

DECOUVERTE LUA

PLUGIN LUA

V 3.1.2.5



GMA 2 - PLUGIN LUA

PRESENTATION

Vous l'avez sans doute remarqué, les versions **GMA2** après 3.0.0.5 intègre une nouvelle palette nommée **PLUGIN**. Je vais vous présenter les **PLUGINS LUA**.

D'abord un peu d'histoire, **Lua** (qui veut dire lune en portugais) est un langage de programmation conçu pour être embarqué au sein d'autres applications.

Il a été créé par l'**université pontificale catholique de Rio de Janeiro**. (Comme quoi tout n'est pas mauvais chez les curés !!) Haaaa je blague !.

Je vais pas plus loin pour l'histoire, vous trouverez suffisamment d'infos si cela vous dit. Sachez seulement que ce langage est très apprécié pour les jeux vidéos, qu'il est compatible avec le langage C et peut donc s'intégrer dans la plupart des projets. Des exemples ? 2 connus : Console sony **PlayStation** et **GTA5**.

Et donc **MA LIGHTING** viens d'intégrer ce langage pour la **GrandMa2**.

C'est complètement personnel et gratuit ce que je vais dire mais qui va créer des plugins en code Lua alors que la plupart des utilisateurs ont du mal avec les macros et la syntaxe voire ne s'y intéressent pas du tout et que sur une échelle de difficultés de 1 à 10 si la création de macros se trouvent à 1 il faudrait placer la création de plugin

Lua à 8 ou 9 .Je pense que dans un futur plus ou moins proche nous achèterons peut être des plugins Lua pour telle ou telle fonction.

Comme je vous dis, c'est mon avis personnel mais j'ai du mal à en imaginer un autre, ou alors.... S'y mettre !.

Et c'est le but de cette découverte, vous donnez envie de vous y intéresser.

DECOUVERTE LUA

Je n'emploie pas le mot tutorial parce que je n'ai pas la prétention de vous apprendre Lua, c'est énormément d'heures de travail sur plusieurs mois pour arriver à maîtriser un tel langage, néanmoins nous allons faire une approche originale nous allons créer un plugin ensemble, même si vous ne captez pas tout, vous pourrez dire Je l'ai fait !

Tout d'abord voici un petit recensement de liens à connaître si vous souhaitez apprendre le langage :

Je commence donc par un livre en Français sur ce langage, un peu cher (50€ pour la totale des modules), mais c'est le seul en Français :

<http://www.d-booker.fr/programmation-et-la.../1-livre-lua.html>

ensuite divers liens qui sont mes sources d'apprentissages certains en Anglais hélas (mais bon je me sers de google traduction et ça fonctionne)

<http://www.lua.org/manual/5.3/>

<http://wxlua.developpez.com/tutori.../general/cours-complet/>

http://www.luteus.biz/.../LUA.../Introduction_Programmation.html

http://www.ozone3d.net/tutorials/lua_coding.php?lang=1

http://wxlua.free.fr/Tutoriel_Lua/sommaire.php

http://wikiromfr.solib.net/index.php/Syntaxe_du_Lua

<http://lua-users.org/wiki/LuaStyleGuide>

<http://lua-users.org/wiki/loLibraryTutorial>

<http://lua-users.org/wiki/ForTutorial>

<http://www.troubleshooters.com/codecorn/lua/luaf.htm>

<http://www.gammon.com.au/scripts/doc.php?lua=tonumber>

http://www.tutorialspoint.com/lua/lua_file_io.htm

http://www.tutorialspoint.com/lua/lua_modules.htm

Tous ces liens sont une mine d'or pour l'apprentissage de lua

REPRISE D'UNE MACRO EN PLUGIN LUA

Tout d'abord pour écrire et lire des fichiers .lua il vous faut télécharger le meilleur et gratuit des logiciels pour cela : **NOTEPAD ++**

<https://notepad-plus-plus.org/fr/>

Nous allons créer un plugin Lua qui va reprendre une macro bien pratique, "**SELECTION INVERSE**" mais qui dans son état ne fonctionne plus maintenant dans la version 3.1.2.3 car les boucles qui étaient nécessaires dans cette macro ne fonctionnent plus après les version 3.0.0.5.

Il faut à présent avoir d'autres macros pour réaliser les boucles, bref comment faire compliqué quand on pouvait faire simple !!

Mais si vous utilisez lua c'est un mal pour un bien parce qu'en plus de pouvoir faire toute sorte de calculs qui sont impossibles à faire en macro, Lua est rapide, très rapide et bien au dessus de la syntaxe MA. Pourquoi cela ?

Le mode **FOLLOW** d'une macro (WAIT) à une vitesse de rafraîchissement de 33ms. Ce qui en fait correspond à la fréquence dmx d'une grand MA2 qui est de 0.033ms.

Ainsi pour 4 lignes de macros avec toutes le mode **FOLLOW** elle sont exécutées comme ceci depuis le lancement de la macro :

Ligne 1 : 0

Ligne 2 : 0.033

Ligne 3 : 0.066

Ligne 4 : 0.099

Ce qui d'ailleurs pose parfois problème, car si par exemple la ligne 2 contient une opération longue à exécuter, la ligne 3 peut finir avant et cela peut planter la macro.

C'est pour cela qu'il est recommandé de mettre un temps au WAIT par ex 0.1 afin de garantir que les lignes ou du moins les opérations qu'elles représentent soient exécutées dans l'ordre. Vous me croyez pas ? Alors faite 2 lignes de macros en mode FOLLOW contenant :

Ligne 1 : **Channel 10 At 50**

Ligne 2: **Park Channel 10**

Et regarder ce qui se passe. Le channel 10 sera bien parké mais pas à 50 car il en a pas eu le temps. L'opération de la ligne 1 est plus longue à réaliser que celle de la ligne 2. Pour que cette macro soit exécutée convenablement il faut obligatoirement mettre un temps dans WAIT dans la ligne 1.

