Documentation LUA-YAG API

Version de ce document

Si vous avez trouvé ce document dans votre installation de Yag, vous aurez peut être besoin de récupérer la dernière version à l'adresse suivante: http://yagame.fr/documentation/

Yag et LUA

Un des buts de Yag est de rester aussi généraliste que possible en ce qui concerne les règles.
Ca soulève le besoin de le personnaliser en fonction du jeu visé.
Mais Yag est codé en c++ et son fonctionnement interne est inaccessible aux joueurs.
Une réponse courante à ce problème est d'embarquer dans le programme hôte (Yag) un langage de script comme LUA.

Pour échanger des données, certaines parties de Yag doivent être exposées à LUA:

- Quelques fonctions et variables doivent être accessibles pour les joueurs à travers LUA
- Yag doit pouvoir appeler des fonctions LUA

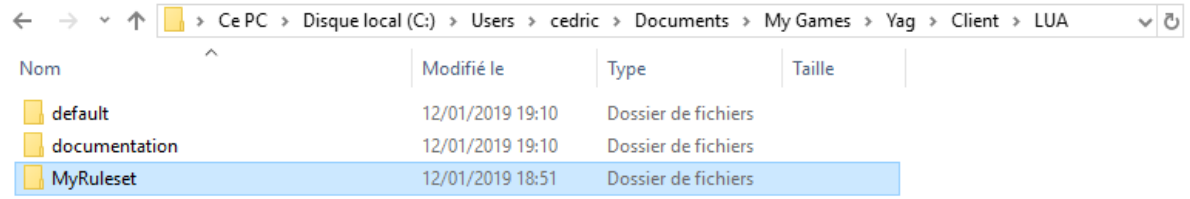
Ce document décrit les outils actuellement disponibles pour ce faire.

La version de LUA embarquée est la v5.3.

Fonctionnement général

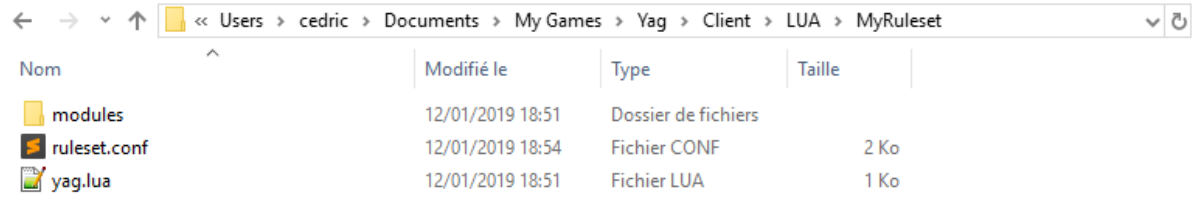
Pour appeler du code LUA, Yag utilise le contenu d'un répertoire qui doit:
- Se trouver dans <my documents>/My Games/Yag/Client/LUA
- Avoir un nom unique qui est spécifié dans le panneau LUA dans l'interface de Yag.

Par exemple si le nom du répertoire est MyRuleSet, il doit être créé comme suit:
<my documents>/My Games/Yag/Client/LUA/MyRuleSet



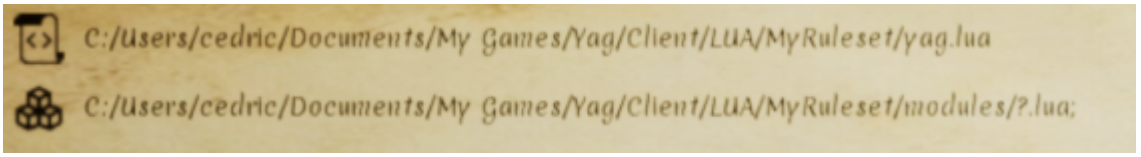
Et être renseigné comme suit dans Yag:

Ce répertoire doit obligatoirement contenir un fichier ruleset.conf à sa racine.



