Les alias dans CocoMUD

Les alias sont une fonctionnalité des clients MUD afin d'abréger une ou plusieurs commandes. Un alias est en vérité une commande, mais au lieu d'être envoyée au serveur, elle est d'abord analysée par le client, qui peut la modifier et la transmettre au serveur si besoin. Il est même possible de créer des alias qui n'ont de sens que pour le client et ne sont pas envoyés au serveur du tout.

Ce document décrit le concept d'alias et explique la façon de les créer et les gérer dans CocoMUD.

Ajouter un alias

Vous pouvez créer un alias via l'interface ou en utilisant la syntaxe du SharpScript. L'interface étant plus intuitive, elle sera décrite en premier.

Création grâce à l'interface

Dans la barre de menu, sélectionnez Jeu -> Alias. Vous vous trouverez dans une boîte de dialogue listant les alias existants pour cet univers. Notez que les alias sont généralement propres à un univers et partagés par les personnages de cet univers, ce qui changera dans les prochaines versions.

Dans cette boîte de dialogue, vous pouvez ajouter, éditer et supprimer des alias. Par défaut, la liste des alias de cet univers sera probablement vide, vous pouvez cliquer sur **Ajouter** pour en ajouter un.

Vous devrez ensuite choisir le nom de l'alias que vous voulez créer. C'est le nom de la commande que vous devrez entrer pour exécuter l'alias. Dans cet exemple, nous allons créer un alias "tts" qui permet d'activer ou de désactiver le TTS (text-to-speech).

Entrez donc "tts" dans ce champ de texte, puis appuyez sur la tabulation. Vous vous trouvez ensuite dans une liste d'actions possibles à lier à cet alias. Par exemple, vous pourriez vouloir créer un alias "rs" qui envoie deux commandes au serveur quand exécuté : "reload" et "shoot". Dans ce cas, vous devrez sélectionner l'action "Envoie une ou plusieurs commandes au serveur".

Poursuivant notre exemple, nous devrons choisir "Active/Désactive le TTS", puisque c'est ce que devrait faire notre alias "tts" quand exécuté. Sélectionnez donc ce choix et faites tabulation pour cliquer sur le bouton **Ajouter l'action**.

Vous vous trouverez dans une nouvelle boîte de dialogue vous demandant des informations supplémentaires. Dans le cas de cette action, il n'y en a aucune, appuyez donc sur "OK" pour ajouter l'action.

La nouvelle action a bien été ajoutée. Le curseur se trouvera sur la liste des actions actuellement liées à cet alias. Vous pouvez lier un alias à plusieurs actions, si besoin lier un alias à la même action plusieurs fois (pour par exemple jouer différents sons en même temps, même si cela n'est peut-être pas bien utile).

La liste des actions liées à cet alias fait un peu peur. Elle ne contient que les informations en résumé. Si vous avez choisi "Active/Désactive le TTS", la ligne d'action #tts sera créée. C'est un raccourci, une ligne d'action décrite en utilisant la syntaxe du SharpScript. Vous n'avez pas vraiment besoin de vous en inquiéter, à moins de vouloir toucher au scripting dans CocoMUD.

Pour résumer le contenu de cette boîte de dialogue quand vous ajoutez ou éditez un alias, vous avez:

- Le nom de l'alias (un champ de texte).
- La liste des actions liées à cet alias. Si l'alias n'est lié à aucune action, cette liste n'apparaîtra pas.
- Un bouton pour éditer la ligne d'action sélectionnée. Si l'alias n'est lié à aucune action, ce bouton n'apparaîtra pas.
- Un bouton pour supprimer la liste d'action sélectionnée. Si l'alias n'est lié à aucune action, ce bouton n'apparaîtra pas.
- Une liste d'actions que vous pouvez relier à cet alias.
- Le bouton pour ajouter une de ces actions.

Vous devriez vous habituer autant que possible à cette boîte de dialogue, car elle sera présente pour la plupart des fonctionnalités incluant du SharpScript dans CocoMUD. Il s'agit en fait de l'éditeur de SharpScript, qui vous permet d'éditer la configuration avancée du client sans ouvrir le fichier de configuration. Vous verrez bien d'autres exemples dans le reste de la documentation.

Pour enregistrer cet alias, n'oubliez pas de sélectionner "OK" plusieurs fois pour bien refermer toutes les boîtes de dialogues ouvertes.

Si dans le client, vous entrez "tts", vous devriez voir :

```
TTS off.
```

Entrez cette commande de nouveau pour l'activer. Le TTS (text-to-speech) sera donc activé ou désactivé en entrant cet alias.

Syntaxe du SharpScript

Vous pouvez aussi ajouter un alias en éditant un fichier de configuration. Cette solution sera peut-être préférée par certains.

Le fichier de configuration lié à cet univers se trouve dans le dossier "worlds", sous-dossier portant le nom de votre univers, fichier "config.set". Vous pouvez ouvrir ce fichier avec un éditeur simple, comme bloc-notes.

Pour ajouter un alias, utilisez l'action #alias avec deux paramètres :

- Le nom de l'alias.
- Les actions liées à cet alias.

Si vous voulez créer un alias "tts" qui active ou désactive le TTS (text-to-speech), écrivez dans votre fichier de configuration :

```
#alias tts #tts
```

Si vous voulez lier votre alias à plusieurs actions, n'hésitez pas à l'écrire sur plusieurs lignes :

```
#alias go {
    #play sounds/go.wav
    #say {C'est parti !}
}
```

Pour une explication plus détaillée de la syntaxe du SharpScript, référez-vous à la section consacrée au SharpScript.

Editer un alias

A tout moment, que vous ayez créé l'alias depuis l'interface ou depuis le fichier de configuration, vous pouvez l'éditer. Si vous l'avez créé via le fichier de configuration, vous pouvez l'éditer via l'interface, et réciproquement.

Souvenez-vous : la configuration est chargée quand l'univers est sélectionnée, vraisemblablement à l'ouverture du programme. Si vous modifiez le fichier de configuration, redémarrez le programme pour voir ces modifications.

Supprimer un alias

Vous pouvez retirer un alias depuis l'interface ou le fichier de configuration. Souvenez-vous de bien quitter l'interface en cliquant sur "OK" même après avoir supprimé un alias, sans quoi la modification ne sera pas sauvegardée.

Les alias avec variables

CocoMUD offre une gestion des variables. On peut les utiliser n'importe où dans les scripts du client, incluant dans toutes les fonctionnalités qui utilisent le moteur du SharpScript (comme les alias, macros ou triggers).

Les variables peuvent être très utiles pour les alias, pour créer des raccourcis de commande avec des paramètres inconnus. Par exemple :

> Je voudrais créer un alias qui commence par =. Ce qui vient après le signe égal doit être envoyé au serveur comme une commande say. Si j'entre "=coucou !", la commande "say coucou !" devrait être envoyée au serveur.

Dans ce cas, on devra utiliser des variables. L'alias doit commencer par un signe égal (=), mais on ne sait pas ce qui vient après.

Comment l'indiquer au client ? On va utiliser le signe * dans notre alias qui veut dire "n'importe quoi, de n'importe quelle longueur".

Si on entre =* comme alias, CocoMUD l'interprétera comme "n'importe quelle commande commençant par un signe égal".

Cependant, on n'a pas seulement besoin de déclencher cet alias. On doit récupérer ce que l'utilisateur a entré après le signe égal. Voici le détail de la création de cet alias, pas à pas, en utilisant l'interface :

- Dans la barre de menu, sélectionnez **Jeu** puis **Alias**.
- Cliquez sur le bouton **Ajouter** pour ajouter un nouvel alias.
- Dans le champ de texte contenant l'alias, entrez =* (le signe égal et un signe astérisque).
- Appuyez sur tab pour vous déplacer sur la liste d'actions pouvant être liées à cet alias.
- Sélectionnez "envoie une ou plusieurs commandes au serveur", appuyez sur tab et cliquez sur **Ajouter l'action**.
- Le client vous demande de configurer cette action. Appuyez sur tab jusqu'à entendre "Commandes à envoyer au serveur". Dans ce champ de texte, entrez say $1 (le $1 est expliqué plus bas).
- Appuyez sur **OK**. L'action devrait être ajoutée, et vous devriez la voir dans la liste :

  ```
  #send {say $1}
  ```

  La syntaxe est expliquée plus bas.
- Appuyez sur tab jusqu'à trouver le bouton **OK** et cliquez dessus. Vous vous retrouvez dans la liste d'alias, appuyez sur "OK" une nouvelle fois pour fermer cette boîte de dialogue en sauvegardant les modifications.

Si vous écrivez dans le client :

```
=Coucou !
```

L'alias enverra "say Coucou !" au serveur.

Ce qui se passe est assez simple :

- Vous entrez "=quelque chose".
- CocoMUD trouve un alias correspondant à cette commande.
- Ce que vous avez entré après le signe égal ("quelque chose", dans notre exemple) est capturé dans la variable $1.
- Quand vous envoyez $1 au serveur, le $1 sera remplacé par "quelque chose".

La variable s'appelle $1 car vous pouvez créer de nombreuses variables. Si vous entrez l'alias :

```
-*+*
```

Vous pourrez entrer comme actions liés à cet alias :

```
remove $1
wear $2
```

Et utiliser cet alias ainsi :

```
-cape+armure
```

Les variables en SharpScript sont davantage expliquées dans la documentation consacrée au SharpScript.

Fonctionnalités basiques de CocoMUD

CocoMUD est un client MUD conçu pour être accessible et agréable avec un lecteur d'écran. Il supporte nativement un système de TTS (Text-to-Speech) pour la plupart des lecteurs d'écran sous Windows, ainsi que la gestion de plages tactiles.

Ce document décrit les fonctionnalités basiques de CocoMUD, savoir ce qu'il permet et comment l'utiliser, avec des exemples. Certaines fonctionnalités plus complexes sont décrites dans des documents à part.

À l'ouverture de CocoMUD

Quand vous ouvrez CocoMUD, vous devriez voir une liste de serveurs configurés (ou d'univers, dans le vocabulaire du client). C'est une liste, vous pouvez sélectionner un autre univers avec les flèches ou vous déplacer plus rapidement en entrant les premières lettres de l'univers en question. Il est possible que le serveur de jeu auquel vous souhaitez vous connecter ne soit pas présent dans la configuration par défaut : pour cela, il vous faut l'ajouter avant de continuer.

Quand vous avez sélectionné l'univers auquel vous souhaitez vous connecter, appuyez sur Entrée (ou cliquez sur le bouton **se connecter**).

La fenêtre principale du client

La fenêtre principale de CocoMUD est aussi simple que possible. Elle ne contient qu'une seule zone de texte, dans laquelle vous pouvez vous déplacer en utilisant les flèches, sélectionner et copier du texte, vous déplacer au début ou à la fin de la fenêtre, etc.

Quand vous écrivez dans ce champ de texte, le curseur est automatiquement déplacé en bas de la fenêtre. Vous pouvez considérer cette zone de texte comme une zone en lecture seule, à l'exception de la dernière ligne. Vous pouvez envoyer votre commande en appuyant sur Entrée comme d'habitude. Vous pourrez ensuite remonter avec les flèches pour voir la réponse du serveur. Cette navigation peut sembler assez étrange de prime abord, mais elle devient très intuitive après quelques essais seulement.

Quand vous appuyez sur Entrée après avoir sélectionné un univers, CocoMUD essaye de se connecter au serveur. Si tout se passe bien, le client se connecte et le message d'accueil du jeu s'affiche dans la zone de texte (vous pouvez le lire en utilisant les flèches). Le message d'accueil devrait également être envoyé au lecteur d'écran (et, si vous avez une plage tactile, le message devrait apparaître dessus également). La plupart des MUD vous demanderont un nom d'utilisateur et un mot de passe.