GMA 2 - PLUGIN LUA

MACRO REVERSE SELECTION

Voici la macro "REVERSE SELECTION". Elle permet d'inverser n'importe quelle sélection.

Edit Macro 3 'REVERSE SELECTION'				
No.	Cmd	Wait	Info	Disal
1	SetUserVar \$LoopCount=0	0.01	Reset de la boucle	
2	Store Group "ORIGINAL" /o	0.01	Sauvegarde le groupe Original	
3	Clear	0.01	Deselection des fixtures	
4	Store Group "NOUVEAU" /o	0.01	Prepares la nouvelle selection	
5	Group "NOUVEAU" ; Group "ORIGINAL"	0.01	Selectionne l'ancien et le nouveau groupe	
6	Inverses	0.01	Selection de la derniere fixture de l'ancienne selection	
7	Store Group "NOUVEAU"/m ; \$Store Group "ORIGINAL"/r	0.01	add to new selection and remove from old selection	
8	AddUserVar \$LoopCount=1	0.01	Augmente la boucle	
9	[\$LoopCount < \$SelectedFixturesCount] Macro 1 "REVERSE SELECTION".5	0.01	Repetition boucle d'entree	
10	SelfIn Group "NOUVEAU"	0.01	Selection nouveau groupe qui est inverse	
11	Delete Group "NOUVEAU" + "ORIGINAL" /nc	0.01	Suppression des groupes devenus inutiles	
New				

Elle fonctionne donc plus dans cet état depuis la version 3.1.1.1. De plus pour des sélections conséquentes elle mettra un certain temps à réaliser l'opération.

CREATION PLUGIN LUA

Lancez **NOTEPAD ++**.

Allez dans le menu **LANGAGE** et sélectionnez dans la liste et à la lettre **L** : **Lua**.

La première chose à faire c'est créer un commentaire. Pour cela c'est très facile, il suffit de commencer par deux tirets --

Notepad ++ d'office colorie le texte. Par défaut tout ce qui est écrit en commentaire est de couleur **VERT**.

```
C:\Documents and Settings\Administrateur\Bureau\ASTUCES GMA2\06 CREATION TUTO\DECOUVERTE LUA\
Fichier Edition Recherche Affichage Encodage Langage Paramétrage Macro Exécution Compléter
decouverte.lua x
1 -- Mon premier plugin lua
2
3
4 --[[
5
6 Mon premier plugin lua.
7 Auteur : moi même.
8
9 ---]]
```

Si vous souhaitez écrire plusieurs lignes de commentaires, plus lisibles que de commencer avec 2 tirets pour chaque ligne, vous pouvez écrire en début --[[et terminer par --]]

Sauvegarder votre fichier et nommez le **REVERSE_SELECTION.lua**

Bon passons aux choses sérieuses.

Il va nous falloir déclarer des variables. Comme je ne vais pas vous apprendre le langage, pour faire simple une variable est un mot tout con qui peut contenir quelque chose. Exemple : **MaVariable = 10**. Le mot MaVariable tout con contient à présent la valeur 10 et chaque fois que j'appellerai dans le code MaVariable elle vaudra 10.

Sachez que n'importe quel mot peut être utilisé sauf si c'est une fonction employée par Lua, il ne doit pas contenir d'espace, de caractères spéciaux et ne doit pas contenir uniquement une série de chiffres ni même commencer par un chiffre.

Il existe 2 sortes de variables : La variable **Globale** et la variable **Locale**.

En Lua, une variable Globale s'écrit **MaVariable**, cette variable sera valable et applicable n'importe où dans le code. Cependant il faudra qu'elle soit tuée en quittant le plug. car elle consomme de la mémoire et surtout elle reste en mémoire même après avoir quitté le plug.

Une variable Locale s'écrit **Local MaVariable** et sera valable et applicable dans le bloc de code (appelé chunk) et sera automatiquement tuée en quittant le bloc.

Elle ne sera par contre plus disponible pour d'autres blocs du code.

Nous allons utiliser que des variables locales dans notre plugin mais sachez que pour tuer une variable Globale il faut utiliser **nil** :

MaVariable = nil

MA à mis en route une série de commande permettant de communiquer avec Lua. Toutes ces commandes se trouvent dans le dossier **PLUGINS** et dans le fichier lua **PLUGIN_1**.