Ce fichier contient 2 variables obligatoires:
- LUA_FILE qui contient le chemin relatif du fichier que Yag lancera pour exécuter du code LUA. Ce chemin doit être donné relativement au répertoire <my documents>/My Games/Yag/Client/LUA/MyRuleSet .
- LUA_PATH qui contient une liste de chemins, également relatifs à <my documents>/My Games/Yag/Client/LUA/MyRuleSet et séparés par un **";"**. Cette variable est utilisée pour créer la variable d'environnement LUA_PATH (dans laquelle les chemins relatifs seront transformés par Yag en chemins absolus) qui informe LUA des chemins où il doit chercher les modules.

Le résultat de ces deux variables est affiché (et non modifiable) dans le panneau LUA:



Si vous avez installé des jeux de dés personnalisés, le fichier ruleset.conf contient également 3 variables optionnelles (CUSTOM_DICE_01, CUSTOM_DICE_02, CUSTOM_DICE_03) qui permettent de spécifier le nom de 3 jeux de dés.
Attention les jeux de dés spécifiés doivent être installés indépendamment du répertoire LUA. Pour plus de détails, vous pouvez regarder la vidéo sur les dés personnalisables:
https://www.youtube.com/watch?v=baECyr8j0Qw

Chaque fois qu'on fait appel à un mécanisme LUA dans Yag, Yag exécute le fichier LUA_FILE qui doit donc contenir les fonctions de l'API dédiés à la communication entre Yag et LUA (voir plus bas).

Par défaut un répertoire default est créé et écrasé chaque fois qu'on lance Yag.
Ce répertoire contient un fichier ruleset.conf ainsi qu'un fichier yag.lua et un module default.lua, qui définissent ensemble le comportement par défaut de Yag.
Ce répertoire est un exemple qui peut être dupliqué et renommé comme point de départ d'un ruleset personnalisé.

IMPORTANT: Yag ne partage aucun mécanisme LUA entre le serveur et les clients.
Chaque script LUA tourne localement, chaque fonction est appelée localement.
Ca signifie que chaque joueur connecté doit installer le répertoire qui sera utilisé par sa copie locale de Yag.
En général ça revient simplement à dézipper une archive préparée par un des joueurs et renseigner son nom dans le panneau LUA de Yag.

Pour lancer du LUA, Yag procède de la façon suivante:

- Il lance toujours le script LUA_FILE complet pour identifier et charger les objets LUA (fonctions…): si on met du code en dehors des fonctions, il sera exécuté
- Il lance optionnellement une fonction LUA de l'api Yag, en fonction du contexte.

Par exemple:

- Le bouton LUA dans le panneau LUA lance simplement le script et n'appelle aucune fonction. Ca peut servir à vérifier si le script et les modules sont présents et accessible, et, si le script fait quelque chose, ça peut transformer Yag en interpréteur LUA.
- Quand on clique sur le bouton LUA dans le panneau des feuilles de perso, Yag lance le script puis appelle la fonction Yag_CharacterSheetUpdate, qui doit être fournie par le joueur dans le script.
- Quand on clique sur un bouton de dés, Yag lance le script puis appelle la fonction Yag_RollDiceButton qui doit être fournie par le joueur. Si les dés sont lancés, après avoir lu les résultats, Yag les envoie au joueur via la fonction Yag_RollDiceCallback qui doit être fournie par le joueur.

Plus de détails sur ces 3 fonctions plus bas.

Dans le script LUA, Yag est accessible par l'objet yag qui contient divers objets et functions.
Par exemple:

- yag.LuaDir contient le chemin complet du répertoire <my documents>/My Games/Yag/Client/LUA.
- yag.PublicMessage() écrit un message public dans le journal

Les structures

Les entrées et sorties des fonctions ont des structures spécifiques qui sont décrites ici.

