à la vitesse de marche de base pour connaître la vitesse de déplacement en mode de discrétion.
"1196"	"fSneakViewMult"	1.5000	Augmente la difficulté d'être ou de rester discret lorsqu'un PNJ regarde dans votre direction.

"1197"	"fSneakNoViewMult"	0.5000	Baisse la difficulté d'être ou de rester discret lorsqu'un PNJ ne regarde pas dans votre direction.
"1198"	"fSneakSkillMult"	1.0000	
"1199"	"fSneakBootMult"	-1.0000	Multiplié à la valeur des bottes (poids?) pour déterminer la réduction de la compétence Discrétion.
"1200"	"fCombatDistance"	128.0000	Combiné à la portée de l'arme pour déterminer la distance effective à laquelle vous pouvez frapper quelqu'un
"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	Les boucliers sont tenus dans la main gauche et bloquent en partie les attaques venant de 90° sur votre gauche à 30° sur votre droite.
"1208"	"fCombatBlockRightAngle"	30.0000	
"1209"	"fCombatDelayCreature"	0.1000	
"1210"	"fCombatDelayNPC"	0.1000	
"1212"	"fAIMeleeWeaponMult"	2.0000	Détermine la distance de fuite d'un PNJ si le PJ combat avec une arme de mêlée
"1213"	"fAIRangeMeleeWeaponMult"	5.0000	Idem si le PJ combat avec un arc ou une arbalète
"1214"	"fAIMagicSpellMult"	3.0000	
"1215"	"fAIRangeMagicSpellMult"	5.0000	Idem si le PJ est prêt à lancer un sort
"1216"	"fAIMeleeArmorMult"	1.0000	
"1217"	"fAIMeleeSummWeaponMult"	1.0000	
"1218"	"fAIFleeHealthMult"	7.0000	Modifie le taux de fuite des opposants à mesure que leur santé diminue
"1219"	"fAIFleeFleeMult"	0.3000	Utilisé pour modifier le taux de fuite de base.
"1220"	"fPickPocketMod"	0.3000	
"1221"	"fSleepRandMod"	0.2500	Affecte la probabilité qu'une créature réveille le joueur s'il dort à la belle étoile.
"1222"	"fSleepRestMod"	0.3000	Inutilisé (merci à Damar Stiehl pour ces deux-là)
"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	Contrôle le délai de disparition de la barre de santé de l'ennemi
"1231"	"fNPCHealthBarFade"	0.5000	Contrôle le nombre de secondes avant que la barre ne s'efface graduellement (au lieu de disparaître brusquement)
"1232"	"fDifficultyMult"	5.0000	
"1399"	"fMagicDetectRefreshRate"	0.0167	
"1400"	"fMagicStartIconBlink"	3.0000	Nombre de secondes avant que l'icône du sort, en bas à droite de l'écran, ne disparaisse graduellement lorsque l'effet s'estompe.
"1401"	"fMagicCreatureCastDelay"	1.5000	
"1431"	"fDiseaseXferChance"	2.5000	La chance d'attraper une maladie si l'on est touché par une créature infectée ou si l'on fouille un corps contaminé.
"1432"	"fElementalShieldMult"	0.1000	
"1435"	"fMagicSunBlockedMult"	0.5000	Faiblesse du vampire
	"fWereWolfRunMult"	1.3000	Coefficient multiplicateur pour la vitesse de course du loup-garou.
	"fWereWolfSilverWeaponDamageMult"	2.0000	Multiplicateur de dommages infligés aux loup-garous avec des armes en argent.
	"iWereWolfBounty"	1000	Caractéristiques et compétences pour les loup-garous.
	"fWereWolfStrength"	150.0000	
	"fWereWolfAgility"	150.0000	
	"fWereWolfEndurance"	150.0000	
	"fWereWolfSpeed"	90.0000	
	"fWereWolfHandtoHand	100.0000	
	"fWereWolfUnarmored"	100.0000	
	"fWereWolfAthletics"	50.0000	
	"fWereWolfAcrobatics"	80.0000	
	"fWereWolfInteligence"	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	
	“iWereWolfFightMod”	100	
	“iWereWolfFleeMod”	100	
	“fWereWolfHealth”	2.0000	
	“fWereWolfFatigue”	400.0000	
	“fWereWolfMagica”	100.0000	
	“fCombatDistaceWereWolfMod”	0.3000	détermine la portée d’attaque d’un loup-garou.
	“fFleeDistance”	3000.0000	Détermine la distance de fuite.

Index

J

L

M

O

P

R

S

T

U

W