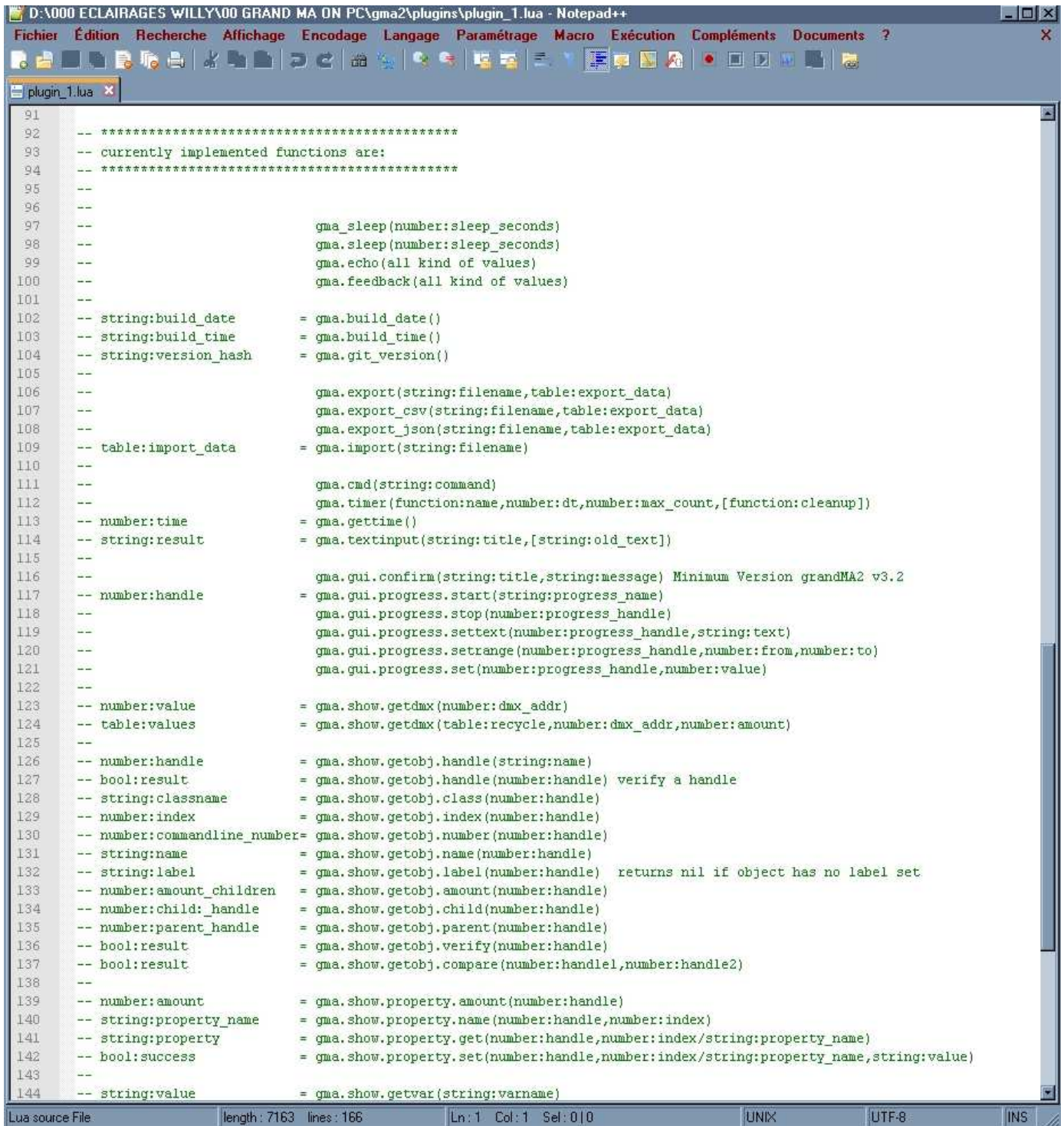
En fin de fichier et en commentaires vous trouverez toutes les commandes. Bien que ça va pas vous parler beaucoup, elle permettent de communiquer entre lua et la console.

Bien sûr c'est assez limité pour le moment, je pense que les prochaines versions verront des ajouts de commandes.

GMA 2 - PLUGIN LUA

COMMANDES MA IMPLEMENTEES POUR LUA

Une partie des commandes implémentées par MA pour communiquer avec Lua que vous trouverez dans le plugin "Plugin_1" fourni par Ma :

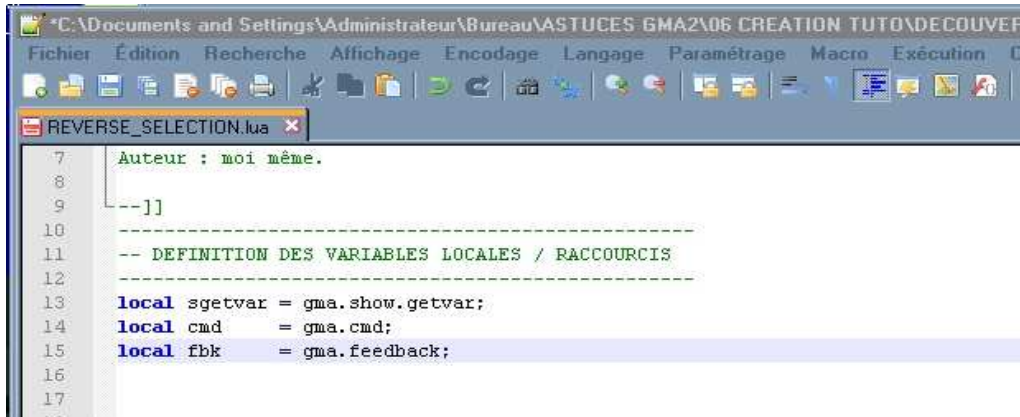


```
91
92 -- *****
93 -- currently implemented functions are:
94 -- *****
95 --
96 --
97 --             gma.sleep(number:sleep_seconds)
98 --             gma.sleep(number:sleep_seconds)
99 --             gma.echo(all kind of values)
100 --            gma.feedback(all kind of values)
101 --
102 -- string:build_date      = gma.build_date()
103 -- string:build_time      = gma.build_time()
104 -- string:version_hash    = gma.git_version()
105 --
106 --             gma.export(string:filename,table:export_data)
107 --             gma.export_csv(string:filename,table:export_data)
108 --             gma.export_json(string:filename,table:export_data)
109 -- table:import_data      = gma.import(string:filename)
110 --
111 --             gma.cmd(string:command)
112 --             gma.timer(function:name,number:dt,number:max_count,[function:cleanup])
113 -- number:time            = gma.gettime()
114 -- string:result          = gma.textinput(string:title,[string:old_text])
115 --
116 --             gma.gui.confirm(string:title,string:message) Minimum Version grandMA2 v3.2
117 -- number:handle          = gma.gui.progress.start(string:progress_name)
118 --             gma.gui.progress.stop(number:progress_handle)
119 --             gma.gui.progress.settext(number:progress_handle,string:text)
120 --             gma.gui.progress.setrange(number:progress_handle,number:from,number:to)
121 --             gma.gui.progress.set(number:progress_handle,number:value)
122 --
123 -- number:value           = gma.show.getdmx(number:dmx_addr)
124 -- table:values           = gma.show.getdmx(table:recycle,number:dmx_addr,number:amount)
125 --
126 -- number:handle          = gma.show.getobj.handle(string:name)
127 -- bool:result            = gma.show.getobj.handle(number:handle) verify a handle
128 -- string:classname       = gma.show.getobj.class(number:handle)
129 -- number:index           = gma.show.getobj.index(number:handle)
130 -- number:commandline_number = gma.show.getobj.number(number:handle)
131 -- string:name            = gma.show.getobj.name(number:handle)
132 -- string:label           = gma.show.getobj.label(number:handle) returns nil if object has no label set
133 -- number:amount_children = gma.show.getobj.amount(number:handle)
134 -- number:child_handle    = gma.show.getobj.child(number:handle)
135 -- number:parent_handle   = gma.show.getobj.parent(number:handle)
136 -- bool:result            = gma.show.getobj.verify(number:handle)
137 -- bool:result            = gma.show.getobj.compare(number:handle1,number:handle2)
138 --
139 -- number:amount          = gma.show.property.amount(number:handle)
140 -- string:property_name    = gma.show.property.name(number:handle,number:index)
141 -- string:property        = gma.show.property.get(number:handle,number:index/string:property_name)
142 -- bool:success           = gma.show.property.set(number:handle,number:index/string:property_name,string:value)
143 --
144 -- string:value           = gma.show.getvar(string:varname)
```


