Chapter 2 – How Do They Do That?

Overview

After you've visited a MOO a few times, requested a character, and started to get the hang of communicating with others, you will undoubtedly begin to notice that there's a whole lot more to MOOing than `say` and `emote`:

- You will find that MOOs have different rooms and locations to explore.
- You will learn that there are commands you can type that will cause some of the depicted objects to behave in various ways.
- You will undoubtedly have seen many *object numbers* (indicated by the #-sign).
- People may arrive and depart in showers of sparks, clouds of smoke, or a burst of flame, and you may begin to wonder, "How do they do that?"
- You'll soon discover that the more experienced players have a great many commands available to them that you do not.

While you don't have to know the particulars of how a clutch works in order to drive a stick-shift car, a basic knowledge of how the clutch pedal, gas pedal and gear shift interact will make the going much easier. Similarly, you don't have to know how to program to enjoy MOOing, but a knowledge of some of the underpinnings will enhance your MOOing experience.

This chapter introduces several of these advanced techniques; the following chapter discusses some of these things in greater depth.

A Very Brief Introduction to Objects

A *MOO* is a multi-user domain that includes a programming language that anyone may use to extend the domain. The particular kind of programming language that MOOs use is called an *object-oriented* language, and as you explore the MOO, you will encounter *objects* everywhere.

What you need to know about objects at this juncture is that they are there. It is also helpful to know that every object has a unique number that identifies it. If you buy something from a mail order catalog, you might call and say, "I'd like to buy one frizzlebopper, please," or you might say, "I'd like to place an order. The first item number is #4612." Object numbers on a MOO are like item numbers in a mail order catalog.

There is an in-depth discussion of objects beginning on page 37.

Exploring an Object-Oriented World

Text-based virtual worlds generally have an underlying metaphor or *theme*. When you first connect, you typically see a description of where you are within that world. It might be an open field, the deck of a space ship, a room in a house, or the reception area of an office building. Within the world described by the words on your screen you can explore and interact with the various things and people you encounter there.

To see a description of the room you are in at any time, type `look`, which can be abbreviated to the single-letter command, `l`.

Let's suppose that you've logged onto LambdaMOO for the first time, and have opted to begin in the quiet location. After seeing all the information about news, the tutorial, and `help manners`, you may want to have another look your surroundings. Type:

```
look
```

You will see the following text:

```
The Linen Closet

The linen closet is a dark, snug space, with barely enough
room for one person in it.  You notice what feel like
towels, blankets, sheets, and spare pillows.  One useful
thing you've discovered is a metal doorknob set at waist
level into what might be a door.  Another is a small button,
set into the wall.
```

Now let's look at some things that are in this linen closet. If you type, `look towels` (or `l towels`), you will see:

```
You can make out the outline of some towels, but it's too
dark to tell what color they are.
```

This is nice, but not especially interesting; how about the blankets?

```
l blankets

You can make out the outline of some blankets, but it's too
dark to tell whether they are flannel, wool, or electric.
```

Well, so far, so boring. But not as boring as it could be, actually. Someone, in fact, went to the trouble of writing text for you to see if you look at the towels, the blankets, the sheets, the pillows, the button, or the door. Most rooms, as a matter of practicality, have some things that are merely mentioned in the room's description, and others that actually exist as things with descriptions in their own right. Suppose the author had elected not to include the pillows as something you could look at. In response to `l pillows`, you would see:

```
I see no 'pillows' here.
```

The author of the linen closet mentions two things as "useful": a doorknob and a button. But *how* are you to make use of them? You might think to try typing `turn doorknob` or `push button`, and if you were to do so, something would happen in either case. Before you leave the linen closet, though, it's worth knowing that you don't have to guess about which objects are interactive and how to interact with them; the command `examine` exists to enable you to determine that. If, instead of typing `look button` you were to type `examine button`, you would see the following:

```
button (aka #53344 and button)
Owned by Groundskeeper.
A small black button, set into a tarnished bronze plate on
the wall.
Obvious verbs:
  press/push/poke button
```

The `examine` command can be abbreviated to `exam`. `Exam doorknob` yields:

```
doorknob (aka #79708, doorknob, and knob)
Owned by Groundskeeper.
You see a plain metal doorknob.
Obvious verbs:
  turn doorknob
```