Historique des commandes

Quand vous appuyez sur Entrée pour envoyer une commande, elle est ajoutée à votre historique de commandes. Vous pouvez naviguer dans cette historique de commande pour envoyer de nouveau une commande ou bien la modifier si nécessaire. Il existe deux façons d'utiliser l'historique des commandes :

- En utilisant CTRL + les flèches haut ou bas, vous pouvez naviguer dans l'historique des commandes. Si vous appuyez sur CTRL + flèche haut, le client affichera la commande que vous venez d'entrer. Vous pouvez l'envoyer de nouveau en appuyant sur Entrée, ou bien la modifier avant envoi si nécessaire. Appuyez sur CTRL + flèche bas pour descendre dans votre historique de commandes.
- En utilisant le mode verrouillé. Pour entrer dans ce mode, appuyez sur échappe. Vous pourrez maintenant naviguer avec les flèches haut et bas pour parcourir l'historique des commandes. Pour repasser en mode non-verrouillé et lire le résultats de vos commandes, appuyez sur échappe de nouveau.

Auto-complétion

CocoMUD propose une fonctionnalité d'auto-complétion. Quand vous commencez à écrire un mot, vous pouvez appuyer sur la touche de tabulation, le client essayera de trouver la fin du mot que vous étiez en train de taper, en se basant sur le texte que vous avez reçu du serveur. Par exemple, si vous écrivez "po" dans le client, puis appuyez sur la touche de tabulation, le client écrira "porte", "portail" ou "potion" ou autre, en fonction des mots les plus fréquents qu'il a reçu. Les mots apparaissant le plus souvent sont proposés en premier.

Quand vous appuyez sur la touche de tabulation, le mot est écrit juste avant le curseur. Vous pouvez continuer à écrire votre commande ou l'envoyer telle qu'elle. Si le mot trouvé par l'auto-complétion n'est pas le bon, vous pouvez de nouveau appuyer sur la touche de tabulation : le client cherchera un autre mot en se basant sur les premières lettres que vous aviez écrites. Vous pouvez appuyer sur la touche de tabulation autant de fois que nécessaire pour trouver votre mot : la sélection se base sur la fréquence (le plus souvent un mot apparaît, les meilleures chances il a d'être choisi pour l'auto-complétion). Si le client ne trouve plus de résultat, il vous remettra les lettres initialement écrites et vous devrez finir le mot vous-même.

Commandes multiples

L'envoi de commandes multiples (command stacking en anglais) est une fonctionnalité permettant d'envoyer plusieurs commandes à la fois. Par exemple :

```
say sounds;say good
```

Cette syntaxe est identique à :

```
say sounds
say good
```

Envoyer plusieurs commandes à la fois peut être utile dans certains cas. Par défaut, le délimiteur pour les commandes multiples est le point virgule (;), mais vous pouvez le modifier dans les options, onglet commandes. Si vous voulez retirer la fonctionnalité des commandes multiples, supprimez simplement le délimiteur dans les options.

Vous pouvez entrer le délimiteur plusieurs fois pour l'envoyer dans vos commandes. Si vous avez laissé le délimiteur par défaut (;) par exemple :

```
say une commande;say une autre commande avec un clin d'oeil ;;)
```

Cette commande enverra :

```
say une commande
say une autre commande avec un clin d'oeil ;)
```

Quand le délimiteur est présent plus d'une fois, il est ignoré par le système des commandes multiples. Ainsi :

Commande	Équivalent
say ;;)	say ;)
say ;;;)	say ;;)
say ;;;;)	say ;;;)

La barre de menu

Certaines options sont accessibles dans la barre de menu. Dans Fichier -> Options, vous verrez plusieurs options que vous pouvez modifier pour rendre votre expérience avec CocoMUD plus agréable.

Les options

Quand vous sélectionnez Fichier -> Options dans la barre de menu, ou entrez *Alt + Entrée*, vous devriez voir une boîte de dialogue avec plusieurs onglets. L'onglet sélectionné par défaut est **général**. Pour l'heure, il ne contient que le réglage de la langue de CocoMUD. Le client devrait être dans la langue de votre système, si une traduction existe dans le logiciel, ou bien anglais si ce n'est pas le cas.

Dans l'onglet **affichage**, vous pouvez sélectionner un encodage différent. Par défaut, CocoMUD utilise un encodage Latin, mais vous pouvez le changer si besoin est.

Dans l'onglet **commandes**, vous pouvez modifier le caractère utilisé pour l'envoi de commandes multiples (command stacking en anglais).

Dans l'onglet **accessibilité** se trouvent plusieurs options qui modifient le comportement d'accessibilité du client. Les voici plus en détail :

- Activer le TTS (Text-to-Speech): vous pouvez ici désactiver le TTS (Text-to-Speech). Le TTS permet d'envoyer au lecteur d'écran les messages reçus par le serveur. Parfois, avoir le TTS actif n'est pas utile, et vous pouvez le désactiver ici.
- Activer le TTS hors de la fenêtre: par défaut, CocoMUD active le TTS même si vous vous trouvez dans une autre fenêtre. Ainsi, vous pourriez être sur le web, ou un livre, ou envoyer un e-mail, quand le lecteur d'écran se met à parler pour vous avertir de messages reçus sur le MUD auquel vous êtes connectés. Cela peut être utile, mais cela peut aussi être problématique, et vous pouvez le désactiver ici.

Ajouter un univers

Si vous voulez ajouter un univers (un serveur de jeu), dans la sélection des univers (à l'ouverture du client), cliquez sur le bouton **Ajouter**. Une boîte de dialogue s'ouvrira, vous demandant trois informations :

- Le nom de l'univers : quel nom donner à ce serveur ? En général, ce sera le nom du MUD, comme VanciaMUD ou MultiMUD.
- Le nom d'hôte du serveur.
- Le numéro de port du serveur.

Le nom de l'univers, nom d'hôte du serveur et numéro de port peuvent être modifiés après coup. Cependant, le nom du serveur est celui donné à son dossier, où les options de connexion et autres informations seront conservées. Changer le nom ne changera pas le nom du dossier.

Une fois créé, vous verrez l'univers dans la liste et pourrez vous y connecter.

h1. Quoi de neuf dans les récents builds

Cette page décrit les changements apportés par chaque build. Vous pouvez naviguer entre builds en utilisant la navigation par titre.

Build 49

- CocoMUD est à présent disponible en espagnol ! Un grand merci au contributeur qui a traduit CocoMUD dans cette langue et a participé à d'autres demandes pendant cette version.
- CocoMUD tourne à présent sous Python 3, à la place de Python 2 (#128). Quelques erreurs mineures de compatibilité pourraient persister, n'hésitez pas à les signaler si vous en rencontrez.
- Les macros et d'autres éléments de configuration sont maintenant correctement liés à une session, pas à un univers. Il n'y a donc plus de conflit si vous ouvrez plusieurs onglets sur le même univers (#136).
- CocoMUD peut maintenant jouer des fichiers audio dans les format MP3, OGG et WAV sans distinction. L'ancienne dépendance sur pygame a été rejetée, CocoMUD utilise à présent une bibliothèque audio bien plus légère et efficace (#134).
- La console est à présent masquée quand CocoMUD se met à jour (#135). Cela fait parti d'un effort pour rendre la mise à jour plus agréable.
- La console Python a été retirée du menu **Fichier**. À la place, un nouveau menu **Outils** a été ajouté la contenant.
- Une console SharpScript a été ajoutée dans le menu **Outils**. Si vous avez utilisé CocoMUD depuis le début, vous vous souviendrez peut-être qu'il était possible d'entrer des instructions SharpScript (comme #say) directement dans le client. Cette option a été retirée car elle créait trop de conflit avec certains jeux. Vous pouvez maintenant entrer les exemples de configuration rapide dans cette console (#138).
- CocoMUD prend à présent un argument de ligne de commande optionnel, permettant de spécifier un dossier de configuration différent. Par défaut, CocoMUD conserve ses données dans le répertoire courant (celui où se trouve **cocomud.exe**). On peut maintenant lui préciser un autre répertoire. Cette option ne sera sans doute pas utilisée par la plupart des utilisateurs, mais elle est bien pratique pour les développeurs (#139). Exemple d'utilisation : cocomud.exe --config-dir=D:\CocoMUD
- Le compteur de notifications non lues augmentait systématiquement même si l'information reçue était dissimulée par un trigger muet. Cela a été corrigé (#127).
- CocoMUD plantait à l'ouverture si aucune connexion Internet n'était disponible. Cela a été corrigé (#132).
- Une régression a été corrigée, interdisant de créer et charger des personnages (#141). Notez également qu'afin de rendre CocoMUD plus facile à utiliser pour une population internationale croissante, la configuration des univers et mondes (principalement les fichiers config.set pour commencer) seront convertis dans l'encodage utf-8. Cette opération se fait automatiquement et ne nécessite généralement aucune action de votre part.
- Les erreurs générées dans la console Python n'étaient pas correctement affichées. Cela a été corrigé (#142).
- Les couleurs en jeu n'étaient pas toujours affichées. Cela a été corrigé et optimisé (#144).
- Note pour les développeurs : CocoMUD utilise maintenant pipenv pour le support de ses dépendances, ce qui permet une installation depuis les sources bien plus simple. Une documentation plus détaillée sera fournie prochainement.

Build 48

- Une option dans la barre de menu (**Jeu** -> **Canaux**) permet d'éditer les canaux. Vous trouverez la documentation sur les canaux ici (#120).
- Une nouvelle fonction SharpScript #randplay a été ajoutée, permettant de jouer un son aléatoirement choisi dans une liste. La fonction est utilisable, mais elle n'est pas encore présente dans l'interface du client.
- Le travail pour supporter les options globales a été fait et sera probablement disponible lors de la prochaine mise à jour.
- On peut maintenant vider la fenêtre de contenu grâce à l'option dans la barre de menu **Jeu** -> **Nettoyer la fenêtre de contenu** (#64).
- Plusieurs correctifs pour rendre l'importation d'univers depuis le site web de nouveau possible (#123).
- Correction de l'erreur en modifiant les préférences (#125).

Build 47

- Vous pouvez simplement exporter un univers dans un fichier ZIP, en choisissant quoi exporter (**Fichier** -> **Exporter cet univers**). Vous pouvez ensuite partager ce fichier ZIP avec d'autres utilisateurs qui pourront l'importer depuis **Fichier** -> **Importer un univers** (#86).
- CocoMUD gère mieux le contenu du presse-papier, en particulier sous Linux (#106). Une case à cocher a été ajoutée dans les options pour désactiver l'envoi automatique du texte quand on colle depuis le presse-papier (#91).
- CocoMUD supporte et affiche les fichiers au format **WAV** et **OGG** pour jouer des sons (#112).
- Le rich-text a été désactivé par défaut pour offrir une expérience utilisateur plus fluide pour les nouveaux utilisateurs (#114).
- Plusieurs améliorations et corrections de bug pour Linux (#105, #106).
- CocoMUD ouvre et enregistre les préférences utilisateurs convenablement même si la langue de l'utilisateur n'est pas supportée par le client (#108).
- Importer un univers depuis le site web ne génère plus d'erreur d'encodage (#117).
- CocoMUD affiche à présent le texte utilisant une police de caractère plus agréable (merci encore aux contributeurs voyants).
- Note : pour des raisons de compatibilité, l'encodage des fichiers de configuration a été modifié. CocoMUD devrait supporter les anciens fichiers de configuration et les convertir à l'ouverture, mais il est possible que certaines erreurs surviennent avec des

fichiers contenant de nombreux caractères spéciaux.