GMA 2 - PLUGIN LUA

DECLARATION DES VARIABLES LOCALES

Déclarez les 3 variables locales comme ci-dessous.



```
7  Auteur : moi même.  
8  
9  --]]  
10  
11  -----  
12  -- DEFINITION DES VARIABLES LOCALES / RACCOURCIS  
13  -----  
14  local sgetvar = gma.show.getvar;  
15  local cmd     = gma.cmd;  
16  local fbk     = gma.feedback;  
17
```

La variable locale **sgetvar** contient la commande MA **gma.show.getvar**. Cette commande va nous servir à interroger la console pour connaître le nombre de fixtures qu'il y a dans la sélection en cours.

La variable locale **cmd** contient la commande MA **gma.cmd**. Cette commande va nous permettre de communiquer à la console par la ligne de commande.

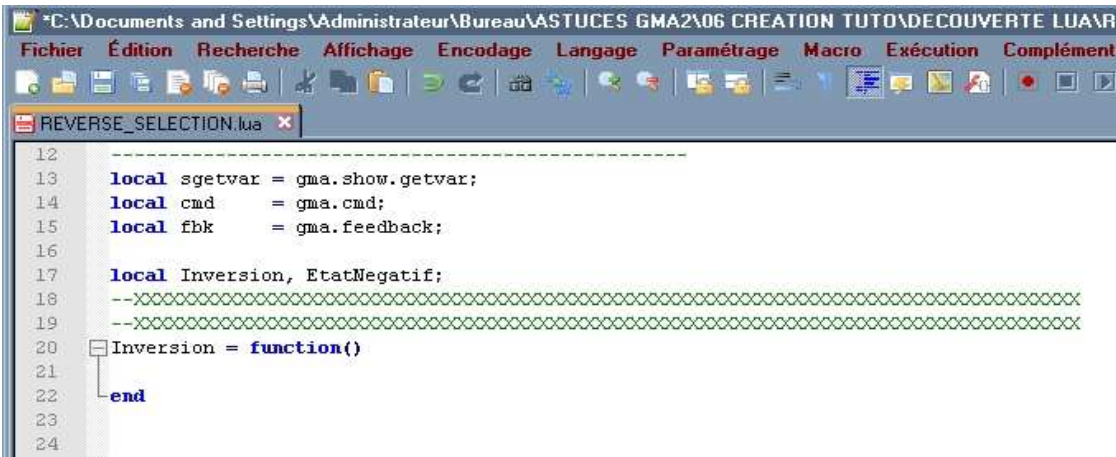
La variable locale **fbk** contient la commande MA **gma.feedback**. Cette commande va nous permettre d'écrire dans la commandLine afin d'avoir des instructions ou simplement des infos dans la commandLine.

sgetvar, **cmd** et **fbk** sont simplement des raccourcis. Nous pourrions dans le code écrire directement la commande mais il se trouve que la commande est réalisée beaucoup plus rapidement en passant par un raccourcis. Me demandez pas pourquoi, c'est ainsi. Demandez au Vatican c'est eux qui ont pondus Lua. :)

Nous allons à nouveau déclarer 2 variables locales. **Inversion** et **EtatNegatif**.

Attention respecter bien l'écriture, les accents sont interdits, les majuscules / minuscules n'ont pas d'importances c'est juste pour que ce soit visuel.

Vous pouvez également remarquer que les deux variables sont écrites sur la même ligne. C'est tout à fait possible, il suffit de bien respecter la virgule. Notez que le point virgule à la fin de la ligne (vous le verrez souvent d'ailleurs) n'est là que par convention. Il ne sert trictement à rien d'autre.



```
12  -----  
13  local sgetvar = gma.show.getvar;  
14  local cmd     = gma.cmd;  
15  local fbk     = gma.feedback;  
16  
17  local Inversion, EtatNegatif;  
18  --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
19  --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
20  Inversion = function()  
21  
22  end  
23  
24
```

Ces 2 variables locale vont nous servir à écrire nos blocs de code.

Ecrivez à l'identique la ligne 20. Notre titre de bloc vient d'être créé.. Faites 2 retours chariots (enter) et terminez par end. Notez que si **function** et **end** ne sont pas de couleur **BLEU** c'est que vous l'avez mal écrit. Il faut par contre pour les fonctions Lua toujours écrire en minuscule.