Let's consider each of these two instances. Both the button and the doorknob are *objects*, which means that they exist as things within the MOO and aren't merely mentioned in the text of a room's description. The first line you see after examining an object is a list of the object's aliases, and the object's number. (For now, it's perfectly fine to ignore the object numbers, but referring to an object by its number always works.) The button has only its object number and its name ("button") as aliases, but the doorknob also has the alias "knob", so the casual user could type `turn knob` and get an appropriate result. The second line you see tells you who owns the object. This, too, you can ignore for now, but if the button or doorknob were to malfunction in some way, the appropriate person to notify would be Groundskeeper, who owns these objects. Then there is a list of "obvious verbs". (It's a mystery to me why objects aren't called "nouns" or why verbs aren't called "commands", but that's the way it is.) Verbs are commands that you can use to manipulate objects. (The word "verb" has a broader definition in MOOs, but it's sufficient, for now, to think of it as the name of a command – something you can do with or to an object.) In these two cases, you could either turn the doorknob or push the button. It is an idiosyncrasy of MOO syntax that definite and indefinite articles are usually omitted.

I will note here that there are inconsistencies within the MOO as to how you can tell, within a room, what objects are in it that might be interesting or useful, and this is because different rooms are programmed differently. Every room has a mechanism (i.e. a program) associated with it to display its contents. Early on, most rooms displayed their descriptions and contents like this:

```
Guest Cottage Porch
You are on a breezy, screened-in porch.  A rocking chair and
a porch swing invite you to stay and relax for a while.  A
```

```
screen door leads west into the cottage; steps lead down to
the lawn.
You see glass of lemonade and harmonica here.
Yib and Bartlebooth are here.
```

First is the name of the room, then the room's description, then a list of non-player objects in the room prefaced by the words, "You see", then a list of the players present. With such a scheme, a player might reasonably pass up trying to examine the rocking chair, the porch swing, the screen door and the steps, and would zero in on (and examine) `glass of lemonade` and `harmonica`, hoping, perhaps, to be able to `drink lemonade` and `play harmonica`. Said visitor to the guest cottage porch might also be moved to greet Yib and Bartlebooth.

At some point, it became possible for rooms to be dark, because some rooms are. The coat closet and the linen closet on LambdaMOO are two examples of dark rooms. These rooms specifically don't list the items or people who are present, unless they're also mentioned as part of the text of the room's description (like the towels in the linen closet).

Later developments in room technology enabled statements about some objects' presence to be integrated into a room's text description. So, for example, if the guest cottage porch were an integrating room, and a model of the gazebo had an integrating message on it, it might look like this:

```
Guest Cottage Porch
You are on a breezy, screened-in porch.  A rocking chair and
a porch swing invite you to stay and relax for a while.  Off
to one side is a model of the gazebo.  A screen door leads
west into the cottage; steps lead down to the lawn.
You see glass of lemonade and harmonica here.
Yib and Bartlebooth are here.
```

Anytime the model of the gazebo was moved to an integrating room, the sentence, "Off to one side is a model of the gazebo," would appear in that room's description rather than in the list of objects that conventionally follows the text description.

Another type of room is the detailed room, whereby an owner can add extra descriptions, so, for example, if a room's description mentioned wallpaper, you could type `look wallpaper` and get some extra detail, even though there wasn't actually an *object* named `wallpaper` in the room. So in some rooms there are objects whose presence is not obvious because a room is dark, or there are objects whose presence is not obvious because the objects are integrated into the room's text description, *and* there are "things you can look at" (in detailed rooms) that aren't actually objects at all. Each of these developments was seen as an improvement at the time. Detailed rooms were thought to be "richer" than rooms that had single, simple descriptions. Integrating rooms were thought to "read more naturally" than rooms that appended, "`You see <stuff> here`," at the end of their descriptions. All I can say is, bear with it, and when in doubt, `examine`. Eventually you get a feel for it.

To summarize, then: `look` or `l` to see a description of the room you are in at any time. `Look <object>` or `l <object>` to see the description of an object, if there is

such an object in your vicinity. `Examine <object>` or `exam <object>` to see a list of its aliases, find out who owns it, and what, if anything, you might try doing with it.

Moving Around in a MOO

MOOs generally depict a spatial or architectural metaphor. When you first connect, you are in a room – either the room designated as `$player_start` (by definition the place where players start), or in another room that you've created and/or set as your home.

There are two basic modes of moving around. These are generally referred to as *walking* and *teleporting*. Another way of thinking about them might be *VR* (Virtual Reality) and *non-VR* or *meta-VR*: ways that transcend or break the frame story of the virtual reality.

In the computer games Adventure and Zork, in which MUDs and MOOs have their roots, most travelling was done using compass directions (and sometimes "up" and "down"). You might see, for example, words to the effect that there was a house to the east, and you would type the command `east` or `e` to enter the house. Although many people find the notion of moving in compass directions non-intuitive – when I leave my house I simply go out the front door without thinking "northeast" – nonetheless compass direction exits are still in frequent use in MOOs today.