Build 46

- Les commandes multiples fonctionnent de nouveau et sont à présent systématiquement testées (#98).
- Les commandes multiples supportent maintenant les caractères spéciaux (#99).
- CocoMUD gère mieux les variables dont les valeurs contiennent des caractères spéciaux(#93).

Build 45

- CocoMUD ne fait plus planter le lecteur d'écran en bloquant la fenêtre (#96).
- CocoMUD supporte maintenant le protocole SSL, permettant de crypter la connexion telnet (#94).
  SSL n'est pas proposé par tous les jeux, et est configuré sur un port différent du protocole telnet.
- Ajoute une option dans la barre de menu pour désactiver tous les sons du client comme ceux générés par des triggers (#89).

Build 44

- Ajoute une action SharpScript pour répéter la dernière commande entrée (#85) :
  Cette action, appelée #repeat, peut être utilisée pour répéter la dernière commande entrée ou envoyer une commande plusieurs fois.
- Quand un raccourci associé à un macro est utilisé, le curseur va automatiquement en bas de la fenêtre (#80).
- Corrige plusieurs conflits quand un univers était ouvert plusieurs fois dans différents onglets (#90).

Build 43

- Ajout d'une page de documentation pour télécharger et installer CocoMUD.
- Ajout de raccourcis clavier pour ajouter, éditer ou supprimer dans toutes les boîtes de dialogue (#87).

Build 42

- Ajoute les triggers avec substitution (#79) :
  Les triggers peuvent maintenant remplacer une ou plusieurs lignes directement dans le client, quand ils s'exécutent.
- Mise à jour des couleurs et du style de la fenêtre en mode rich-text.
- Corrige un bug avec le curseur instable quand plusieurs onglets sont ouverts (#53).

Build 41

- Ajout des personnages par défaut (#77) :
  Dans la boîte de dialogue pour configurer un personnage, on peut maintenant préciser si ce personnage doit être chargé par défaut. Si oui, à la connexion, en choisissant un univers dans la liste, CocoMUD sélectionne automatiquement ce personnage. Cela permet de gagner du temps si l'on se connecte souvent au même personnage d'un univers, et n'empêche pas de se connecter à d'autres.
- Quand on crée un personnage avec des caractères spéciaux (comme des lettres accentuées), CocoMUD ne plante plus au démarrage (#78).
  Un merci tout particulier au contributeur qui a signalé ce bug sur le site (#76).
- Quand CocoMUD perd la connexion au serveur, il essaye de se reconnecter, entrant le nom d'utilisateur / mot de passe si configuré pour (#67) :
  Cette correction n'est pas parfaite, il est difficile de gérer les erreurs de connexion.
- Quand aucune mise à jour n'est disponible, le message s'affiche correctement (#75).

Build 40

- Ajoute une option pour désactiver le "rich text" dans le client (#82).
  Cette option se trouve dans le menu **Fichier** -> **Options**, onglet **Accessibilité**. Désactiver le "rich text" est parfois utile pour l'accessibilité, bien que cela retire toute couleur du client.
- Créer un univers et fermer la boîte de dialogue ne crée plus d'erreur (#69).
  Un merci tout particulier au testeur qui a reporté ce bug (#68).
- Corrige un bug quand on ferme tous les onglets de CocoMUD.

Build 39

- Ajoute un bouton pour importer un univers depuis l'écran de connexion (#60):
  Il est désorlmais possible d'importer un univers avant de se connecter (ce qui est assez logique la plupart du temps). Cliquez

simplement sur le bouton **Importer** sur l'écran de connexion, et choisissez si vous désirez importer l'univers depuis un fichier ou depuis le site Web.
- Affiche une boîte de dialogue pour confirmer que l'installation d'un univers s'est bien déroulée (#63).
- Installer un univers avec canaux n'ouvre plus de boîtes de dialogues secondaires (#81).
- Redémarrer CocoMUD après avoir installé un univers n'est plus nécessaire.

Build 38

- Ajout de bloc-notes pour chaque univers et chaque personnage (#62):
  CocoMUD permet désormais d'accéder à des bloc-notes indépendants, des fichiers de texte que vous pouvez utiliser pour conserver certains points de repère, des informations de quête ou autre. Il y a un bloc-notes différent pour chaque univers, que vous pouvez ouvrir via la barre de menu -> **Jeu** -> **Bloc-notes** -> **Pour cet univers...**. Vous pouvez écrire du texte dans ce bloc-notes, appuyer sur échappe pour le fermer et l'enregistrer. Le second bloc-notes fonctionne selon le même principe, et est accessible via la barre de menu **Jeu** -> **Bloc-notes** -> **Pour ce personnage...**. Ce second bloc-notes ne sera pas partagé entre les personnages de cet univers.
- Ajoute une documentation pour les macros.
- Ajoute les triggers marqués dans la boîte de dialogue des triggers (#30).
- Autorise les triggers sans action, particulièrement utile pour les triggers marqués.
- Corrige plusieurs bugs d'affichage dans l'interface.

Build 37

- Ajout une fonctionnalité pour créer des personnages (#61) :
  On peut désormais configurer un univers avec plusieurs personnages. Un personnage possède des informations de configuration plus spécifiques (comme des alias, macros ou triggers qui lui sont propres), mais il possède surtout des informations de connexion (un nom d'utilisateur et mot de passe, par exemple). Ces informations sont conservées dans un fichier crypté.
  Pour créer un nouveau personnage, dans la boîte de dialogue de connexion, sélectionnez l'univers auquel vous souhaitez vous connecer, puis faites Tab pour voir la liste des personnages associés à cet univers. Par défaut, il ne devrait y en avoir qu'un seul, appelé "aucun". Validez sur ce personnage (ou cliquez sur le bouton **se connecter**). CocoMUD vous connecte à un univers sans personnage. Pour créer un personnage, rendez-vous dans la barre de menu, **jeu** -> **Changer les options de ce personnage...**. Précisez dans cette boîte de dialogue le nom du personnage. Vous pouvez préciser les commandes à entrer avant le mot de passe (généralement un nom d'utilisateur), le mot de passe-même et les commandes à entrer après le mot de passe (si il y en). Cliquez sur **OK** pour sauvegarder le personnage dans un fichier crypté. À la prochaine connexion, vous verrez ce personnage dans la liste (choisissez l'univers, faites Tab pour vous retrouver dans la liste des personnages associées à cet univers). Les commandes que vous avez entrées dans la boîte de dialogue seront envoyées au serveur de jeu dès l'ouverture de la session, pour vous connecter plus rapidement. Elles seront de nouveau envoyées si vous souhaitez vous reconnecter.
- Corrige un bug empêchant de créer un univers (#66).
- Ajout de la boîte de dialogue en cas de crash :
  Cette boîte de dialogue apparaît quand une erreur imprévue survient dans le client. Elle vous donne plus d'informations sur l'origine du bug, ainsi que des explications sur la façon de rapporter ce bug à l'équipe des développeurs.

Build 36

- Ajoute les triggers marqués (#30).
  Une marque peut être placée sur un trigger, ce qui déplace le curseur automatiquement sur la ligne reçue ayant déclenché le trigger. Cela peut être utile quand vous recevez un nouveau message, si un trigger y est associé. Cela peut aussi être utile pendant l'exploration, pour déplacer directement le curseur sur la liste des sorties, par exemple.
- Met à jour la fenêtre visuellement pour essayer de la rendre plus agréable.
- Ajoute la gestion des couleurs ANSI dans le client (#65).

Build 35

- Ajoute la fonctionnalité pour importer un univers depuis un fichier, ou depuis Internet (#60).
  Dans le menu **fichier**, il y a un nouveau menu "importer un univers". Dedans se trouve deux sous-menus : le premier sert à importer un univers depuis un fichier, le second depuis Internet. Ce dernier regardera les univers configurés sur le site web du projet. Vous pourrez télécharger l'un d'entre eux depuis cette interface.

Build 34

- Ajoute un système de canaux (#50):
  Les canaux sont utiles pour surveiller plusieurs évènements. Ils sont surtout utiles pour conserver les messages de canaux de communication sur un jeu. Grâce au système de triggers, il est possible d'envoyer les messages à un canal spécifique. Grâce aux macros ou alias, il est possible d'afficher cette liste dans une boîte de dialogue séparée.
- Dans les préférences, onglet **accessibilité**, il est maintenant possible de configurer l'interruption du TTS si de nouveaux

messages arrivent pendant la lecture (#40).
- Quand les options du TTS sont modifiées dans les préférences, les changements sont appliqués instantanément (#55).
- Corrige quelques bugs dans le moteur SharpScript.

Build 33

- Les alias/macros/triggers décrits sur plusieurs lignes marchent de nouveau (#63).
- On peut maintenant écrire des instructions SharpScript dans des macros (#59).
- Les variables sont à présent expliquées dans la documentation des alias.
- Le titre de la fenêtre quand plusieurs onglets sont ouverts est mis à jour correctement (#51).

Build 32

- Ajoute une syntaxe pour écrire des variables en SharpScript.
  La syntaxe est $variable. Les variables ont été ajoutées aux alias (#45) et aux triggers (#44).
- Corrige un bug mineur dans les commandes multiples.
- Ajoute les triggers muets, très utiles pour sonoriser un prompt.

Build 31

- Améliore le système de logging.
- Met à jour les catalogues utilisés pour la traduction (#41).

Build 30

- Ajout d'un système de logging pour débug les évènements automatiques.

Build 29

- Corrige un bug des triggers sonores quand plusieurs univers sont ouverts en même temps (#48).

Build 28

- Ajout des commandes multiples (command stacking en anglais), pour envoyer plusieurs commandes à la fois, en utilisant le point virgule ou un autre caractère (#32).

Build 27

- Permet d'ouvrir plusieurs univers dans des onglets (#42) :
  Il est maintenant possible d'ouvrir plusieurs univers dans des onglets séparés, ou même d'ouvrir le même univers plusieurs fois. Dans le menu fichier se trouve à présent trois options pour créer un univers, ouvrir un univers dans un onglet différent et fermer l'onglet actuel. Pour naviguer entre les onglets, utilisez Ctrl + tab et Ctrl + Shift + tab comme d'habitude.
  La fonctionnalité permettant de changer le titre de la fenêtre si des messages non lus sont reçus par le client (#20) prend maintenant en compte l'onglet sélectionné uniquement.
- Ajout de menus pour se déconnecter et se reconnecter à un univers (#43):
  Un nouveau menu dans la barre de menu, appelé connexion, permet de se déconnecter ou se reconnecter à l'univers actuellement sélectionné.
- Le client ne lag plus quand il essaye de se connecter à un serveur distant (#21).

Build 26

- Quand l'utilisateur ne se trouve pas dans la fenêtre du client, si des messages sont reçus, le titre de la fenêtre change pour avertir l'utilisateur que des notifications l'attendent sur le client.

Build 25

- Ajout de l'historique des commandes
  L'historique des commandes se souvient de chaque commande que vous avez entrée. Vous pouvez l'utiliser en appuyant sur CTRL + flèche haut ou bas pour naviguer dans vos commandes entrées. Vous pouvez aussi utiliser l'historique de commande en mode verrouillé. Pour ce faire, appuyez sur échappe. Vous pourrez ensuite naviguer dans l'historique en utilisant les flèches. Repassez en mode non-verrouillé en appuyant sur échappe de nouveau.
- Vous pouvez maintenant coller plusieurs lignes dans le client pour envoyer plusieurs commandes d'un coup (#27).

Build 24

- Met à jour la documentation des fonctionnalités basiques (#36).
- Retire les options obsolètes se basant sur une zone d'entrée qui n'existe plus (#35).
- Quand un utilisateur appuie sur la touche de tabulation pour l'auto-complétion, le TTS prononce (et affiche) le résultat.