Les feuilles de personnages (cs)

cs = character sheet = feuille de perso en anglais
Les feuilles de perso dans LUA sont des tables avec la structure suivante:

```
{
  name = “my cs name”,
  pictureUrl = “https://mySite/myPicture.png”,
  init = “1d20+1”,
  lines =
  {
    id1 = { id = “id1”, mod = true, modTargetId = “target id”, tab = 1, extraString= “whatever”, extraBool = true, string1 = “type”, string2 = “name”, formula1 = “1d20”, formula2 = “1d8”, show = {showMod = true, showModTargetId = true, showId = true, showExtraString = true, showExtraBool = true, showString1 = true, showString2 = true, showFormula1 = true, showFormula2 = true, showRollDice = true}, position = { x = 13.4, y = 17.2}, scale = 1.5, fontSize = 12,  showFrame = true},
    id2 = { id = “id2”, mod = true, modTargetId = “target id”, tab = 1, extraString= “whatever”, extraBool = true, string1 = “type”, string2 = “name”, formula1 = “1d20”, formula2 = “1d8”, show = {showMod = true, showModTargetId = true, showId = true, showExtraString = true, showExtraBool = true, showString1 = true, showString2 = true, showFormula1 = true, showFormula2 = true, showRollDice = true}, position = { x = 13.4, y = 17.2}, scale = 1.5, fontSize = 12,  showFrame = true},
  ...
  },
  ids = { 1 = “ID1”, 2 = “ID2”, … }
}
```

ou plus court:

```
{
name,
pictureUrl,
init,
lines {
    id {
        id,
        mod,
        modTargetId,
        tab,
        extraString,
        extraBool,
        string1,
        string2,
        formula1,
        formula2,
        show = {showMod, showModTargetId, showId, showExtraString, showExtraBool, showString1, showString2, showFormula1, showFormula2, showRollDice},
        position = {x, y},
        fontSize,
        showFrame,
        scale
    }
     ... },
ids
```

}

name (string) est le nom du perso
pictureUrl (string) est l'URL de l'image du perso
init (string) est la formule de l'initiative
lines est la structure d'une ligne:

- mod (booléen) active ou désactive la ligne comme modificateur
- Lorsque la ligne est activée comme modificateur, modTargetId (string) est un filtre sur les ids des autres lignes. Le modificateur sera appliquée à toute autre ligne dont l'id contient son modTargetId .
- tab (entier) est le numéro de l'onglet dans lequel la ligne doit apparaître sur le bureau portable.
- extraString (string) est un champ qui n'est pas utilisé par Yag, il est donné au joueur pour n'importe quelle utilisation où quelque chose doit être utilisé par LUA et stocké dans Yag: variables, conditions, listes de propriétés… C'est juste une string gratuite.
- extraBool (booléen) est un champ non utilisé par Yag, il est donné au joueur pour n'importe quell utilisation dans LUA. C'est juste une checkbox gratuite.
- string1 (string) est une étiquette libre qui représente en général le type de la ligne ("caractéristique", "compétence", "inventaire", "attaque", etc..)
- string2 (string) est une étiquette libre qui représente en générale le nom de la ligne ("force", "perception", "épée de feu", etc…)
- formula1 (string) est une formule de dés, correspondant en général au test de la ligne (toucher, test de force, etc…)
- formula2 (string) est une formule de dés, correspondant en général à l'effet de la ligne (dégats).