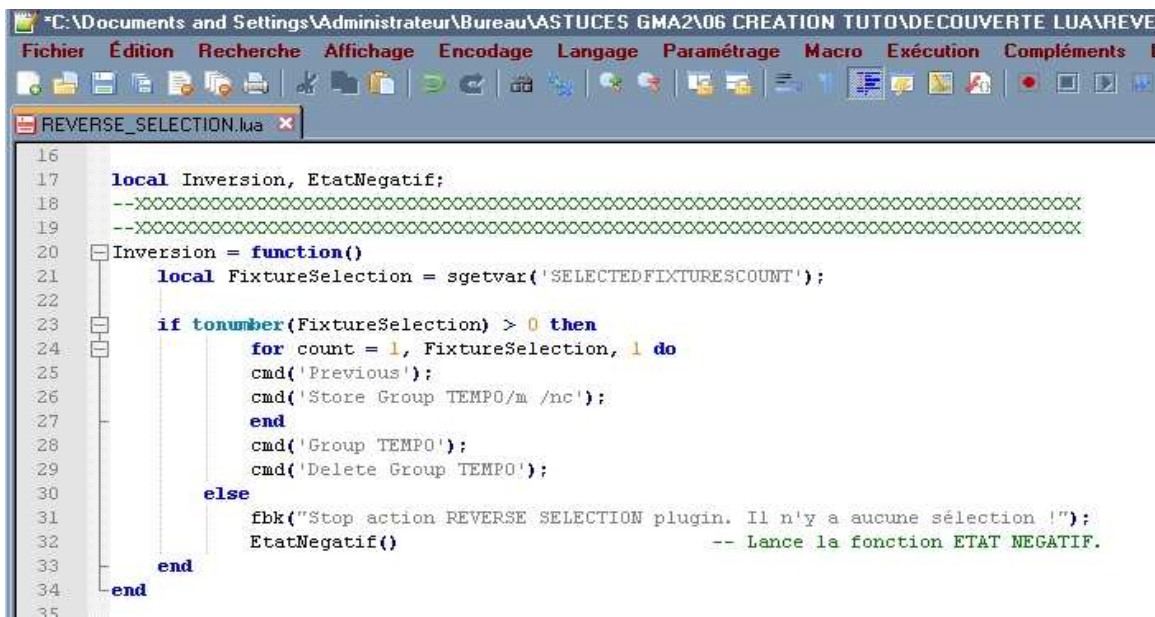
Nous allons maintenant écrire le code qui permet d'inverser une sélection dans le bloc **Inversion**.

GMA 2 - PLUGIN LUA

ECRITURE DE LA FONCTION INVERSION

Voici le code à écrire dans le bloc **INVERSION** pour les lignes 21 à 33.

Ne vous inquiétez pas je vais vous les commenter ligne par ligne.



```
16
17 local Inversion, EtatNegatif;
18 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
19 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
20 Inversion = function()
21     local FixtureSelection = sgetvar('SELECTEDFIXTURESCOUNT');
22
23     if tonumber(FixtureSelection) > 0 then
24         for count = 1, FixtureSelection, 1 do
25             cmd('Previous');
26             cmd('Store Group TEMPO/m /nc');
27         end
28         cmd('Group TEMPO');
29         cmd('Delete Group TEMPO');
30     else
31         fbk("Stop action REVERSE SELECTION plugin. Il n'y a aucune sélection !");
32         EtatNegatif()
33     end
34 end
35
```

Ligne 21 : `local FixtureSelection = sgetvar('SELECTEDFIXTURESCOUNT');`

Cette ligne permet grâce à la commande `sgetvar` (rappelez vous, `gma.show.getvar`) d'interroger la console pour récupérer le nombre de fixtures qui sont dans la sélection grâce à la variable MA `$SELECTEDFIXTURESCOUNT` (celle-ci existe dans la syntaxe depuis la version 3.1.1.1)

Veuillez noter que le signe \$ ne doit pas être écrit dans une commande Lua.

Ligne 23 : `if tonumber(FixtureSelection) > 0 then`

Il veut dire si. Si le nombre `FixtureSelection` (ma sélection donc) est supérieure à zéro (c'est à dire qu'il y a une sélection donc) alors faire ...

Ligne 24 : `for count = 1, FixtureSelection, 1 do`

For permet de faire une boucle. (Ha si ça pouvait exister en macro !!). Cette boucle va donc s'effectuer de 1 à `FixtureSelection`, c'est à dire au nombre que la sélection contiendra, par exemple avec une sélection de 50 fixtures **for** va faire boucler 50 fois.

Le dernier 1 est facultatif, c'est le nombre de pas pour la boucle. 1 est le pas standard, il aurait pu y avoir 0.1 pour le pas (cela dit je vous conseille pas !!).

Enfin **do** est la terminaison de **for**, une boucle **for** se finie toujours par **do**.

Ligne 25 : `cmd('Previous');`

C'est ici que nous communiquons avec la console grâce à la variable **cmd** (commande `gma.cmd`). Nous lui demandons de faire un Previous.

Ligne 26 : `cmd('Store Group TEMPO/m /nc');`

La ça devrait vous être familière. C'est simplement la syntaxe que l'on utilise en macro. Cela store un groupe qui va s'appeler TEMPO en merge et sans confirmation (nc).

Ligne 27 : `end`

C'est la fin de la boucle **for**. Qu'a fait cette boucle **for** ?

Elle a bouclé de 1 à x (x étant le nombre de fixtures de la sélection en cours) à fait pour chaque tour un Previous et à storer le groupe TEMPO en merge.

En clair nous avons stocké dans le groupe TEMPO la sélection en l'inversant mais de manière ultra rapide.