Build 23

- Ajout de l'auto-complétion (#34):
  Quand le client reçoit des messages du serveur de jeu, tous les mots contenus dans ces messages sont classés par fréquence (du plus fréquent au moins fréquent). Si vous commencez à écrire une lettre ou plusieurs, puis appuyez sur la touche de tabulation, le client cherchera à finir le mot que vous aviez commencé à écrire. Si le résultat trouvé par le client ne vous convient pas, vous pouvez appuyer de nouveau sur la touche de tabulation jusqu'à trouver le mot que vous vouliez entrer.

Les canaux dans CocoMUD

CocoMUD offre la fonctionnalité des canaux. Un canal CocoMUD est, tout simplement, une liste de messages. Vous pouvez alimenter cette liste comme bon vous semble. Il est courant d'associer un ou plusieurs canaux du jeu avec un ou plusieurs canaux CocoMUD : quand un message est reçu provenant de l'un de ces canaux (comme public ou hrp), il est capturé automatiquement dans un canal CocoMUD. Un raccourci clavier peut être associé à un canal pour afficher la liste des messages qu'il contient. Cette fonctionnalité s'avère très pratique en cas d'exploration ou de combat dans le jeu, quand on ne souhaite pas nécessairement être interrompu par ce type de messages mais désirons les conserver pour les lire plus tard.

Nous utiliserons cet exemple tout au long de la documentation : essayons de capturer le contenu du canal hrp dans un canal CocoMUD. Cet exemple est le même que celui donné pour les triggers, ce qui n'est évidemment pas une coïncidence.

Créer un nouveau canal

La première chose à faire est de créer un canal CocoMUD, une liste vide qui pourra recevoir certains messages. Pour l'exemple, nous allons créer un canal "hrp". Souvenez-vous, les canaux CocoMUD peuvent contenir tout type d'information, bien qu'ils soient souvent utilisés pour capturer des canaux du jeu.

Dans la barre de menu du client, sélectionnez **Jeu** -> **Canaux**. Vous vous retrouvez dans une boîte de dialogue listant les canaux existants. Cette liste est probablement vide pour l'instant. Tabulez jusqu'à trouver le bouton **Ajouter** et validez. Le client vous demande le nom du canal à créer. Un nom court et explicite est préférable : ici, nous l'appellerons hrp .

On peut maintenant envoyer des messages à ce canal en utilisant [[Triggers|des triggers]. Si vous n'êtes pas familier du concept des triggers, il est recommandé de lire la documentation des triggers avant de continuer.

Nourrir un canal avec des triggers

Premier réflexe quand il s'agit de créer un trigger, regardons les lignes qui devraient le déclencher. Il y en a deux dans notre cas :

Quand on envoie un message sur le canal "hrp", on pourrait voir quelque chose comme :

```
[hrp] Vous dites : mon message
```

Si c'est quelqu'un d'autre qui parle sur ce canal, vous pourriez voir :

```
[hrp] Quelqu'un dit : mon message
```

Voyons ce qui reste identique et ce qui diffère entre ces deux lignes :

- Et bien, d'abord, on a "[hrp]"(c'est-à-dire "hrp"entouré de crochets) suivi d'un espace ;
- Ensuite il y a le nom de l'auteur du message, soit "Vous" soit "Quelqu'un" ;
- Il y a ensuite le mot "dit" après un nouvel espace... ha bien non, si on écrit, le mot est "dites", si c'est un autre, c'est "dit". Que faire ?
- Un nouvel espace, un signe deux points (:), un espace et le message.

Pour résoudre notre problème entre dit/dites, on a deux solutions :

- On peutécrire un seul trigger qui se déclenche dans les deux cas, en incluant dans le déclencheur "dit*". Cela voudra dire "dit" suivi de n'importe quoi ;
- On peut aussi créer deux triggers distincts, l'un pour la première ligne et l'un pour la seconde.

J'ai tendance à préférer la seconde solution, mais je sais que certains préféreront la première, Nous verrons donc les deux ici :

Un seul trigger

Voyons notre déclencheur pour un seul trigger. Vous êtes prêt ?

```
[hrp] * dit*: *
```

Trois astérisques ! Le premier capturera le nom de l'auteur du message. Le second capturera rien ou "es" (en fonction de si c'est vous ou un autre qui parle). Nous ne l'utiliseront pas. Le troisième capturera le message.

Pour créer ce trigger en passant par l'interface :

- Ouvrez la barre de menu, **Jeu** -> **Triggers** ;
- Cliquez sur **Ajouter** pour créer un trigger ;
- Entrez le déclencheur [hrp] * dit*: * avant d'appuyer sur Tab ;
- Sélectionnez "Envoie un message dans un canal" ;
- Cliquez sur le bouton **Ajouter l'action** ;
- Dans le nom du canal dans lequel envoyer le message, écrivez hrp ;
- Dans le message à envoyer au canal, entrez :

```
$1: $3
```

- Cliquez sur **OK** plusieurs fois pour fermer la boîte de dialogue en sauvegardant.

  Qu'est-ce que c'est que $1 $3?

$1 contient l'auteur du message (vous ou quelqu'un). $3 contient le message envoyé. Quand on reçoit la ligne :

```
[hrp] Jamie dit : bien joué !
```

Notre canal CocoMUD devrait recevoir le message :

```
Jamie: bien joué !
```

Vous pouvez entrer #channel hrp pour voir la liste des messages sur ce canal. Il est possible de relier cette action à un macro, pour afficher le canal quand on presse une touche de raccourci.

Vous pourriez avoir fait la même chose en utilisant une seule ligne de SharpScript:

```
#trigger {[hrp] * dit*: *} {#feed hrp {$1: $3}}
```

C'est peut-être un peu plus dur à comprendre en SharpScript, mais si vous êtes habitué à la syntaxe, c'est définitivement plus rapide.

Un dernier mot concernant ce trigger : vous aurez peut-être remarqué qu'on ne joue pas de son si ce trigger se déclenche. Rien ne vous empêche de le faire au travers de l'interface (d'avoir deux actions liés à un trigger). La même chose peut se faire en SharpScript :

```
#trigger {[hrp] * dit*: *} {
    #feed hrp {$1: $3}
    #play sounds/public.wav
}
```

Deux triggers distincts

Comme dit plus haut, il y a deux façons de régler le problème posé par dit/dites. L'une de ces solutions est de créer deux triggers, l'un pour quand on est l'auteur, l'autre pour les autres. Cela demandera un peu plus de configuration, mais cela reste plus lisible, je trouve. Parfois il sera impossible de faire autrement, cela dépend de la langue dans laquelle le MUD se trouve.

Nos deux déclencheurs seraient donc :

```
[hrp] Vous dites : *
[hrp] * dit : *
```

Je ne peux pas dire pour vous, mais ça me semble personnellement bien plus clair. Le premier trigger ne se déclenche que quand on est l'auteur du message (peut-être qu'on ne jouera pas de son à ce moment), le second se déclenche quand c'est quelqu'un d'autre qui parle sur le canal. Voici les instructions SharpScript pour créer ces triggers, mais comme d'habitude, vous pouvez le faire via l'interface :

```
#trigger {[hrp] Vous dites : *} {#feed hrp {Vous : $1}}
#trigger {[hrp] * dit : *} {
    #feed hrp {$1 : $2}
    #play sounds/public.wav
}
```

Choisissez la méthode qui vous para^8it la plus simple, en fonction du contexte.

Associer un macro à un canal

Envoyer des messages à un canal est certainement utile, sous réserve de pouvoir l'afficher. La plupart du temps, on crée un macro pour afficher le contenu du canal. C'est assez facile à faire :

Commencez par créer un macro comme d'habitude. Dans l'interface du client :

- Ouvrez la barre de menu **Jeu** -> **Macros**.
- Cliquez sur **Ajouter** pour ajouter un nouveau macro.
- Appuyez sur les touches à associer à ce macro. Par exemple : CTRL + H.
- Tabulez deux fois pour trouver la liste des actions possibles. Dans cette liste, vous devriez trouver l'action "Crée ou affiche un canal".
- Tabulez une fois pour ajouter cette action. Le client vous demande le nom du canal à afficher. Entrez hrp dans cet exemple.
- Fermez ces boîtes de dialogue en appuyant plusieurs fois sur **OK** pour enregistrer. Vous pouvez à présenter entrer le raccourci CTRL + H pour afficher le canal HRP. Victoire !

Supprimer un canal

On peut supprimer un canal en utilisant la même interface :

- Ouvrez la barre de menu **Jeu** -> **Canaux**.
- Tabulez jusqu'à trouver le bouton **Supprimer**. Le client vous demandera quel canal supprimer. Sélectionnez-en un, tabulez de nouveau pour le supprimer. Si vous validez en appuyant sur **OK** ensuite, le canal sera retiré de votre configuration.

Gardez à l'esprit cependant qu'un ou plusieurs triggers peuvent toujours essayer d'alimenter ce canal. Cela ne marchera pas, le canal n'existant plus, mais le trigger existera toujours car il pourrait faire d'autres choses.

Configuration des personnages dans CocoMUD

CocoMUD permet de configurer un ou plusieurs personnages par univers. Un personnage configuré offre plusieurs avantages :

- Se connecter par un personnage entrera automatiquement votre nom d'utilisateur, mot de passe et autres informations nécessaires.
- Le nom d'utilisateur et mot de passe sont cryptés par CocoMUD, ces informations n'apparaîtront pas en clair, ni dans votre historique de commande, ni dans l'un des fichiers de configuration de CocoMUD.
- Si vous utilisez une connexion sécurisée (comme SSL ou SSH) votre mot de passe sera protégé et invisible depuis votre ordinateur ou depuis le réseau.
- Vous pouvez avoir des bloc-notes CocoMUD séparés pour chacun de vos personnages.

Quand vous choisissez un univers auquel se connecter, vous pouvez choisir plus spécifiquement un des personnages configurés sur cet univers. Cependant, vous pourrez toujours, peu importe le nombre de personnages configurés, demander à CocoMUD de vous connecter au jeu sans utiliser un personnage.

Ajouter un personnage à un univers

Par défaut, vous n'avez aucun personnage configuré sur vos univers. Pour ajouter un personnage, il vous faut vous connecter à l'univers pour commencer. Quand vous ouvrez CocoMUD et que vous vous trouvez sur la liste des univers configurés, choisissez-en un et appuyez sur ENTRÉE pour s'y connecter.

CocoMUD essayera alors de se connecter à cet univers. Il n'entrera aucune information automatiquement. Pour créer un personnage, ouvrez la barre de menu, sélectionnez **Jeu** -> **Changer les options de ce personnage...**.

Une boîte de dialogue s'ouvrira alors, vous demandant plusieurs informations :

- **Nom** : le nom du personnage à ajouter. Cette information n'est utilisée que par CocoMUD et ne sera pas envoyée au serveur de jeu. Le nom entré sera affiché sur l'écran de connexion pour vous permettre de vous connecter plus rapidement (voir ci-dessous).
- **Nom d'utilisateur ou autres commandes à envoyer avant le mot de passe** : vous devez entrer ici la première commande (ou commandes) à entrer une fois connecté au jeu. En général, le jeu vous demande le nom d'utilisateur dès la connexion. Sur d'autres jeux, vous pourriez avoir d'autres commandes à entrer. N'écrivez pas votre mot de passe dans ce champ, entrez juste la ou les commandes à entrer avant votre mot de passe. Si vous avez plusieurs commandes à entrer, placez-les sur des lignes distinctes en appuyant sur ENTRÉE.
- **Mot de passe optionnel** : il vous faut ici préciser votre mot de passe (si il y en a un à envoyer au jeu pour vous connecter, ce qui est souvent le cas). Ce champ n'est pas seulement protégé (vous ne verrez que des signes * à la place de chaque lettre), le mot de passe sera crypté par CocoMUD et n'apparaîtra pas en clair dans la configuration du client.
- **Les commandes optionnelles à envoyer après le mot de passe** : sur la plupart des jeux, vous n'aurez que votre nom d'utilisateur et mot de passe à entrer pour vous connecter. D'autres attendront des informations supplémentaires, que vous pouvez entrer ici. De même que les commandes à envoyer avant le mot de passe, vous pouvez préciser plusieurs commandes en les séparant sur plusieurs lignes.
- **Positionner ce personnage comme choix par défaut** : si cette case est cochée, quand vous sélectionnerez l'univers pour vous y connecter, ce personnage sera automatiquement sélectionné. Vous ne pouvez avoir qu'un seul personnage par défaut configuré sur chaque univers.