Most room descriptions whose exits are named for compass directions will give cues to that effect. For example, part of the description of LambdaMOO's library alcove reads, "`The library itself is back to the east. A spiral staircase leads up.`" You might reasonably expect that typing `east` would move you to the library and that typing `up` would move you up the stairs, and you would be right. It is possible to look before you leap, so to speak. You can type `look east` or `l east` before typing `east`. Failing to look is rarely of consequence, but LambdaMOO builders are strongly encouraged to give their exits descriptions, and from time to time there's an especially good one. It's worth knowing that the descriptions are there, at any rate.

On some MOOs, you can type `help map` to see a map of the MOO's main area or areas[8]. There are also some non-VR ways to check for available exits and get your bearings. You can type `@ways` for a list of obvious exits from the room you are in.

Exits can have other names besides compass points and "up" and "down". Some are intended to be intuitively obvious ("`out`", for example, from LambdaMOO's coat closet and linen closet), and some are obscure ("`vent`" from LambdaMOO's kitchen

[8] On LambdaMOO, there is an atlas on the mantel in the Living Room (to use it, go to the Living Room, type `take atlas from mantel`, and `examine atlas` to see its commands). There is also a copy of it on the geography shelf in the library.

You can also add the Compass Rosette Feature Object (#23824) (see pages 41 and 224) and then type `@rose`. Obvious Features (#41975) has the command `@lrs` for "long-range-scan" which lists all the rooms within three exits of your current location.

will move you into the vent system from there). Though not the norm on LambdaMOO, some areas there and on other MOOs use non-compass exits primarily, and often the names of these exits are capitalized within the text of a room's description.

The `go` command lets you string multiple exits together. Typing:

```
go north west
```

is the same as typing `north` and *then* typing `west`. You can also abbreviate exits in combination with the `go` command. From LambdaMOO's living room, for example, you can type `go n w` and wind up in the dining room as if you had typed each exit name separately.

A later development was the concept of a room-within-a-room, and there are two major implementations of this idea: one is a *container room*, that – as you might expect – is a hybrid between a room and a container. Like other containers, you can open and close it, and can put things into it (including people!). In addition, since such a room is *inside* another room, instead of entering it via a conventional exit (i.e. an exit that you simply type the name of to use), you must enter the container room explicitly. The dishwasher in LambdaMOO's kitchen is an example of this kind of room. If no one is around to stuff you in, you can check out the inside by typing:

```
open dishwasher
enter dishwasher
```

Sometimes it doesn't make sense to open and close a room as if it were a container, and so someone developed a portable room that wasn't container-like. By convention, you go into such a room by using the `enter` command, e.g. `enter helicopter`.

In either case, with portable rooms or container rooms, you can leave by typing either `exit` or `out`. (Most of them will also let you type `look out` from within, to see what's outside.)

There are other ways of getting around that are said to be "consistent with the VR". These include elevators, trains, planes, paintings that transport you to the places they depict when you gaze at them, and so forth. Rooms that can be reached exclusively by conventional exits or in ways that are consistent with the VR are said to be *connected*.

Teleporting

Teleporting is a way of moving around that "breaks", "violates", or "bypasses" the virtual reality, which is another way of saying that it is a way of going from one room to another without using conventional exits or other VR means. The three most commonly used commands to do this are `@join`, `@go`, and `home`[9]. Notice that

[9] In the early days of LambdaMOO, teleportation in the form of `@join` and `@go` didn't exist. Morpheus made a generic ring of teleportation which functioned in a way somewhat similar to the way `@addroom` works today. Eventually everyone had one. At that time, quota was object-based rather than

two of these commands begin with the @-sign, a common signal that a command is not strictly VR.

Suppose you are in the LambdaMOO library and your friend Werebull pages you from the pool. He pages, "Come on down, we're having a party!" Maybe you don't want to take the trouble of going south to the corridor, west (four times) to the entrance hall, south to the living room, southeast to the deck, and then south to the pool deck. Or maybe you don't know the way. At any rate, on this particular occasion, you just want to be there already and not take all the time and trouble of walking. You can type `@join Werebull` and voilà! You instantly join him wherever he is (in the swimming pool, in this case).

On another occasion, you might want to teleport to a particular room (regardless of who might already be there) without going to the trouble of walking, and for this there is the `@go` command. In this case, you will need to specify the room to which you wish to teleport, and this is one case where all those pesky object numbers beginning with the #-sign come into play. To teleport to a distant room, you will generally need to specify that room's number. For example, to go to the foyer of the LambdaMOO Museum, you would type:

```
@go #50827
```

How do you find out a room's number? Several ways. You can ask around. If it's a popular room, many players will know the room's number by heart and will be able to tell you. If you are in a room, you can find out its number for future reference by typing `exam here`. You might read a room's object number in other written references to it, and it is for this reason that written references to rooms (and other objects of general interest, for that matter) usually include the object number.