Ces champs correspondent à ceux situés dans la ligne cochée ici:

Par exemple, si la feuille de perso ci dessus est stockée dans la variable cs:

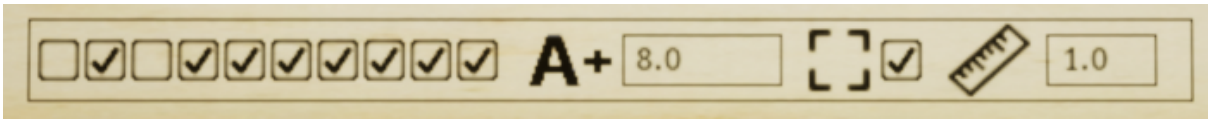
- cs.lines["w3"].formula1 va retourner "1d20+1"
- cs.lines["dex"].string1 va retourner "ability"
- cs.lines["w3"].extraString va retourner "bow"
- cs.lines["int"].extraBool va retourner true
- cs.lines["str"].mod va retourner false

Les champs suivants sont les propriétés graphiques:

- position (flottants) sont les coordonnées x et y du champ sur la surface. (0, 0) est en haut à gauche. La position n'est pas accessible numériquement dans l'UI, on la modifie en déplaçant les champs à la souris.
- show est une collection de booléens (showMod, showModTargetId, etc.) décidant si on affiche ou pas la propriété correspondant sur le champ graphique.

- fontSize (entier) est la taille de la police du champ graphique
- showFrame (booléen) décide si on encadre chaque champ.
- scale (flottant) est l'échelle du champ graphique

Ces champs correspondent aux suivants, situés sur le bureau:



ids est une table contenant les identifiants des pions attachés à la feuille de perso. L'id d'un pion est celui qu'on trouve sur la feuille des pions avec l'icone de code-barre:

Attention à ne pas confondre les ids de lignes avec les ids de pions:

- cs.lines["LineID"] = la ligne (de la feuille de perso) identifiée par "LineID"
- cs.ids[1] = l'id du premier pion attaché à la feuille de perso

Chaque id (string), ici id1 et id2, devrait être unique. Notez que, pour laisser une liberté complète au joueur, Yag ne s'occupe pas de l'unicité des ids.
Les objets (pions sur la map ou lignes au sein d'une feuille de perso) avec des ids identiques peuvent créer des problèmes (disparition d'objets ou mauvaises mises à jour).
Aussi bien pour les id de lignes que de pions, Yag génère un id unique par défaut avec les lettres "ID" suivies d'un nombre séquentiel.
Pour les feuilles de perso, ce nombre est réinitialisé quand la feuille ne contient aucune ligne.
Pour les pions, ce nombre est réinitialisé quand la map ne contient plus aucun pion.

Les feuilles de pions (ps)

ps = pawn sheet = feuille de pion en anglais
Une feuille de pion est une table contenant des information sur un pion.
Voici la structure d'une ps:

```
{
 health = 10,
 maxhealth = 100,
 mana = 50,
 maxmana = 100,
 initiative = 14,
 round = true,
 righthand = true,
 lefthand = true
}
```

health, maxhealth, mana, maxmana et initiative sont des entiers.
round est un booléen spécifiant si le pion participe au round courant.
Ils correspondent à ces champs dans le panneau des pions:

righthand est un booléen spécifiant si le contenu de la main droite est visible (pour les pions humains)
lefthand est un booléen spécifiant si le contenu de la main gauche est visible (pour les pions humains)

Les résultats de dés (dr)

dr = dice result = résultat de dés en anglais
Les résultats de dés contiennent beaucoup d'informations: la formule d'origine, la valeur, l'étiquette et la couleur de chaque dé lancé.
Voici la structure d'un jet complet "3d6 + 1d4 + 7":

```
{
 1 =
 {
  formula = "+bonus",
  color = "wood",
  result = { 1 = { value = 7, label = "7" } }
 },
 2 =
 {
  formula = "3d6",
  color = "white",
  result =
  { 2 = { value = 3, label = "three" }, 3 = { value = 6, label = "six" }, 1 = { value = 5, label = "five" } }
 },
 3 =
 {
  formula = "1d4",
  color = "black",
  result =
  { 1 = { value = 4, label = "four" } }
 }
}
```

ou plus court:
{ { formula, color, result { {value, label} … } }, ... }

La clé pour chaque formule est un nombre
La formule est une string
La clé de chaque line { value, label, color } est l'ordre des dés dans chaque formule: dans "3d6", une clé "2" signifie que que 3 était le deuxième dé lancé. Dans l'exemple ci dessus, dans le résultat "3d6", 5 était le premier dé, 3 le deuxième, et 6 le troisième.

