Introduction technique au MOO

Code: moo-tech

Originaux

url: http://tecfa.unige.ch/guides/tie/html/moo-tech/moo-tech.html
url: http://tecfa.unige.ch/guides/tie/pdf/files/moo-tech.pdf

Auteurs et version

- Daniel K. Schneider

- Version: 0.10 (modifé le 15/8/01 par DKS)

Prérequis

- Aucun

Autres Modules

Module concepts/théorie suppl.: moo-tecfa

Objectifs

- Découvrir l'historique des MOOs
- Connaître les commandes de communication et de navigation dans un MOO

1. Table des matières détaillée

1. Table des matières détaillée 3
2. Historique des MUD, Muses, Mushes, MOOs 4
3. Comment se connecter ? 5
4. Commandes de base 7
   - 4.1 Commandes de base pour la communication synchrone 7
   - 4.2 Autres commandes (pas disponibles pour les guests) 10
   - 4.3 Navigation de base 11
   - 4.4 Simple manipulation d'objets 13
   - 4.5 Création et modification d' objets 14
5. Autres activités 16
   - 5.1 Informations techniques 17

2. Historique des MUD, Muses, Mushes, MOOs

- 1975 Jeux d'aventures textuels
- 1978 Premier MUD ("Multi User Dungeon")
- 1989 TinyMUD (variante Unix)
  (naissance de jeux de rôles virtuels et de "social MUDs",
  explosion du nombre de MUDs,
  "Multi User Dimensions")
- 1990 Premier MOO ("Mud Object Oriented")
- 1991 Annonce officielle de l'ouverture du LambdaMOO
- 1991 MUSE (environnement MUD pour enfants, références à LOGO)
- 1992 Premiers usages "sérieux" du MOO
- 1993 (septembre) Création du "Diversity University MOO"
- 1994 (novembre) Création du TecfaMOO
- 1995 Interfaces WWW (aussi au TecfaMOO) et VRML
- 1996 Concurence des "Chat-worlds" 2D et 3D
- 1997 Entrée en jeu des mondes VRML
- 2001 Toujours pas de cyberspace protocol équivalent en puissance conceptuelle

3. Comment se connecter ?

A. Clients MOO

☞ Avec un browser "Java-enabled"
- la plupart des Netscape 2.x, 3.x ou 4.x ou Ms-Explorer
- URL: http://tecfa.unige.ch/moo/connect-page.html

☞ Avec un client "MUD/MOO"
- un bon client (par exemple tkmoo light) vous donne plus de possibilités.
- (nécessite un certain travail d'installation, le langage tk/tcl en plus)

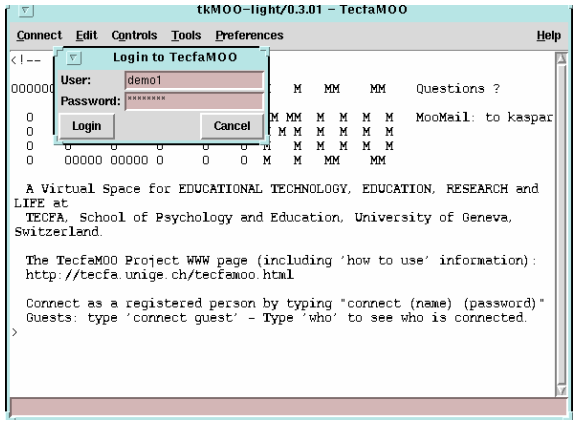
☞ Note: les interfaces WWW ont une fonctionalité limitée
- pas de possibilité de communication !

Au pire des cas on peut se connecter via un simple telnet

B. Addresse du MOO et login

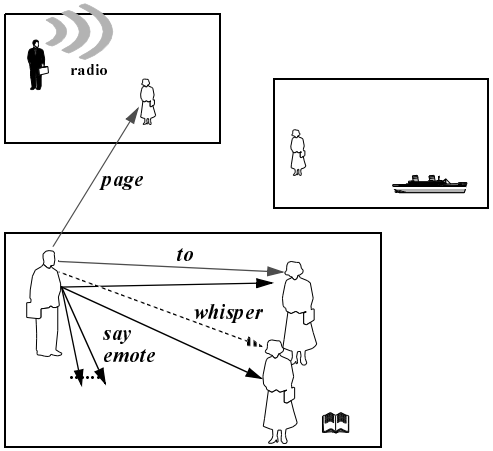
Il faut spécifier (généralement dans un menu)
- Le nom de la machine
- Le numéro de la porte Internet
- (ces paramètres peuvent être sauvés en règle générale)

TecfaMOO
- TecfaMOO se trouve à l'adresse: tecfamoo.unige.ch:7777
- voir: http://tecfa.unige.ch/moo/how_to.html