Ligne 28 : `cmd('Group TEMPO');`

Nous sélectionnons le groupe TEMPO afin d'activer cette nouvelle sélection inversée.

Ligne 29 : `cmd('delete Group TEMPO');`

Et enfin nous supprimons ce groupe devenu inutile.

Ligne 30 : `else`

Else veut dire sinon. Dans le bloc Inversion nous avons démarré par **if** (à la ligne 23) qui voulait dire si le nombre `FixtureSelection` est supérieur à zéro faire la boucle **for**, else signifie que si la sélection est à zéro (c'est à dire qu'il n'y a pas de sélection) faire les lignes qui suivent.

Ligne 31 : `fbk("Stop action REVERSE SELECTION PLUGIN. Il n'y a aucune sélection !");`

C'est une ligne d'info qui sera écrite dans la CommandLine grâce à la variable **fbk** (command `gma.feedback`) afin d'informer l'utilisateur qu'il n'y avait pas de sélection et donc le plugin n'a pas réalisé l'action d'inversion.

Vous remarquerez peut être une petite subtilité. Entre les parenthèses je n'utilise plus des apostrophes mais des guillemets. En fait c'est pour le code la même chose, vous pouvez utiliser guillemets ou apostrophes. Dans ce cas présent par contre il y a une apostrophe au mot **il n'y a pas**... et du coup vous êtes obligé d'utiliser des guillemets pour pouvoir écrire cette apostrophe, sans ça, cela provoquerai une erreur.

Ligne 32 : `EtatNegatif()`

C'est la seconde variable locale que nous avons déclarée au tout début. En fait dans ce bloc elle appelle un autre bloc que nous allons détaillé maintenant.

Ligne 33 : `end`

C'est le End pour fermer le **if** de la ligne 23.

Ligne 34 : `end`

C'est le End pour fermer le bloc de code Inversion. (appelé également je vous le rappelle chunk).

GMA 2 - PLUGIN LUA

ECRITURE DE LA FONCTION ETAT NEGATIF

Notre premier bloc Inversion() étant écrit nous allons maintenant écrire notre second et dernier bloc nommé EtatNegatif.

Les lignes 35 à 37 permettent en commentaire de donner le maximum d'infos et de rendre le code plus lisible par des annotations. Ne négligez pas cette façon de faire, il est très important de commenter du code pour vous, quand par exemple vous voulez y revenir plusieurs mois après pour faire une retouche ou l'améliorer.

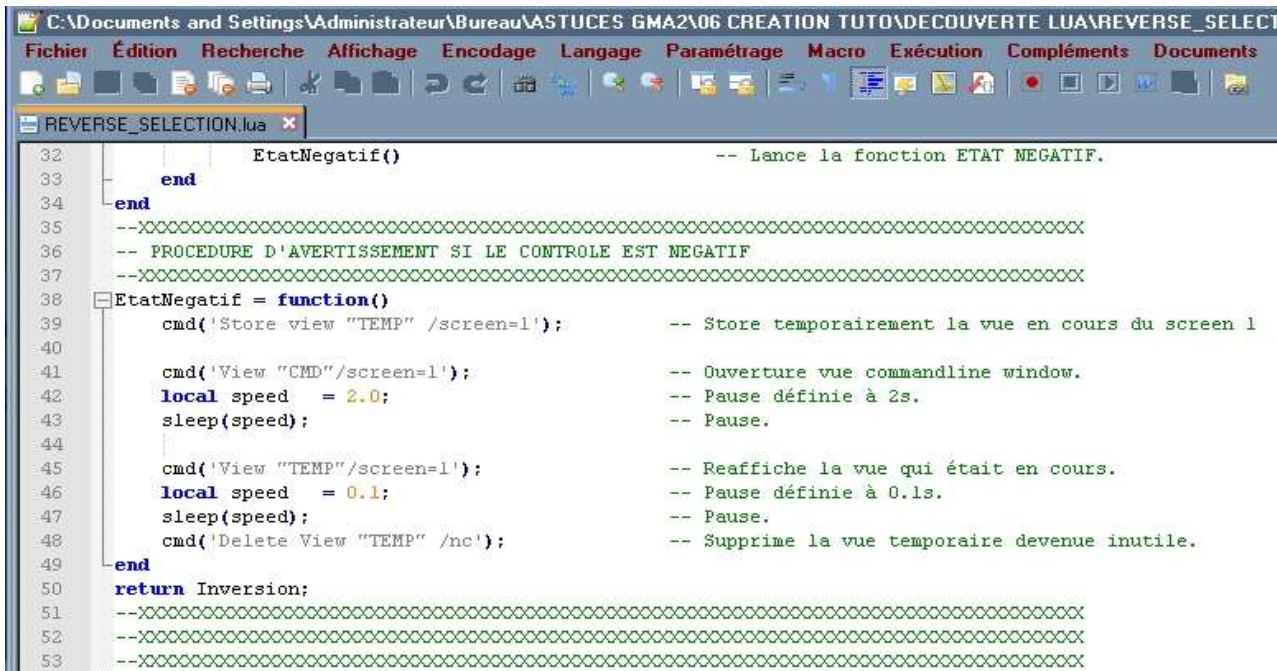
De même pour les noms de variables, pensez à choisir des noms explicites. Il en doit jamais y avoir d'espace dans un nom de variable mais par contre celui-ci peut être long; Par exemple ; **Inversion** est tout de même plus parlant que si j'avais choisi **a**. J'aurai d'ailleurs même du choisir **InversionSelection**().

A quoi sert le bloc EtatNegatif ?