Notez que LUA ne garantit pas l'ordre de ses tables, donc ces clés sont le seul moyen d'être certain de l'ordre des dés lancés.

Origine du jet de dés (dro)

dro = dice roll origin = origine du jet de dés en anglais
dro (dice roll origin) est un nombre de 1 à 3 qui spécifie l'origine du jet de dés

- 1 lancé en cliquant sur un bouton de ligne de feuille de perso:
  - 

- 2: lancé en cliquant sur le bouton du panneau des dés
  - 

- 3: lancé en cliquant sur le bouton d'init:

- 

L’objet yag (LUA vers Yag)

Depuis LUA on peut accéder à l’objet yag qui nous permettra d’échanger avec Yag.
Il a les propriétés suivantes.

yag.LuaDir

- Le répertoire <my documents>/My Games/Yag/Client/LUA
- Utile si on distribue du contenu LUA et qu’on a besoin d’un répertoire reconnu sur tous les ordinateurs.

yag.RollDice(f1, f2, ui, us)

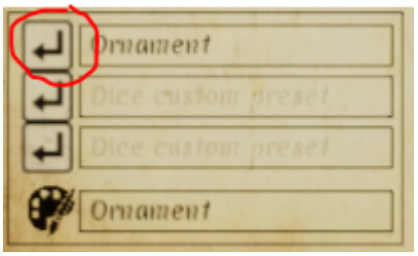
- Demande à Yag de lancer les 2 formules f1 et f2
- f1 et f2 sont des strings
- ui (user integer) est un nombre libre que vous pouvez utiliser comme bon vous semble: il sera passé tel quel à la fonction Yag_RollDiceCallback. Vous pouvez l’utiliser pour propager dro de Yag_RollDiceButton vers Yag_RollDiceCallback.
- us (user string) est une string libre que vous pouvez utiliser comme bon vous semble: elle sera passée telle quelle à la fonction Yag_RollDiceCallback.
- yag.RollDice(f1, f2, ui, us) ne renvoie rien
- Après que les dés aient été lancés et les résultats lus, Yag appellera la fonction callback Yag_RollDiceCallback, voir plus bas pour les détails.

yag.ClearDice()

- Arrête tout jet de dés en cours et efface tous les dés.
- yag.ClearDice() ne prend pas d’argument et ne renvoie rien

yag.LoadDice(ds)

- Charge le jeu de dés ds. (ds = DiceSet = jeu de dé en anglais)
- ds est une string
- yag.LoadDice(ds) ne renvoie rien
- yag.LoadDice(ds) a le même effet que cliquer sur une présélection de dés ds dans l’interface
- Par exemple, avec la configuration ci dessous, l’appel yag.LoadDice(“Ornament”) aura le même effet qu’un clic sur le bouton cerclé de rouge dans le panneau des dés:



- yag.LoadDice(“”) recharge les dés par défaut.

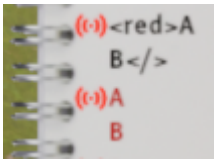
yag.PublicMessage(s)

- Ecrit la string s dans le journal.
- Tous les joueurs verront le message: peut être utilisé pour les résultats de dés
- s est une string
- Des parties de s peuvent être colorées avec des balises:
  - <color>un mot</> affichera “un mot” dans la couleur spécifiée
  - Exemple: yag.PublicMessage("Red <red>fish</> is <green>not green</>")

  - Les couleurs suivantes sont autorisées: red, green, blue, magenta, cyan, yellow, gray, black
  - Les noms des couleurs sont en anglais (traductions: red=rouge, green=vert, blue=bleu, magenta=magenta, cyan=cyan, yellow=jaune, gray=gris, black=noir)
  - black est la couleur par défaut si rien d’autre n’est spécifié.
  - Les balises ne fonctionnent pas autour d’un saut de ligne (“\n”):