4. Commandes de base

4.1 Commandes de base pour la communication synchrone



Dans les exemples qui suivent vous êtes "John"

say **Dire quelque chose à tout le monde dans une salle**
" **(abréviation)**

Exemples:
`(1) say Hello, there.`
Vous allez voir: You say, "Hello, there."
`(2) "salut`
Vous allez voir: You say, "salut".

emote **Expressions "non-verbales"**
: **(abréviation)**

Exemple:
`(1) emote smiles.`
Vous allez voir: John smiles.
`(2) :smiles`
(la même chose)

whisper **communication privée**

Exemple
`(1) whisper "Hello, Jane darling" to Jane`
Visible seulement pour Jane (si elle se trouve dans la même salle).
Vous allez voir: You whisper, "Hello, Jane darling" to Jane.

- **S'adresser publiquement à une seule personne**
‘ **(abréviation, backquote!)**

Exemple:
```
(1) -Jane I agree with you.
```
Vous allez voir: John [to Jane]: I agree with you.
```
(2) - “... I agree again
```

page **Communication à distance**

Exemple:
```
(1) page Jane Do you have some free time?
```

who **Afficher toutes les personnes connectées au MOO (+ le temps d'inactivité)**

who <nom>... donne de l'information sur une personne

where **Afficher les personnes connectées et leur emplacement dans le MOO**

4.2 Autres commandes (pas disponibles pour les guests)

think **“penser” (Visible pour tout le monde dans une même salle)**

Exemple:
```
(1) think I think therefore I am.
```
Vous allez voir: John . o O (I think therefore I am)

+ **“emote” à distance**

Exemple:
```
(1) +guest waves
```
Peut donner: Jane waves to you (si vous êtes “guest”).

finger **Regarder les intérêts des gens**

Exemple:
```
(1)finger Daniel
```
Vous allez voir (par exemple):
```
This is Daniel K. Schneider's MOO wizzard character.
Daniel is the founder of this MOO and main coordinator of
research & education projects in TecfaMOO.
Other "MEs": Daniel, MooBoy, Dan_Test
```


A. Les canaux "radio"

- Il faut demander une autorisation pour écouter et utiliser certains canaux
- Si c'est pas déjà fait (!) il faut configurer son personnage:
  @add-feature &channel_feature
- Aide (il exist pleins(!) de commandes, ci-dessous juste les plus importantes
  help &channel_fature

x .... **say**

Exemple:
```
(1)x Je suis perdu, HELP !
```
Tout le monde branché sur le même canal peut voir ce message,

x: .... **emote ..... sur votre canal actuel**

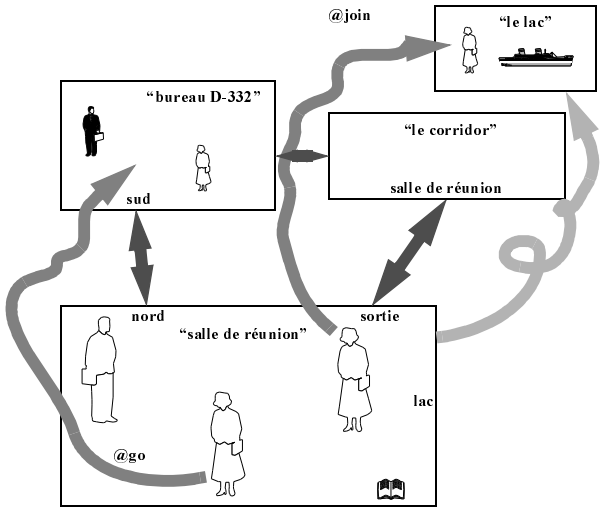
```
Exemple:
x: est fatiqué
Les autres voyent: [+][Despotic Announce] Kaspar est réveillé
```

xx>>*canal* **say (en changeant le canal!).**

xx//*canal* **say sur un autre canal.**

@xwho **who est sur un canal**

4.3 Navigation de base



go **marcher dans une direction**
En règle générale il faut lire la description d'une salle. Elle indique les sorties (par ex "north", "south", "out"). En tapant ces directions ou "go <direction>" on peut se déplacer.