Cliquez sur **OK** pour confirmer vos options. Vous pouvez ensuite fermer l'onglet et le ré-ouvrir : sur l'écran de connexion, sélectionnez l'univers avec les flèches. Appuyez sur la touche de tabulation: vous devriez vous retrouver sur une seconde liste, qui est la liste des personnages associés à cet univers. Le premier élément en haut de la liste est "aucun personnage", un choix qui vous permet de vous connecter au jeu sans entrer les commandes spécifiques à un personnage. Si vous avez configuré un personnage par défaut, le curseur sera sur ce choix automatiquement. Utilisez les flèches pour naviguer dans la liste des personnages et appuyez sur ENTRÉE pour se connecter à l'univers en utilisant ce personnage.

Quand vous vous déplacez dans la liste des univers (la liste sur laquelle vous vous trouvez quand CocoMUD s'ouvre), si vous sélectionnez un univers avec les flèches, son personnage par défaut sera automatiquement sélectionné. Nul besoin de faire **tab** pour le sélectionner. Vous pouvez appuyez sur ENTRÉE après avoir sélectionné l'univers si vous souhaitez vous connecter au personnage par défaut.

Changer la configuration du personnage

Si vous avez changé le mot de passe du personnage sur le jeu ou une autre information, vous pouvez vous connecter sur ce personnage et retourner dans la barre de menu, **Jeu** -> **Changer les options de ce personnage...**. La boîte de dialogue s'ouvrira de nouveau avec les options déjà renseignées pour ce personnage. Vous pouvez les modifier et cliquez sur **OK**.

Ajouter plusieurs personnages à un univers

Ajouter plusieurs personnages est assez semblable à ajouter un seul personnage : l'astuce ici est dans la façon de se connecter à l'univers. Vous devez sélectionner l'univers sur l'écran de connexion et tabuler une fois pour sélectionner "aucun personnage", ce qui veut dire qu'aucun personnage associé à cet univers ne sera choisi. De nouveau, rendez-vous dans la barre de menu, **Jeu** -> **Changer les options de ce personnage...**. La prochaine fois que vous vous trouverez sur l'écran de connexion, vous verrez que le nouveau personnage est bel et bien présent dans la liste. Souvenez-vous, vous ne pouvez avoir qu'un seul personnage par défaut configuré par univers. Tous les personnages sont affichés dans l'ordre alphabétique, vous pouvez aisément naviguer entre le personnage par défaut et d'autres personnages configurés.

Une note concernant la configuration

Pour l'instant, la configuration est seulement propre à l'univers. Cela signifie que tous les personnages configurés sur un univers partageront les mêmes alias, triggers ou canaux.

Télécharger et installer CocoMUD

CocoMUD est prêt à l'emplois sous Windows, dans une version portable. Vous n'avez pas besoin de l'installer sur votre système, juste le télécharger sous forme d'archive (voir les liens ci-dessous), l'extraire et lancer **cocomud.exe**. La procédure est détaillée dans les sections suivantes de ce document.

Télécharger CocoMUD

Build	Windows
49	CocoMUD pour Windows

Si vous utilisez un autre système d'exploitation que Windows, vous pourriez vouloir installer CocoMUD depuis les sources. CocoMUD fonctionne sous Linux et Mac OS, bien qu'il n'existe pas encore de version prête à l'emplois pour ces systèmes.

Si vous accédez à ces informations depuis une version déjà installée de CocoMUD, les liens donnés ci-dessus pourraient ne pas être à jour. Il vous est recommendé de vous rendre sur la documentation en ligne pour avoir les liens de téléchargement à jour.

Installer CocoMUD

Une fois l'archive téléchargée, vous devez l'extraire dans un dossier. CocoMUD n'a pas besoin de s'installer pour fonctionner, il tourne en version portable (vous pourriez même le mettre sur une clé USB pour l'emporter avec vous si vous voulez).

1. Commencez par extraire l'archive **CocoMUD.zip** dans le dossier que vous voulez. CocoMUD ne pourra pas se lancer directement depuis l'archive, et même si il y arrive, il ne pourra pas écrire dedans.
2. Dans le dossier où vous avez extrait l'archive, vous devriez trouver un dossier **CocoMUD**. Ouvrez-le.
3. Dedans se trouve un fichier **cocomud.exe** (parmi beaucoup d'autres). Pour lancer CocoMUD, il suffit de lancer ce fichier **cocomud.exe**. CocoMUD devrait s'ouvrir sur son écran de connexion.
4. Si vous appréciez CocoMUD, n'hésitez pas à faire un raccourci menant au fichier **cocomud.exe**, cela vous simplifiera la vie.

Pour apprendre à utiliser CocoMUD, vous devriez lire un aperçu des fonctionnalités basiques de CocoMUDD. Ce petit tutoriel est fait pour apprendre à utiliser les fonctionnalités les plus simples de CocoMUD. D'autres documents sont disponibles pour apprendre à utiliser les fonctionnalités plus avancées du client.

Mettre à jour CocoMUD

CocoMUD possède un updater intégré pour se mettre à jour automatiquement. Cet updater se lancera avec CocoMUD. Si il trouve une version plus récente de CocoMUD sur le site officiel, il vous proposera de l'installer. Si vous répondez "oui", CocoMUD sera refermé pendant que l'updater télécharge et installe la nouvelle version. Si tout va bien, CocoMUD s'ouvrira tout seul après la mise à jour.

Vous pouvez vérifier à tout moment si il y a des mises à jour. Dans le menu **Aide** de CocoMUD, choisissez **Vérifier les mises à jour disponibles**. La réponse pourrait se faire attendre quelques secondes, car CocoMUD regarde les mises à jour sur le site officiel. CocoMUD va vérifier si il existe des mises à jour et vous dire si il y en a. Si c'est le cas, il vous proposera de mettre à jour.

La mise à jour automatique pourrait ne pas s'exécuter correctement dans certains cas. Le pare-feu de Windows, Windows Defender, un logiciel antivirus ou un logiciel conçu pour protéger votre système pourraient élever des objections à la mise à jour automatique de CocoMUD.

Si la mise à jour rencontre des erreurs, CocoMUD pourrait devenir corrompu (c'est-à-dire que CocoMUD pourrait ne pas se lancer, ou bien générer de nombreuses erreurs au lancement). Si cela se produit, il y a plusieurs moyens de résoudre le problème :

1. Forcer la mise à jour : dans certains cas, CocoMUD essaye de se mettre à jour, mais certains fichiers ou dossiers ne sont pas modifiés. Un logiciel qui essaye de protéger votre système pourrait augmenter les risques qu'une telle situation se produise. Le pare-feu Windows et Windows defender ne semblent pas causer ce problème. Si vous cliquez sur **cocomud.exe** et que rien ne se passe, ou bien que CocoMUD se lance en générant de nmbreuses erreurs, vous pouvez forcer la mise à jour de CocoMUD. Pour ce faire, exécutez le fichier **dbg_updater.exe** dans votre dossier CocoMUD. **dbg** indique **debug**, et cette version de l'updater se veut aussi légère que possible. Même si il semble que votre version de CocoMUD soit à jour, le **dbg_updater** va télécharger et installer la version la plus récente depuis le site officiel, en gardant votre configuration. Ce fichier devrait ouvrir une console avec une barre de progression, mais vous n'avez pas besoin de vous en inquiéter : laissez-le travailler, télécharger et installer CocoMUD. Si tout va bien, CocoMUD sera ouvert automatiquement dans sa nouvelle version. Si cela ne change rien, essayez de lancer **dbg_updater** une nouvelle fois avant de passer à l'étape 2.
2. Mise à jour manuelle : parfois, CocoMUD est tellement corrompu que **dbg_updater** ne peut même pas se lancer, ou bien qu'il n'arrive pas à faire grand chose. Si cela se produit, vous devrez réinstaller CocoMUD manuellement. Téléchargez la version la

plus récente de CocoMUD (voir les liens ci-dessus). Décompressez l'archve dans un nouveau dossier. Pour conserver votre configuration, rendez-vous dans votre ancien dossier. Copiez les dossiers **settings** et **worlds** et placez-les dans votre nouveau dossier **CocoMUD**. Vous devriez ensuite pouvoir lancer CocoMUD normalement.

Si vous rencontrez toujours des erreurs, n'hésitez pas à contacter l'équipe des développeurs en créant un rapport de bug .

Nous vous souhaitons l'expérience de jeu la plus agréable possible avec CocoMUD !

Features

Commandes multiples

L'envoi de commandes multiples (command stacking en anglais) est une fonctionnalité permettant d'envoyer plusieurs commandes à la fois. Par exemple :

```
say sounds;say good
```

Cette syntaxe est identique à :

```
say sounds
say good
```

Envoyer plusieurs commandes à la fois peut être utile dans certains cas. Par défaut, le délimiteur pour les commandes multiples est le point virgule (;), mais vous pouvez le modifier dans les options, onglet commandes. Si vous voulez retirer la fonctionnalité des commandes multiples, supprimez simplement le délimiteur dans les options.

Vous pouvez entrer le délimiteur plusieurs fois pour l'envoyer dans vos commandes. Si vous avez laissé le délimiteur par défaut (;) par exemple :

```
say une commande;say une autre commande avec un clin d'oeil ;;)
```

Cette commande enverra :

```
say une commande
say une autre commande avec un clin d'oeil ;)
```

Quand le délimiteur est présent plus d'une fois, il est ignoré par le système des commandes multiples. Ainsi :

Commande	Équivalent
say ;;)	say ;)
say ;;;)	say ;;)
say ;;;;)	say ;;;)

Macros

CocoMUD propose un système de macros comme la plupart des autres clients MUD, qui permet de relier une touche de raccourci avec une action (comme une commande à envoyer au serveur. Ce document explique la création de macros simples et plus complexes.

Fonctionnalité basique

Dans sa version la plus simple, un macro fait le lien entre une touche de raccourci et une commande : par exemple, si vous appuyez sur Ctrl + F1, CocoMUD pourrait envoyer la commande "regarder" au serveur de jeu. Dans cette documentation, nous verrons comment créer un macro reliant la touche F1 avec la commande "nord".

Ajouter un macro

Il existe deux moyens d'ajouter un macro. L'interface étant l'alternative la plus simple, ce moyen sera décrit en premier.

Via l'interface

Dans la barre de menu, cliquez sur **Jeu** -> **Macro**. Une boîte de dialogue devrait s'ouvrir, contenant la liste des macros actuellement configurés. Cette liste sera probablement vide la première fois que vous ouvrirez cette interface. Cliquez sur le bouton **Ajouter**.

Une autre boîte de dialogue devrait s'ouvrir pour ajouter un nouveau macro. Le curseur devrait se trouver sur une zone de texte, mais il ne vous est pas demandé d'entrer du texte ici : à la place, appuyez sur la combinaison de touches que vous souhaitez associer à ce macro. Par exemple, appuyez sur la touche F1.