    ```
    yag.PublicMessage("<red>A\nB</>")
    yag.PublicMessage("<red>A</>\n<red>B</>")
    ```

  - Les balises ne peuvent pas être imbriquées:

    ```
    yag.PublicMessage("<red>A<blue>B</></>")
    ```

- yag.PublicMessage(s) ne renvoie rien

yag.LocalMessage(s)

- Ecrit la string s dans le journal.
- Seul le joueur appelant verra le message: peut être utilisé pour des logs personnelles ou des jets de dés privés.
- s est une string
- Des parties de s peuvent être colorées, ça fonctionne identiquement à yag.PublicMessage(s).
- yag.LocalMessage(s) ne renvoie rien

yag.GetPawnInfo(id)

- Fournit des informations sur le pion identifié par id
- id est une string qui doit être l’identifiant d’un pion

- yag.GetPawnInfo renvoie dans l'ordre:
    - une structure ps qui est la feuille de pion du pion id
    - une structure cs qui est la feuille de perso du pion id (peut être nil si le pion n'est attaché à aucune feuille de perso
- ex: MyPawnPs, MyPawnCs = yag.GetPawnInfo("MyPawnId")
- **Note importante:** GetPawnInfo ne fonctionne que pour les pions sélectionnés au moment de la requête LUA, c'est à dire les pions qui sont dans la liste sp des fonctions LUA appelées par Yag. Ce n'est pas un choix mais une contrainte technique.

yag.UpdatePawn(ids, ps)

- Met à jour tous les pions identifiés dans la table ids avec la structure ps
- ids est une table de strings contenant les ids des pions: { "ID1", "ID24", "MyID", … }
- ps est une structure ps
- yag.UpdatePawn ne renvoie rien
- Attention: la même structure ps (avec toutes ses propriétés) sera appliquée à tous les pions dans ids, il est donc parfois plus prudent de mettre à jours vos pions un par un en utilisant une table contenant un seul id:
    - yag.UpdatePawn({ "MyID" }, MyPs })

yag.PawnAttack(ids)

- Déclenche les animations d'attaque de tous les pions identifiés dans la table ids.
- ids est une table de strings contenant les ids des pions: { "ID1", "ID24", "MyID", … }
- yag.PawnAttack ne renvoie rien

yag.UpdateCS(cs)

- Cette fonction permet de modifier la feuille de perso courante (en anglais "feuille de perso" se dit "character sheet", d'où le nom "cs").
- cs a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- yag.UpdateCS ne renvoie rien

Les fonctions LUA appelées par Yag (Yag vers LUA)

Quand on a besoin de demander à Yag d'effectuer une tâche LUA, comme mettre à jour une feuille de perso ou personnaliser un résultat de dé, on clique des boutons dans l'interface Yag.
Ces boutons appellent des fonctions LUA dans le script LUA du joueur.
Ces fonctions doivent être fournies par le joueur dans son script, et sont décrites ici.

Yag_CharacterSheetUpdate(sp, cs)

- Cette fonction est appelée quand on modifie une feuille de perso (en anglais "feuille de perso" se dit "character sheet", d'où le nom "cs").
- sp (selected pawns) est une table contenant les id (string) de tous les pions sélectionnés.
- cs a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).

- Cette fonction prend une cs en argument et renvoie une cs.
- Voici la forme minimale de cette fonction qui doit apparaître dans le script LUA appelé par Yag (elle ne fait rien puisque la cs est renvoyée non modifiée):

```
function Yag_CharacterSheetUpdate(sp, cs)
  return cs
end
```

Quand c'est demandé, Yag envoie la feuille de perso courante en argument de cette fonction. Quand Yag reçoit la cs retournée par la fonction, Yag va mettre à jour la feuille de perso dans l'interface en conséquence.