@nicknames **se souvenir d'une "chambre"**
```
(1) @addnickname atrium to #101
(2) addnickname atrium to here
```

@nicknames all **Afficher la liste des chambres mémorisées**

@go **téléportage vers un objet**

@join **joindre une personne à un endroit**
```
(1) @join Daniel
```
Vous rejoignez Daniel à l'endroit où il est

Notez qu'il est poli de "frapper" avec @knock ou de faire un "page" avant de joindre une personne qui se trouve visiblement pas dans un endroit public.

artéfacts **(par exemple "trains", "ascenseurs", "bus", "stations de téléportage ....)**

4.4 Simple manipulation d'objets

look **Examiner un objet (afficher une description)**
Exemples:
```
(1) look
```
Afficher la description de la salles
```
(2) look board
```
Regarder l'objet "board"

inv **Lister l'inventaire des objets que vous portez**

get **Prendre un objet (take=synomyme)**
Exemples:
```
(1) get boite
```
Prendre la boite qui se trouve "par terre"
```
(2) get boite from box
```
Prendre la boite qui se trouve dans la box

drop<objet> **Laisser tomber un objet par terre**

put **Mettre un objet quelque part**
Exemple:
```
(1) put boite in box
```

give **Donner un objet**
Exemples:
```
(1) give Pet to Daniel
```


Donner l'objet "pet" à l'utilisateur Daniel

4.5 Création et modification d' objets

- quelques commandes de base uniquement
- Note: Ces commandes ne sont pas disponibles pour les "guests"

@dig **construction d'un "room"**
Exemples:
```
(1) @dig palace
```
"Creuse" un "room" nommé "palace"
```
(2) @dig n,north|s,south to "maison à côté"
```
Crée un nouveau room "maison à côté", puis rajoute une sortie ("exit") nommé "n" (avec l'alias "north") depuis le room courant vers le room nouveau, et une sortie "s" depuis le nouveau room vers le room depuis lequel vous "creusez".
```
(3) @dig out to #312
```
Crée une sortie vers le room possédant le No. #312. Notez que vous devez posséder l'objet #312. Sinon il faut utiliser les commandes @add-exit et @add-entrance.

@sethome **Définit votre "home"**
Définit le home dans le "room" courant (à condition d'avoir l'autorisation)

@describe **Décrire un objet**
```
@describe here as "A lovely park with thousands of pink flowers"
```
Décrit le "room" courant (à condition qu'il vous appartienne), Avec la commande '@notedit here' vous pouvez faire la même chose avec l'éditeur de texte interne au MOO.

@rename **Donner un autre nom à un objet**

Exemple:

```
(1) @rename me to “débutant”
```

Vous transforme en débutant ...

```
(2) @rename #500 to “Porte rouge”, sortie, exit
```

Renomme la sortie #500 (si elle existe) en “Porte rouge” avec 2 alias.

@gender **détermine le sexe d’un personnage**

Exemple:

```
(1) @gender male
(2) @gender female
```

@create **Créer un objet**

Exemple:

```
@create $note named feuille
```

Crée un objet de type “$note” avec le nom “feuille”.

5. Autres activités

- chapitre à refaire ....

A. La communication asynchrone

- Système de courier interne
- Système de “conférences” (à la “News”)
- Journaux virtuels
- Tableaux, affiches, notes, livres, etc.
- Passerelles vers www, gopher et email externe

B. “Artéfacts”

- Salles de classes virtuelles (avec tableau, tables, etc.)
- Moyens de transport
- “Robots” (par ex. “pattern matchers” de type Eliza)
- Divers supports pour écrire
- ... et beaucoup plus !

5.1 Informations techniques

A. Fonctionnement

- Système (serveur) multi-utilisateurs interactif, programmable de l’intérieur
- Les participants se connectent via un “client” à un “caractère” qui réside dans la base de données du serveur
- Les utilisateurs peuvent entrer des commandes qui sont analysées et interprétées (exécutées) par le serveur
- Ces commandes peuvent accéder aux informations et altérer l’état de cette “réalité virtuelle” (déplacements, communication, apparence d’objets)
- L’environnent peut être élargi soit en “construisant” soit en programmant (les participants ont donc un rôle très actif)
- Différentes classes de “participants”: “Wizards”, administrateurs, programmeurs, constructeurs, simples utilisateurs, “guests”.

B. Programmation

- un langage orientée objets:
  - objets (numérotés) ayant des propriétés (“slots”)
  - objets génériques
  - héritage simple (mais “features” pour “mixer”)
- un système de permissions (par ex. r w x f pour les objets)
- Analyse de commandes
  - est basée sur un modèle de syntaxe simple (verbe - objet direct - préposition - objet indirect)
  - Le serveur fait un “matching” par rapport aux objets dans la même salle et appelle un verbe “présent” (sur le participant, la salle, l’objet direct, l’objet indirect)
  - Le “verb” reçoit une liste de “builtins” (player, liste d’arguments, objet direct, indirect, auto-référence, etc).
- Syntaxe: ressemble un peu à C, mais le langage est plus près de LISP
- Extensions du langage: objets “utilitaires” (Lambda Core Database, extension du core, etc.)