Remembering Rooms' Object Numbers

Since teleporting is so common, and since remembering all those object numbers is so cumbersome (for most people, anyway), there is a facility for you to associate the name of a room with its object number and store that information, and the `@go` command is able to use that list. The three commands for maintaining your list of rooms are `@rooms`, `@addroom`, and `@rmroom`. These commands are provided as part of LambdaCore.

Suppose you are exploring, find the foyer of the LambdaMOO museum, and think that you think you'd like to return to that location to from time to time. While there, you could type:

---

byte-based, which means that people were limited to a certain fixed *number* of objects (ten, I think it was), regardless of their size, rather than an indefinite number of objects limited by the total amount of storage they take up. Creating a ring of teleportation, then, used up a significant percentage of your building allotment. Since everyone had one, and since it was perceived as being expensive, someone lobbied to have the code added to a player class, instead. This was done. To "encourage" people to recycle their teleportation rings, a note was posted on the refrigerator telling people to be sure to wash their hands after a food fight. When a person did so, e lost eir ring down the drain and into the garbage disposal, with suitably appropriate sound effects. (With thanks to Doug (#3685) who shared this story with me in September, 1999.)

```
@addroom museum
```

The system will then add the name "museum", paired with its object number, to your player object, and thereafter if you want to teleport there, you can type `@go museum` instead of `@go #50827`. The place where this information is stored is called a player's `.rooms database`. (I usually pronounce the period as "dot", so if I were speaking, I would say, "a player's dot-rooms database.") On most MOOs, guests and new players are provided with a preliminary list of rooms the wizards think they might want to teleport to by name, and players can add to this list as desired (using the aforementioned `@addroom` command). To see a list of the rooms you have specified so far, along with those that have been specified for you, type:

```
@rooms
```

To remove a room from your `.rooms database`, you can type:

```
@rmroom <name>
```

Finally, you can type `@go home` to teleport to your home.

You can also type:

```
home
```

on a line by itself to teleport to your home. This is an exception to the @-sign convention, for historical reasons (some MUDs only had one teleport command, and that was `home`).

As a point of etiquette, it can be awkward for all concerned if you teleport somewhere only to find that you've barged in on a private conversation or a tryst, and some players, even if they're alone, will react negatively if you teleport to their location without first paging them to find out whether you might be welcome at that particular time. It is possible for players to lock rooms that they own in order to prevent unwanted or uninvited entry. It is the case, however, that players don't always do this, yet still complain if someone teleports in. Private and public rooms are addressed briefly in the section on privacy that begins on page 31, but if in doubt, page before teleporting.

The Give and Take of a Multi-User Environment

One of the draws of a multi-user environment is that you get to meet and interact with other people whom you might otherwise never have encountered. And one of the drawbacks of a multi-user environment is that you may encounter people who not only do not embrace the same norms of behavior that you do, but whose behavior may be annoying at best. Different MOOs have different themes, aims, and local customs – thus different conventions of acceptable behavior. Some MOOs are family oriented, some are educational, some are Dungeons and Dragons games, some are professional, and some are explicitly "anything goes". It would be presumptuous to assert that any particular way of behaving is generally acceptable or generally unacceptable. Many MOOs have documentation available like `help manners` or

`help rules`, and I would urge you to seek this document out on your MOO and acquaint yourself with its contents.

This section details a variety of defense mechanisms, built in to the MOO, that you can use to counter various annoyances. From these, one may infer that certain behaviors carry the risk that you may annoy others if you engage in them – two sides of the same coin.

Noise Abatement

@gag

There may come a time when you wish you could just "turn someone off", so to speak, and you can, after a fashion. The `@gag` command prevents you from seeing text generated by a specified player or object.[10] The syntax is:

```
@gag <player or object>
```

It remains in effect until you type:

```
@ungag <player or object>.
```

You can see a list of people and objects you are `@gagging` with the `@gaglist` command. `@gaglist all` will show you a list of people who are `@gagging` *you*, though on large systems that command may be slow to execute, because it has to look at each player in the database individually.

Sometimes it is just as effective, if you have the self-discipline for it, to ignore someone *as if* you were `@gagging` em even though you aren't. The principle, here, is not to reward obnoxious behavior with a reaction. (Whom you are `@gagging` is readable by others. The information is kept in a player's `.gaglist` property, and some players take umbrage at *being* `@gagged`, which is one situation where simulated gagging might be one way to go.)