Yag_RollDiceButton(sp, cs, id, f1, f2, dro)

- Cette fonction est appelée quand on lance des dés (en cliquant un bouton dans l'interface).
- sp (selected pawns) est une table contenant les id (string) de tous les pions sélectionnés.
- cs a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- id est une string contenant l'id unique de la ligne à laquelle f1 et f2 appartiennent.
- f1 et f2 sont deux strings contenant les 2 formules à lancer.
- dro (dice roll origin) est un nombre donnant l'origine du jet de dé (voir le chapitre des structures plus haut). Vous avez la possibilité de le propager à la fonction Yag_RollDiceCallback avec le paramètre ui de la fonction Yag_RollDice.
- Yag_RollDiceButton ne renvoie rien mais quand le jet est terminé et que les résultats ont été lus par Yag, la fonction Yag_RollDiceCallback est appelée par Yag pour permettre au joueur de faire ce qu'il veut avec les résultats des dés.
- Voici la forme minimale de cette fonction qui doit apparaître dans le script LUA appelé par Yag, elle ne fait rien (les dés ne seront même pas lancés):

```
function Yag_RollDiceButton(sp, cs, id, f1, f2, dro)
end
```

Notez que lorsque l'appel est fait depuis une feuille de perso, f1 et f2 ne sont pas nécessaires puisqu'ils peuvent être retrouvés avec cs et id comme suit:

- f1 = cs.lines[id].formula1
- f2 = cs.lines[id].formula2

Mais parfois il n'y a ni cs ni id (quand un jet est demandé directement depuis le panneau des dés par exemple), donc ils sont fournis en entrée.

Notez également que cette fonction ne fait rien par défaut: si on le souhaite, il faut explicitement lui demander de nettoyer les dés et de les lancer avec les fonctions yag.ClearDice and yag.RollDice. Il est donc possible de ne pas utiliser du tout les dés de Yag: il suffit de ne jamais appeler la fonction yag.RollDice et de gérer les dés nous même avec des nombres aléatoires et les règles qu'on veut.

Yag_RollDiceCallback(sp, cs, id, dr1, dr2, pd, ui, us)

- C'est la fonction callback appelé après un jet de dés, automatique (appel par Yag de Yag_RollDiceButton(sp, cs, id, f1, f2)) ou manuel (appel par le joueur de yag.RollDice(f1, f2)).
- sp (selected pawns) est une table contenant les id (string) de tous les pions sélectionnés.
- cs a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- id est une string contenant l'id unique de la ligne à laquelle f1 et f2 appartiennent.

- dr1 et dr2 ont la structure d'un résultat de dés ("dice result" en anglais, d'où le nom "dr") (voir le chapitre des structures plus haut)
- dr1 et dr2 sont les résultats respectifs de f1 et f2.
- pd (private dice) est un booléen qui passe exactement la valeur décidée dans Yag pour les dés privés sur les cases à cocher suivantes:  .
  Ainsi pd vous dit ce que veut l'utilisateur et vous pouvez décider d'utiliser yag.PublicMessage ou yag.LocalMessage conformément pour afficher les résultats dans le journal.
  L'utilisation standard de pd pour respecter le souhait de l'utilisateur est la suivante:

```
if (pd) then
    yag.LocalMessage(rs)
else
    yag.PublicMessage(rs)
end
```

  où rs est la string du résultat à afficher dans le journal.
- ui (user integer) est le paramètre ui passé par yag.RollDice.
- us (user string) est la string us passée par yag.RollDice.
- Yag_RollDiceCallback(sp, cs, id, dr1, dr2) ne renvoie rien.
- Voici la forme minimale de la fonction qui doit apparaître dans le script LUA lancé par Yag, elle ne fait rien:

```
function Yag_RollDiceCallback(sp, cs, id, dr1, dr2, pd, ui, us)
end
```