Nous avons vu dans le premier bloc (Inversion) si la sélection est supérieure à zéro (donc qui contient une sélection) la boucle **for** effectue l'inversion. Si par contre la sélection est vide, par exemple tout simplement parce que l'utilisateur a lancé le plugin et a omis de sélectionner les fixtures à inverser avant, plutôt que le plugin s'arrête et qu'il ne se passe rien d'autre, il est parfois utile de l'avertir. J'ai en tout cas fait ce choix dans le plugin.

Avant d'écrire le bloc EtatNegatif il faudra créer une vue dans le screen 1 qui contiendra la CommandLine et que vous nommerez **"CMD"**. Storer la.

Cette vue à présent sera stockée dans la palette **VIEW**, pas forcément affichée dans le screen 1 mais qui pourra être appelée par le plugin quand c'est nécessaire.



```
32      EtatNegatif()                                -- Lance la fonction ETAT NEGATIF.
33  end
34 end
35 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
36 -- PROCEDURE D'AVERTISSEMENT SI LE CONTROLE EST NEGATIF
37 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
38 EtatNegatif = function()
39     cmd('Store view "TEMP" /screen=1');             -- Store temporairement la vue en cours du screen 1
40
41     cmd('View "CMD"/screen=1');                     -- Ouverture vue commandline window.
42     local speed = 2.0;                             -- Pause définie à 2s.
43     sleep(speed);                                  -- Pause.
44
45     cmd('View "TEMP"/screen=1');                     -- Reaffiche la vue qui était en cours.
46     local speed = 0.1;                             -- Pause définie à 0.1s.
47     sleep(speed);                                  -- Pause.
48     cmd('Delete View "TEMP" /nc');                 -- Supprime la vue temporaire devenue inutile.
49 end
50 return Inversion;
51 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
52 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
53 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Ligne 38 : EtatNegatif = function()

Je déclare donc la variable locale EtatNegatif en fonction. Ce sera en quelque sorte le titre de mon second bloc.

Ligne 39 : cmd('Store View "TEMP" /screen=1');

J'utilise la variable cmd pour indiquer à la console de stocker la view en cours du screen 1. Cette vue temporaire va permettre de retrouver ensuite son screen 1 original.

Ligne 41 : cmd('View "CMD" /screen=1');

J'utilise la variable cmd pour indiquer à la console de m'afficher la vue "CMD" dans le screen 1 .

Ligne 42 : local speed = 2.0

La variable locale speed créée ici contient une valeur de de 2.0. Cela équivaut à 2 secondes et sera utilisée pour la ligne suivante.

Ligne 42 : sleep(speed);

La variable sleep (gma.sleep) est une pause et va faire poser le code pendant 2 secondes.

Ligne 45 : cmd('View "TEMP" /screen=1');

J'utilise la variable cmd pour indiquer à la console de m'afficher la vue "TEMP" dans le screen 1 . Je retrouve mon screen 1 comme à l'origine.

Vous remarquez donc quye la vue "CMD" est resté affichée 2 secondes puis vous retrouvez le screen 1 d'origine.

Ligne 46 : local speed = 0.1

La variable locale speed est remis à 0.1 qui est sa vitesse de pause de croisière.

Ligne 47 : sleep(speed);

La variable sleep (gma.sleep) exécute la nouvelle valeur de speed qui a été définie à 0.1.

Ligne 48 : cmd('Delete View "TEMP" /nc');

Supprime la vue TEMP devenue inutile. /nc veut dire No Confirm dans la syntaxe MA et permet de supprimer la vue sans ouvrir de fenêtre de confirmation.

Ligne 49 : end

C'est le End final du bloc EtatNegatif.

et enfin la dernière ligne mais qui est la plus importante, sans qui rien ne fonctionnerai :

Ligne 50 : return Inversion;

Qui permet en lançant le plugin d'exécuter la fonction Inversion. Si à la place de Inversion on remplaçait cela par EtatNegatif, au lancement du plugin c'est le bloc EtatNegatif qui serait exécuté.

GMA 2 - PLUGIN LUA

UTILISER LE PLUGIN

Voilà votre plugin est terminé (et sauvegardé bien sur).

Avant de l'utiliser il nous faut encore créer un fichier .xml qui va permettre d'importer le plugin. **GrandMa2** ne traite qu'avec des fichiers .xml et donc un fichier .lua pour qu'il puisse l'importer il lui faudra le faire au travers d'un fichier xml.

Nous allons pas écrire ce fichier car c'est long et chiant.

Prenez et faite une copie du fichier .xml plugin_1 (que nous avons déjà vu et qui contient toutes les commandes MA pour Lua).

Renommez le à l'identique que votre plugin soit **REVERSE SELECTION**.

Clic droit sur le fichier pour l'éditer et sélectionner **Edit With notepad++**. Hé oui notepad peut également éditer un fichier .xml. Je crois qu'il peut d'ailleurs éditer pratiquement n'importe quel fichier de codage.

Vous y trouverez 5 lignes.

Ligne 2 : En bout de la première ligne vous y trouverez :

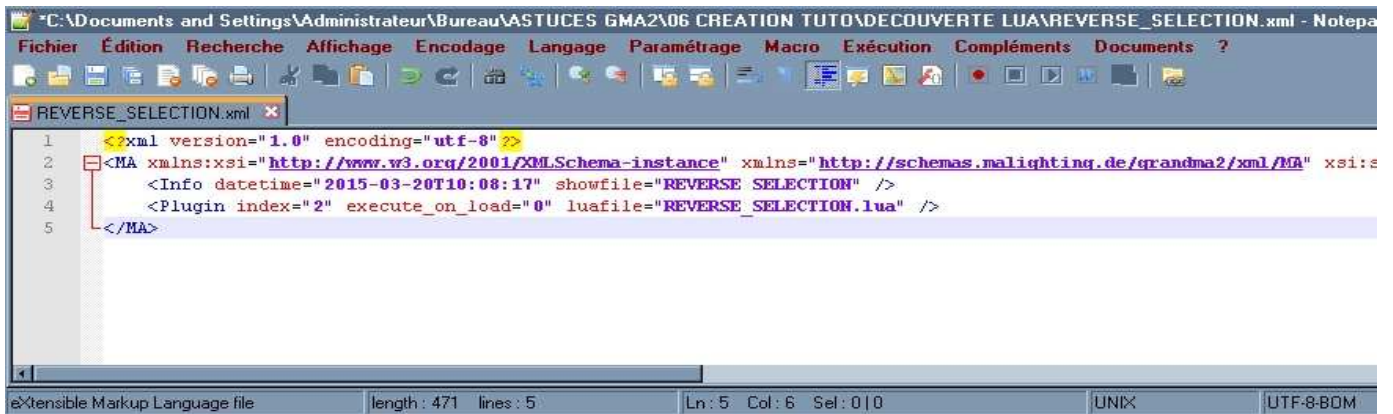
```
major_vers="3" minor_vers="0" stream_vers="187">
```

Remplacer la version par 3.1.2 pour correspondre avec la version en cours, comme ceci