Dans la zone de texte devrait apparaître le nom du raccourci, "F1" dans notre cas. Vous pouvez utiliser de nombreuses touches de raccourci pour vos macros. Par exemple, appuyez sur la touche Ctrl, maintenez-la enfoncée, appuyez sur la touche I et relâchez les deux touches. Dans la zone de texte devrait apparaître :

```
Ctrl + I
```

Vous pouvez utiliser Shift, Ctrl ou Alt dans vos macros. En pratique, ce n'est pas conseillé d'utiliser Alt, qui est une touche réservée pour les menus (Alt + F ouvre le menu **fichier**, par exemple). Il y a malgré tout un grand nombre de raccourcis possibles, par exemple :

```
Shift + Backspace
Ctrl + F12
Ctrl + Shift + O
Ctrl + PavNum8
```

Si vous voulez connecter un macro à une touche du pavé numérique, vous devez activer ce dernier avant de cliquer sur le bouton **Ajouter**, ou bien naviguer dans une autre partie de la fenêtre pour ce faire. Sinon, CocoMUD va associer la touche du verrouillage de pavé numérique avec le macro, ce qui n'est pas souvent souhaitable.

Après avoir entré le raccourci voulu, appuyez sur la touche de tabulation. Le second champ de cette boîte de dialogue est également un champ de texte, celui-ci devant contenir la ou les commandes à envoyer au serveur de jeu quand on presse le raccourci renseigné précédemment. En poursuivant notre exemple, après avoir sélectionné **Ajouter**, appuyé sur la touche F1, puis sur la touche de tabulation, vous pouvez maintenant écrire "nord".

La zone suivante est une liste d'actions à associer à ce macro. Cette fonctionnalité sera détaillée plus bas, elle n'est pas nécessaire pour les macros simples. Tabulez jusqu'à trouver le bouton **OK** et validez. Vous devriez voir le macro que vous avez ajouté dans la liste.

Si vous voulez changer la ou les commandes à envoyer au serveur quand le macro est activé, vous devez l'éditer (en le sélectionnant dans la liste, et tabuler jusqu'à trouver le bouton **Editer**). Cependant, si vous souhaitez changer le raccourci associé à un macro, il vous suffit de le sélectionner et de tabuler une fois. Vous devriez voir une zone de texte (ou plutôt, une zone d'édition du raccourci). Appuyez sur la nouvelle combinaison de touches que vous souhaitez associer à ce macro, puis tabulez jusqu'à trouver le bouton **OK**. Si vous quittez cette boîte de dialogue sans cliquer sur **OK**, vos modifications seront perdues (cela inclut le macro que vous avez ajouté).

De retour dans la fenêtre principale de CocoMUD, appuyez sur F1. La commande "nord" est envoyée (silencieusement) au serveur de jeu qui devrait vous répondre, en fonction de l'univers. Pour moi, je reçois le message :

> Vous ne pouvez aller par là...

Via le SharpScript

Vous pouvez aussi ajouter un macro via une syntaxe du scripting SharpScript. En fait, c'est comme ça que vos triggers, macros et alias sont enregistrés dans votre fichier **config.set** (une configuration par univers).

Pour créer un macro, utilisez l'action #macro. Elle prend deux paramètres :

- Le raccourci à associer à ce macro ;
- La commande à envoyer au serveur quand on entre ce raccourci.

Dans notre exemple, on aurait pu créer notre macro en entrant l'instruction SharpScript suivante :

```
#macro F1 nord
```

Vous pouvez coller cette instruction directement dans CocoMUD et appuyez sur Entrée, comme pour envoyer une commande. CocoMUD voit que cette commande commence par un seul signe dièse et l'envoie à l'interpréteur SharpScript. Si vous ajoutez un macro de cette manière (et si vous n'avez pas fait d'erreurs de syntaxe), le macro sera visible instantanément dans la barre de menu **Jeu** -> **Macro**.

Si l'un des paramètres contient des espaces, n'oubliez pas de l'entourer d'accolades :

```
#macro {Ctrl + Shift + O} {dire c'est plutôt une longue commande.}
```

Les accolades ne sont absolument nécessaires que si le paramètre contient des espaces. Mais vous pouvez mettre des accolades autour des paramètres même si ils n'ont pas d'espaces, le moteur SharpScript les retirera sans faire d'histoire.

Editer un macro

Nous avons vu plus haut comment éditer un macro. Il n'y a rien de très compliqué, en vérité. L'important à retenir, c'est que si vous voulez changer le raccourci associé à un macro, vous devez juste le sélectionner et faire Tab pour entrer le nouveau raccourci. Si vous souhaitez changer la ou les commandes associées à ce macro, cependant, vous devez le sélectionner et cliquer sur le bouton **Editer**. La boîte de dialogue est identique à celle pour ajouter un macro, décrite précédemment.

Supprimer un macro

Supprimer un macro se fait des plus simplement. Dans la boîte de dialogue (menu **Jeu** -> **Macro**), sélectionnez le macro que vous voulez supprimer, et tabulez jusqu'à trouver le bouton **Supprimer**. Une confirmation de suppression sera affichée. N'oubliez pas de validez en cliquant sur le bouton **OK**, sans quoi le macro ne sera pas complètement supprimé.

Macros un peu plus complexes

Les macros relient souvent une touche de raccourci à une ou plusieurs commandes à envoyer au serveur. Cependant, vous pouvez aussi relier un raccourci clavier avec une action plus complexe, comme jouer un son, afficher un message (avec variables, éventuellement) ou afficher un canal. L'intérêt de telles action n'est pas forcément apparent tout d'abord. Un exemple de macro plus complexe peut se trouver dans l'utilisation des prompts sonores.

Le principe d'un prompt sonorisé est que la barre de prompt (contenant vos points de vie, magie et mouvement, par exemple) est interceptée par un trigger et n'apparaît plus. Votre vie, magie et mouvement actuels sont conservés dans des variables à part. Ceci dépasse le cadre de cette documentation, mais l'important est qu'un macro peut être créé pour afficher ces variables. Par exemple, si vous appuyez sur Ctrl + J, vous pourriez voir votre vie actuelle.

Nous avons utilisé la seconde zone de texte pour préciser quelles commandes envoyer au serveur de jeu quand on entre le raccourci indiqué. En vérité, on peut entrer des instructions SharpScript dans cette zone de texte. Mais puisque écrire les instructions SharpScript à la main n'est pas le plus facile, on peut passer par l'éditeur SharpScript. Cela n'est vraiment pas compliqué : dans l'interface d'ajout ou d'édition d'un macro, laissez le second champ de texte vide, et tabulez une fois pour vous trouver sur la liste des actions. Sélectionnez une action, par exemple "Affiche un message et l'envoie au lecteur d'écran". Cliquez sur le bouton **Ajouter l'action**. On vous demandera d'entrer d'autres informations (dans notre cas, le message à afficher, et des informations sur comment l'afficher). Validez une fois ces informations renseignées. N'oubliez pas d'appuyer sur **OK** pour fermer la boîte de dialogue en sauvegardant les modifications.

Les triggers dans CocoMUD

Il est possible que les triggers soient la fonctionnalité la plus puissante de tout client MUD. CocoMUD propose un système simple et flexible de triggers, qui sera présenté ici avec de nombreux exemples.

Qu'est-ce qu'un trigger ?

Normalement, si vous entrez une commande dans le client, le serveur devrait vous "répondre" en envoyant une ou plusieurs lignes. Cette réponse est constituée en grande partie (sinon en totalité, à l'exception des couleurs) de texte, tout comme votre commande.

Les triggers peuvent intercepter une ligne particulière et réagir en fonction, en faisant quelque chose. C'est l'expression la plus simple d'un trigger : je surveille le texte envoyé par le serveur, je réagis si je rencontre une certaine ligne et je fais quelque chose en retour.

Les triggers étant potentiellement puissants, leur configuration a été répartie dans différentes couches de complexité : vous n'avez pas besoin de comprendre (ni même de connaître l'existence) des types de triggers les plus avancés pour utiliser cette fonctionnalité. Cette documentation va décrire dans l'ordre la mise en place, pas à pas, des triggers les plus simples jusqu'aux triggers les plus complexes.

Créer un trigger

Commençons par quelque chose de simple : un trigger qui joue un son quand quelqu'un (n'importe qui) parle sur un canal du jeu.

Cet exemple sera différent sur différents MUDs, bien entendu, n'hésitez donc pas à l'adapter en fonction de vos besoins. Pour notre exemple, admettons que nous pouvons utiliser la commande "hrp" pour envoyer un message sur le canal "hrp", comme ceci :

```
hrp bonjour à vous !
```

Et si tout va bien, le MUD devrait répondre avec :

```
[hrp] Vous dites : bonjour à vous !
```

Ou bien, si quelqu'un d'autre parle sur le canal "hrp", vous pourriez voir un message comme ceci :

```
[hrp] Aaron dit : coucou
```

En bref, si nous voulons créer un trigger qui intercepte ces lignes, il nous faut un trigger qui intercepte les lignes qui commencent par "[hrp]" ("hrp" entouré de crochets).

> Pourquoi intercepter ces messages ?

Si vous êtes nouveau dans l'utilisation des triggers, vous pourriez vous demander à quoi ils servent. La réponse dans ce cas précis pourrait être que, quand on reçoit un message sur le canal "hrp", le client joue un son. C'est sur cela que nous allons travailler.

Nous allons donc voir comment créer un trigger qui :

- Réagit quand le client reçoit une ligne commençant par "[hrp]" ;
- Joue un son à ce moment.

Comme la plupart des fonctionnalités sur CocoMUD, on peut créer des triggers en passant par l'interface ou par une instruction SharpScript. La première étant la plus facile pour commencer, nous allons la voir d'abord.

Depuis l'interface

Pour ajouter, éditer ou suppriemr un trigger depuis l'interface de CocoMUD, ouvrez la barre de menu, **Jeu** -> **Triggers**.

Vous devriez vous trouver dans une boîte de dialogue listant les triggers actuellement configurés sur cet univers. Il est très possible que cette liste soit vide la première fois que vous ouvrez cette boîte de dialogue. Pour créer un trigger, cliquez sur le bouton **Ajouter**.

Une nouvelle boîte de dialogue s'ouvre alors. Le curseur devrait se trouver dans une zone d'édition, où l'on vous demande d'entrer le trigger (c'est-à-dire, son déclencheur, la partie qui va indiquer à CocoMUD quoi surveiller).

Pour notre exemple, nous avons déterminé que notre trigger devrait s'exécuter quand le client reçoit une ligne commençant par [hrp]. Vous pouvez écrire [hrp] dans cette zone de texte, mais ne la quittez pas encore : si vous créez un trigger déclenché par [hrp], le trigger ne sera exécuté que si le client envoie une ligne ne contenant **que** [hrp]. Ce n'est pas ce que l'on veut faire ici : nous voulons que le trigger réagisse à une ligne commençant par [hrp] . La solution est de mettre un signe astérisque (*) après {hrp}, pour dire au client que [hrp] peut être suivi de n'importe quoi.

Peut-être reconnaissez-vous cette syntaxe : c'est la même que pour créer des alias avec variables, et ce n'est pas une coïncidence, nous verrons plus tard pourquoi.

Pour l'instant, vous pouvez écrire dans ce champ de texte :

```
[hrp]*
```

Si vous appuyez sur Tab, vous devriez vous trouver dans une liste d'actions que l'on peut associer à ce trigger. Dans notre exemple, nous voudrions jouer un son quand ce trigger se déclenche : parcourez la liste jusqu'à trouver l'action :

```
Joue un son
```

Et cliquez sur le bouton suivant : **Ajouter l'action**.