```
major_vers="3" minor_vers="1" stream_vers="2">
```

Ligne 3 : remplacer **preset bug** par **REVERSE SELECTION**. C'est facultatif.

Ligne 4 : remplacer **plugin_1.lua** par **REVERSE SELECTION.lua**. Ici c'est obligatoire sinon le plugin ne sera jamais importé dans GrandMa2.



Voilà c'est presque fini.

placez les deux fichiers REVERSE SELECTION .lua et .xml dans le dossiers **plugins** (même niveau que les dossiers macros, show, effects, etc..... Du dossier principal **gma2** sur votre clé usb.

Dans la console, afficher dans un écran vide la palette **PLUGINS**. (dernier onglet nommé).

Editer un espace plugin vide.

Cliquer sur le bouton **IMPORT**.

Sélectionner votre clé usb et choisissez votre plugin.

C'est cette fois fini !! Pffffff

Faites une sélection ou activer un groupe puis cliquer sur le plugin. Clic clac merci Kodak (c'est pas une expression de jeune ça :)).

N'oubliez pas de créer cette vue "CMD" qui contient la CommandLine pour le screen 1 .

Après l'avoir créé, lancez le plug mais cette fois sans aucune sélection.

Miracle la vue s'affiche et repart seule après 2s.

Vous pourrez également apprécier la vitesse d'exécution du plugin avec un nombre conséquent de fixtures dans la sélection.

CONCLUSION

J'espère avoir été assez clair et peut être cela vous donnera l'envie d'aller plus loin en Lua. C'était le but sans prétention aucune, je passe énormément de temps sur ce code j'aime ça, il faut d'ailleurs aimer ça pour se farcir des lignes de code.

Cependant vous pourrez déjà convertir de nombreuses macros existantes en plugin Lua. C'est déjà pas mal.

Je tiens à remercier **Jean-Benoit meunier** (un opérateur GrandMa2 Canadien) qui à l'origine à créer le premier plugin de reverse sélection, je n'ai fait que l'adapter à mesure que je progressais dans le code lua mais en tout cas il m'a donné envie d'aller plus loin en lua.

Qu'il en soit une nouvelle fois remercier.

William.

GMA 2 - PLUGIN LUA

PLUGIN REVERSE SELECTION

Le code du plugin REVERSE SELECTION en entier.

```
C:\Documents and Settings\Administrateur\Bureau\ASTUCES GMA2\06 CREATION TUTO\DECOUVERTE LUA\REVERSE_SELE...
Fichier Édition Recherche Affichage Encodage Langage Paramétrage Macro Exécution Compléments Documents ? X

REVERSE_SELECTION.lua x
1  -- Mon premier plugin lua
2
3
4  --[[
5
6  Mon premier plugin lua.
7  Auteur : moi même.
8
9  --]]
10 -----
11 -- DEFINITION DES VARIABLES LOCALES / RACCOURCIS
12 -----
13 local sgetvar = gma.show.getvar;
14 local cmd      = gma.cmd;
15 local fbk      = gma.feedback;
16
17 local Inversion, EtatNegatif;
18 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
19 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
20 Inversion = function()
21     local FixtureSelection = sgetvar('SELECTEDFIXTURESCOUNT');
22
23     if tonumber(FixtureSelection) > 0 then
24         for count = 1, FixtureSelection, 1 do
25             cmd('Previous');
26             cmd('Store Group TEMPO/m /nc');
27             end
28             cmd('Group TEMPO');
29             cmd('Delete Group TEMPO');
30         else
31             fbk("Stop action REVERSE SELECTION plugin. Il n'y a aucune sélection !");
32             EtatNegatif() -- Lance la fonction ETAT NEGATIF.
33         end
34     end
35 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
36 -- PROCEDURE D'AVERTISSEMENT SI LE CONTROLE EST NEGATIF
37 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
38 EtatNegatif = function()
39     cmd('Store view "TEMP" /screen=1'); -- Store temporairement la vue en cours du screen 1
40
41     cmd('View "CMD"/screen=1'); -- Ouverture vue commandline window.
42     local speed = 2.0; -- Pause définie à 2s.
43     sleep(speed); -- Pause.
44
45     cmd('View "TEMP"/screen=1'); -- Reaffiche la vue qui était en cours.
46     local speed = 0.1; -- Pause définie à 0.1s.
47     sleep(speed); -- Pause.
48     cmd('Delete View "TEMP" /nc'); -- Supprime la vue temporaire devenue inutile.
49 end
50 return Inversion;
51 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
52 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
53 --XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Lua source File |length: 2049 |lines: 53 |Ln: 1 Col: 1 Sel: 0|0 |Dos\Windows |UTF-8 |INS
```