On vous demande maintenant de configurer cette action (dans notre cas, choisir le fichier sonore à jouer quand ce trigger se déclenche). Vous avez un bouton "Parcourir" qui vous permet de trouver le fichier sonore sur votre disque dur. Le bouton "Test" permet de vérifier que CocoMUD parvient bien à lire et jouer ce fichier (CocoMUD lit les fichiers .wav et .ogg). Si tout se passe bien, cliquez sur **OK**. L'action sera ajoutée au trigger. vous vous trouverez dans la liste des actions liées à ce trigger :

```
#play mon_fichier.wav
```

Bien sûr, "mon_fichier.wav" sera remplacé par le chemin et le nom du fichier que vous avez sélectionné. La représentation de l'action est en version courte (version SharpScript), cela explique pourquoi vous voyez #play mon_fichier.wav. Ne vous en inquiétez pas trop, c'est surtout un point de repère pour vérifier que l'action a bien été ajoutée, ainsi qu'un moyen de l'éditer, si vous voulez par exemple jouer un autre fichier sonore à la place.

Ces deux listes et les quelques boutons qui l'entourent forment l'éditeur SharpScript, permettant de configurer des actions assez complexes sans toucher au SharpScript de près ou de loin. Cette fenêtre peut s'avérer un peu intimidante au premier abord, voici donc un résumé de ce qu'elle contient. Notez que vous trouverez la même hiérarchie pour configurer les alias ou les macros et d'autres fonctionnalités de CocoMUD dépendantes du SharpScript :

- La première chose à renseigner pour les triggers est le déclencheur, comme nous l'avons fait plus haut. C'est une zone de texte toute simple ;
- Au-dessous se trouve une liste d'actions connectés à ce trigger. C'est une liste, vous pouvez naviguer avec les flèches pour la parcourir ou sélectionner une action précise ;
- À droite se trouve le bouton pour éditer la ligne d'action sélectionnée (dans notre cas, nous pourrions vouloir changer le fichier sonore joué par le trigger) ;
- Toujours à droite se trouve le bouton pour supprimer la ligne d'action. Notez que ces trois champs (liste d'actions, bouton éditer et bouton supprimer) n'apparaissent pas si aucune action n'est associée à ce trigger ;
- Au-dessous se trouve une liste d'actions que l'on pourrait vouloir lier à ce trigger. C'est dans cette liste que nous avons sélectionné "joue un son" dans l'exemple ci-dessus. Elle apparaît dans tous les cas, car un trigger peut être connecté à aucune, une, deux ou de nombreuses actions, il n'y a pas vraiment de limite.
- À droite se trouve le bouton pour ajouter l'action sélectionnée.
- Le reste de la fenêtre contient d'autres cases à cocher et options qui seront détaillées dans la suite de cette documentation.

Nous avons créé notre trigger [hrp]* connecté à une action pour jouer un son : vous pouvez donc cliquer sur **OK**. Vous devriez vous retrouver dans la liste des triggers actuels, sur le trigger que vous venez d'ajouter. Appuyez de nouveau sur **OK** pour fermer la boîte de dialogue en sauvegardant. Si vous quittez la boîte de dialogue autrement, le trigger ne sera pas ajouté.

Pour vérifier que notre nouveau trigger marche correctement, essayons d'envoyer un message sur le canal "hrp" :

```
hrp Est-ce que ça marche ?
```

Si tout va bien (et partant du principe que le serveur répond comme on l'attend), on devrait recevoir la ligne :

```
[hrp] Vous dites : est-ce que ça marche ?
```

Et vous devriez entendre le son que vous avez sélectionné à l'étape précédente. Voilà ! Ce n'était pas si difficile, si ?

Depuis le SharpScript

Comme toujours, l'interface permet de manipuler des options potentiellement complexes, mais elles sont toutes converties en SharpScript à la fin, même si vous n'avez pas vraiment besoin de vous en occuper. Créer le trigger de notre exemple en SharpScript est assez facile : vous pouvez entrer cette ligne directement dans votre client, ou bien dans le fichier "config.set".

```
#trigger [hrp]* {#play mon_fichier.wav}
```

C'est une instruction SharpScript. De gauche à droite, nous avons :

- #trigger est le nom de l'action SharpScript. Ici, #trigger crée simplement un nouveau trigger ;
- [hrp]* est notre déclencheur, ce que le trigger doit surveiller quand on reçoit des messages du serveur. Tout comme audessus, on a précisé [hrp]* , qui veut dire "toute ligne commençant par [hrp]" ;
- Ensuite, on doit préciser la ou les actions à exécuter quand ce trigger se déclenche. Ici, #play mon_fichier.wav. #play est une autre action SharpScript, qui permet simplement de jouer le fichier passé en argument. Puisque notre argument contient un espace entre le nom de l'action #play et l'argument de la fonction mon_fichier.wav, on doit entourer l'argument d'accolades.

Ajouter un trigger en SharpScript est peut-être bien plus rapide, mais le système ne vous pardonnera pas facilement si vous faites des erreurs de syntaxe. L'interface nécessite d'avantage d'étapes, mais elle est généralement plus sûre.

Editer un trigger

Dans la boîte de dialogue des triggers (menu **Jeu** -> **Triggers**), vous pouvez éditer un trigger. Vous aurez besoin d'éditer un trigger dans deux cas : si vous voulez changer son déclencheur (la partie qui indique au trigger quel morceau de texte surveiller) on ses actions (la partie qui décrit quoi faire quand ce trigger se déclenche).

Supprimer un trigger

Supprimer un trigger peut se faire depuis l'interface également. Cliquez simplement sur le bouton **Supprimer** après avoir sélectionné un trigger existant. Confirmez que vous voulez supprimer ce trigger. N'oubliez pas de fermer la boîte de dialogue en cliquant sur **OK**, sans quoi votre trigger ne sera pas effacé.

Utiliser des variables dans les triggers

Il est tant de regarder plus attentivement le signe astérisque (*) que nous avons utilisé plus haut. Ce n'est pas un hasard si vous reconnaissez cette syntaxe depuis la documentation des alias, puisqu'il s'agit de la même chose : l'astérisque veut dire "tout et n'importe quoi".

Voici une liste des syntaxes possibles, pour vous donner une idée :

Syntaxe	Signifie

Bienvenue*	N'importe quelle ligne commençant par Bienvenue .
*classe	N'importe quelle ligne finissant par classe .
passage	N'importe quel ligne contenant (au début, à la fin ou au milieu) passage. Notez que ce trigger sera aussi déclenché si la ligne contient passager, par extension.
* passage *	N'importe quelle ligne contenant le mot passage entouré d'espaces. Le mot passager ne déclenchera plus ce trigger cette fois.
Vous gagnez * crédits en *.	Des lignes comme Vous gagnez 80 crédits en combat. ou Vous gagnez 10 crédits en management. déclencheront ce trigger. La ligne Vous gagnez un certain nombre de crédits en quelque chose. déclenchera le trigger également.

En bref, un signe astérisque veut dire n'importe quoi : une lettre, un chiffre, un mot, un nombre, un espace, un signe de ponctuation, un message assez long... voire rien du tout.

Un mot de mise en garde : la syntaxe de vos déclencheurs est très importante, et vous devriez vérifier avec soin les lignes que vous souhaitez intercepter.

```
*table*
```

Ce trigger se déclenchera quand le mot table se trouve dans une ligne, au début, au milieu ou à la fin. Ce trigger se déclencherait donc avec les lignes suivantes :

```
~ un tableau noir (commande board) se trouve ici
Dans l'angle nord-est de la cour se trouve un long bâtiment de bois peint en rouge, probablement u
ne étable.
```

Souvenez-vous qu'un trigger est autant facile à restrindre que facile à déclencher. Il vous faut trouver le bon équilibre entre les deux.

Retournons aux variables. Le signe astérisque (*) fait deux choses :

- Il aide à décrire quand le trigger doit se déclencher (il veut dire "n'importe quoi") ;
- Il écrit dans une ou plusieurs variables.

Utilisons le même exemple, avec notre trigger [hrp[*. Qu'arrive-t-il lorsque vous recevez une ligne comme celle-ci :

```
[hrp] Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ?
```

D'abord, le trigger [hrp]* est déclenché, puis la partie après [hrp] (celle décrite par le signe astérisque) est capturée dans une variable. Les variables permettent de conserver des informations, et c'est justement ce qu'elles font ici. Les variables sont numérotées en partant de $1, $2, $3 et ainsi de suite. Donc dans notre exemple, si on reçoit la ligne suivante, le trigger va créer une variable $1 contenant la partie après [hrp] :

```
 Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ?
```

Que peut-on faire avec cette variable ? Beaucoup de choses. Chaque paramètre de nos actions liées au trigger peuvent utiliser les variables du trigger. Nous verrons des exemples concrets un peu plus bas, mais pour l'heure, nous pouvons commencer par l'afficher. Vous pouvez entrer l'instruction suivante dans votre client :

```
#say $1
```

Qui devrait vous afficher :

```
Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ?
```

> Pourquoi la ligne commence par un espace ?

Si vous vous posez cette question, essayez d'écrire le déclencheur du trigger et la ligne reçue l'un à côté de l'autre, cela devrait vous aider à comprendre :

- Déclencheur : [hrp]* ;
- Ligne reçue : [hrp] Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ? .

Vous avez trouvé ? La ligne reçue par le client commence par "hrp" entre crochets, un espace puis le nom de la personne qui parle... alors que notre déclencheur capture tout ce qu'il y a après le crochet fermant de "[hrp]", ce qui inclue notre signe espace dans ce cas.

La solution : modifier quelque peu notre déclencheur.

```
[hrp] *
```

Cette fois, on met un espace entre le crochet fermant et le signe astérisque. Si l'on reçoit la ligne :

```
[hrp] Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ?
```

Et qu'on affiche $1, on devrait voir :

```
Edgar dit : je n'y comprend rien, quelqu'un pourrait-il m'aider ?
```

Les espaces dans les déclencheurs pourraient bien être l'une des erreurs principales quand on crée ses propres triggers la première fois. Le meilleur conseil que je puisse vous donner est de regarder avec attention les lignes que vous voulez surveiller à l'aide de triggers, et de n'utiliser le signe astérisque (*) que quand vous n'avez pas de moyen de savoir ce qui s'y trouve.

Les canaux CocoMUD dans les triggers

CocoMUD offre un système assez puissant de canaux. Cette fonctionnalité est à différencier des canaux en jeu, qui sont dépendants du jeu où vous vous connectez. Les canaux CocoMUD sont des listes d'`'evènements dans lesquels vous mettez ce que vous voulez. Ce n'est pas obligatoire de mettre des canaux en jeu dans des canaux CocoMUD, c'est juste plus comfortable. Nous allons voir cela (pourquoi l'utiliser et comment faire).

Vous trouverez plus d'informations sur ce concept et leur utilisation avec triggers en lisant la documentation des canaux.

Les triggers muets

Dans certains cas, quand on reçoit une certaine ligne, on ne veut pas l'afficher sur le client.

> Cela arrive vraiment ?

Parfois. Par exemple, certains MUD envoient des messages d'ambiance régulièrement (toutes les 3-5 secondes) quand vous vous trouvez dans une certaine situation. Ce peut être agréable pour mettre dans l'ambiance, mais pas toujours avec un lecteur d'écran.

```
Un noeud dans le bois éclate en une pluie d'étincelles.
```

Joli et, bien il faut le reconnaître, près d'un feu de camp il est normal qu'il pétille, lance des étincelles et craque de temps en temps en temps. Mais avec un lecteur d'écran, ce n'est pas le plus utile. Nous allons donc retirer cette ligne de l'affichage.

Pour ce faire, créez un nouveau trigger. En passant par l'interface :

- Ouvrez le menu **Jeu** -> **Trigger** ;
- Ajoutez un trigger en cliquant sur **Ajouter** ;
- Collez la ligne dans le déclencheur : Un noeud dans le bois éclate en une pluie d'étincelles. ;
- Pas besoin de sélectionner une action, sauf si vous voulez jouer un son approprié, cela dit.
- Tabulez jusqu'à trouver la case "Trigger muet". Cochez-la.
- Validez plusieurs fois sur **OK** pour sortir de la boîte de dialogue en sauvegardant.

Si le client reçoit cette ligne, il ne l'affichera plus.

Créer ce trigger en SharpScript est assez simple :

```
#trigger {Un noeud dans le bois éclate en une pluie d'étincelles.} {} +mute
```

Notez que le second paramètre (définissant la liste d'actions associées à ce trigger) est vide dans notre exemple. Le troisième paramètre quant à lui est un flag, commençant par + ou - et suivi du nom du flag (ici mute pour créer un trigger muet).

Vous pouvez très bien avoir un trigger muet qui exécute cependant des actions, cela n'est pas du tout incompatible.

Les triggers marqués

Les triggers marqués peuvent être utiles pour l'accessibilité. Quand un trigger marqué se déclenche, il place le curseur directement sur la ligne ayant déclenché ce trigger. Vous pourriez avoir un trigger sur la ligne décrivant les sorties d'une salle, par exemple. Si ce trigger est marqué, quand vous explorerez plusieurs salles, le curseur sera toujours déplacé sur la listes des sorties, plutôt que ramené en bas de la fenêtre, où vous devrez appuyez plusieurs fois sur la flèche haut pour lire les sorties disponibles.

C'est le même principe pour créer ce trigger :

- Dans la barre de menu, sélectionnez **Jeue** -> **Triggers**.
- Cliquez sur le bouton **Ajouter** pour créer un nouveau trigger.
- Écrivez dans le déclencheur quelque chsoe comme : Sorties :* .
- Tabulez plusieurs fois pour cocher la case "trigger marqué".

La prochaine fois que vous recevrez une ligne commençant par Sorties : , le curseur sera déplacé automatiquement dessus.

Créer ce trigger avec une instruction SharpScript donne :

```
#trigger {Sorties : *} {} +mark
```

Les triggers avec substitution

Dans certains cas, quand un trigger s'exécute, on veut modifier la ligne qui a déclenché le trigger. L'un des exemples les plus fréquents est pour réduire une ligne un peu longue. Certains MUDs ont de longues lignes de texte et mettent l'information importante à la fin, ce qui n'est pas super pratique avec un lecteur d'écran. Par exemple :

```
Quelqu'un parle publiquement sur le canal 'hrp' avec une petite voix inquiète : c'est sans danger
?
```

Bien que cette ligne de texte en apprenne beaucoup, l'information vraiment importante (le message) se trouve tout à la fin. Ce pourrait être bien de raccourcir un petit peu cette ligne de texte, et peut-être changer l'ordre des informations.

```
[hrp] Quelqu'un : c'est sans danger ? (une petite voix inquiète)
```

Pour faire cela, il faut créer un trigger, et créer une ligne de substitution. Quand le trigger est appelé, la ligne de substitution s'affichera à la place de la ligne d'origine qui a déclenché le trigger.

- Dans la barre de menu **Jeu** -> **Triggers** ;
- Cliquez sur **Ajouter** pour créer un nouveau trigger ;
- Écrivez le déclencheur comme d'habitude :

```
* parle publiquement sur le canal '*' avec * : *
```

- Notez que $1 contient le nom de l'auteur du message, $2 contient le nom du canal, $3 contient la voix utilisée et $4 contient le message lui-même ;
- Tabulez jusqu'à trouver la zone de texte appelée "Message de remplacement de la ligne ayant déclenché le trigger".
- Dedans, écrivez :

```
[$2] $1 : $4 ($3)
```

C'est compréhensible ? Si ce n'est pas le cas, prenez le temps de relire à quoi correspond chaque variable.

Créer le même trigger en SharpScript serait :

```
#trigger {* parle publiquement sur le canal '*' avec * : *} {} {[$2] $1 : $4 ($3)}
```

Note importante : nous avons trois arguments ici, dans notre action #trigger. Le premier contient toujours le déclencheur du trigger, le second la liste d'actions (vide ici). Le troisième contient la chaîne de substitution. Si il n'y a pas de troisième paramètre, ou que le paramètre est vide, la ligne est affichée telle qu'elle (c'est le cas dans tous nos exemples précédents).

Les triggers déclenchés par expressions régulières

Cette section et celles qui suivent sont un peu plus avancées.

Le symbole astérisque (*) est très pratique. Mais il n'est pas très précis. CocoMUD permet d'écrire des expressions régulières . Le but ici n'est pas de décrire la syntaxe de ces expressions, c'est un sujet à part entière, mais vous trouverez de nombreuses ressources (à commencer par le lien ci-dessus).

Pour utiliser des expressions régulières dans des déclencheurs de trigger, il suffit de commencer le trigger avec le signe ^. CocoMUD comprend que ce déclencheur doit être une expression régulière. Par exemple :

```
^Vous recevez \d+ XP.$
```

Ce trigger sera déclenché si vous recevez la ligne "Vous recevez ... XP.", avec ... étant un ou plusieurs chiffres. Ce trigger ne sera pas déclenché par la ligne : "Vous recevez un peu d'XP."

Une chose importante à garder à l'esprit quand on utilise des triggers avec expressions régulières, cependant, est que si on veut capturer des informations, il faut utiliser des groupes de capture (des parenthèses).

```
^Vous recevez (\d+) XP.$
```

Ici, le nombre d'XP sera mis dans $1. Vous pouvez aussi utiliser des groupes nommés et y faire référence avec $nom_du_groupe.

Des triggers avancés en Python

Le moteur SharpScript est léger et puissant, mais ce reste un langage de script qui ne permet pas tout. Et qui ne permettra pas tout : il est censé resté léger et optimisé. Ce n'est pas un langage de programmation. Mais Python en est un, et CocoMUD est développé en Python. Le moteur SharpScript a une syntaxe particulière pour envoyer du code à Python directement, ce qui permet d'outre-passer les limites du moteur SharpScript de façon assez définitive. Depuis le code Python, on peut toujours utiliser les fonctions SharpScript, ce qui rend l'élaboration de triggers plus complexes assez simple, dès lors que l'on connaît Python.

Nous allons prendre un exemple, comme toujours, mais souvenez-vous qu'il n'y a pas de réelle limite à ce que vous pouvez faire, tant que Python le permet.

Admettons que le jeu envoie une ligne quand on reçoit de l'XP mais précise aussi le nombre d'XP nécessaire pour passer au niveau suivant.

Par exemple :

```
Vous recevez 37 XP et avez besoin de 500 pour passer au niveau suivant.
```

On va vouloir extraire ces deux nombres et afficher un pourcentage à la place : xp / total * 100. Ce n'est pas possible en utilisant simplement du SharpScript.

Pour l'instant, ce n'est pas possible d'utiliser l'interface pour manipuler du code Python. Vous devrez faire les essais en éditant directement le fichier "config.set".

```
#trigger {Vous recevez * XP et avez besoin de * pour passer au niveau suivant.} {+
    # $1 contient le nombre d'XP gagné
    # $2 contient le total pour passer au niveau suivant
    xp = args["1"]
    total = args["2"]

    # On convertit ces deux nombres
    try:
        xp = int(xp)
        total = int(total)
    except ValueError:
        # La conversion n'a pas marché, mais on ne fait rien
        pass
    else:
        pourcent = xp * 100.0 / total
        say("Vous recevez {}/{} XP ({}%).".format(xp, total, int(pourcent)))
}
```

Voilà du trigger plutôt avancé ! Quelques explications :

- Le déclencheur du trigger ne devrait pas être une grosse surprise pour vous à ce stade ;
- Le second paramètre commence par {+ (accolade gauche suivi du signe +). C'est la syntaxe pour dire à CocoMUD que ce qui se trouve entre les accolades est du code Python.
- Notez que tout le code est indenté. Il ne s'agit plus d'un comfort de lecture ici, mais d'une nécessité. Python a besoin de l'indentation pour fonctionner. J'ai utilisé 4 espaces, mais vous pouvez en utiliser un (ou un signe de tabulation si vous voulez) ;
- Quelques commentaires. Cela peut toujours être utile.
- On extrait les deux nombres xp et total. Pour accéder aux variables du trigger, on utilise args qui est un dictionnaire les contenant. La première variable étant $1, on y accède en Python grâce à args["1"].
- On a besoin de convertir ces variables. Pourquoi ? Ce sont toujours des chaînes de caractère à ce stade, il faut les convertir en nombre. CocoMUD ne vérifie pas que ce sont des nombres, on doit donc le faire manuellement. C'est pourquoi on convertit

dans un bloc try/except/else. Notez qu'on ne fait rien si la conversion échoue pour X raison.
- On crée ensuite le pourcentage. Puisqu'il s'agit de Python2, il faut lui indiquer de faire attention aux virgules.
- On affiche notre message ensuite, en utilisant la fonction say|(). C'est exactement comme appeler la fonction #say en SharpScript, c'est la même fonction qui est appelée. De même façon, on peut utiliser les fonctions send(), play(), feed() et autre. La syntaxe diffère car on est en Python, mais ce sont les mêmes fonctions et mêmes arguments.

Bien... je ne suis pas vraiment satisfait par le trigger que je viens de créer... on pourrait le raccourcir un petit peu, et le rendre plus lisible. Il ne faut pas toujours chercher la concision, mais ça aide. Si on était sûr que nos variables contiennent des nombres, on aurait moins de code à faire :

```
#trigger {^Vous recevez (\d+) XP et avez besoin de (\d+) pour passer au niveau suivant.$} {+
    # $1 contient le nombre d'XP gagné
    # $2 contient le total pour passer au niveau suivant
    xp = int(args["1"])
    total = int(args["2"])
    pourcent = xp * 100.0 / total
    say("Vous recevez {}/{} XP ({}%).".format(xp, total, int(pourcent)))
}
```

Utiliser les expressions régulières ici ajoute un peu de complexité au déclencheur, mais ça rend le code bien plus lisible je trouve.

Si vous voulez plus d'informations sur l'utilisation de code Python dans les instructions SharpScript, vous trouverez une section détaillée dans la documentation du SharpScript.

CocoMUD

CocoMUD est un client MUD qui se veut simple et accessible avec un lecteur d'écran. Son but est de proposer autant de fonctionnalités que n'importe quel client MUD disponible, mais en étant plus accessible. Une des fonctionnalités basiques du client est un Text-To-Speech (TTS) permettant d'envoyer du texte pour être lu par le lecteur d'écran, ou affiché par une plage braille.

Accueil Téléchargement Premiers pas

CocoMUD dans d'autres langues: Anglais.

Utiliser CocoMUD

Pour télécharger le client CocoMUD, rendez-vous sur la page de téléchargement. Si c'est la première fois que vous utilisez CocoMUD, vous pourriez vouloir lire les premiers pas avec CocoMUD pour apprendre à utiliser les fonctionnalités basiques du client.

Fonctionnalités basiques

Les fonctionnalités suivantes servent de base à la planification du développement du client, en suivant sa feuille de route (road map). Vous pouvez cliquer sur l'une de ces fonctionnalités pour voir son avancement (en anglais).

- Un client MUD avec un système réseau stable (#5).
- Un client accessible avec la plupart des lecteurs d'écran (#6).
- Un client MUD portable sous Windows, Linux et Mac OS (#7).
- Une interface traduite en plusieurs langues (#8).
- Un mécanisme simple mais puissant pour personnaliser l'expérience utilisateur (#9), avec des alias, des macros, des triggers et bien d'autres choses.

Licence

CocoMUD est distribué sous la licence 3-clause BSD. Son code source est disponible dans son dépôt Github . Malgré tout, la majorité du projet (discussions, bugs, suggestions, forums, documentation) est hébergé par PlanIO.