The flip side of `@gag` is generating unwanted noise, either generally or directed at one person in particular. Typing in all uppercase letters is sometimes construed as shouting. A different definition of shouting is writing a program that broadcast's text to all connected players (or almost all – leaving out one or two people doesn't exactly exonerate you). *Spamming* refers to generating so much text that its sheer quantity is offensive regardless of its content. Spam can be more than just offensive – it can be disabling for another user who has a very slow communications link to the MOO. You might also be `@gagged` just for making a nuisance of yourself – it's the prerogative of the person hearing the noise to `@gag` you or not, as e sees fit. If you have been `@gagged` by someone and would like to ask em to reconsider, you might ask a mutual acquaintance to intervene on your behalf.

---

[10] On LambdaMOO, new players are created with a lag-reduction feature pre-installed, and must first type the command `@rmlag` before `@gag` will work.

@paranoid and @check-full

Sometimes it's hard to tell whom to `@gag`, because someone may cause *unattributed* text to be displayed to your screen. Unattributed text is text whose origin is unclear. Falsely attributed text is text that appears to have been generated by one person when in fact it was generated by someone else – this is called *spoofing*. The `@paranoid` command records not only the last <number> lines that have been displayed to your screen, but their origins as well. The syntax is any one of:

```
@paranoid
@paranoid off
@paranoid immediate
@paranoid <number>
```

If you see some baffling bit of text, you can then type `@check-full <text>` and the system will print out a trace from which you can usually make a fair guess as to who initially typed in the command.[11]

The sort of behavior that occasioned `@paranoid` and `@check-full` is producing unattributed or falsely attributed text, especially with the intention of confusing or deceiving others. For example, if Klaatu causes the text, "`Yib farts loudly,`" to be displayed, Yib would have cause to complain and Klaatu is probably in the wrong (depending, as always, on local custom).

@refuse

The `@refuse` command lets you refuse certain actions from all players or from a specified player. The syntax is `@refuse <action> [from <player>] [for <duration>]`. The parts in square brackets `[]` are optional. The actions that you can refuse are:

- `page` – prevent someone from paging you
- `whisper` – prevent someone from whispering a message to you
- `mail` – prevent someone from sending you a message via MOOmail
- `move` – prevent someone from teleporting you (Note, this can affect teleportation – @go, @join, etc. – because if you @refuse move without specifying a particular player, you refuse to be moved by *anyone*, including yourself!)
- `join` – prevent someone from entering the same room as you (only works in a few rooms that support this functionality)
- `accept` – prevent someone from handing you an object (or teleporting it to you)
- `flames` – posts from the refused player(s) are suppressed on mail lists
- `politics` – refuse programmatic campaign solicitations (LambdaMOO only)

[11] As with `@gag`, on LambdaMOO you must type `@rmlag` before being able to use this facility.

- `all` – all of the above

The flip side of `@refuse` is rather general, but you might deduce that moving someone without eir consent might result in an `@refusal`, as might sending em obnoxious mail, teleporting unwanted things into eir inventory, etc.

You can type `@refusals` to see what actions you are currently `@refusing`, and can type:

```
@unrefuse <action> from <player>
```

to cease refusing an action. As with your gaglist, your `@refusals` list is readable by others. To see someone else's refusals, type `@refusals for <player>`.

Privacy

Just because it's pseudonymous, doesn't mean that it's private, and in fact, privacy on a MOO is almost impossible to guarantee. People may not know your off-MOO identity, but they can and do seem to pay a surprising amount of attention to what other people are doing. If you have something truly sensitive to discuss, it really is better to take it off the MOO.

@lock

You can use the `@lock` command to prevent unexpected and/or unwanted entry into a room you own. The syntax is `@lock here with <key>` e.g., `@lock here with me || <someone else>`. You can specify as many people as you wish, separated by "`||`" which translates to "or" in the locking syntax. To unlock your room again, type `@unlock here`. Various room classes on different MOOs may provide more elaborate programming for ease of controlling access.

The flip side of locking and room security is joining people when you are uninvited, unannounced, or unwanted. There may be MOOs where there is a way to designate a room as public or private, but I am not acquainted with any, which means that in many cases you may have to guess, and/or learn from experience. It is probably a pretty good bet to guess that a room named "Yib's Room" is private, and that you might receive a less than enthusiastic welcome if you teleport in when Yib is in the middle of a private conversation with someone else. It might be argued that the onus is on Yib to lock her room if she doesn't want surprise visitors, but the reality is that people often forget to do this. I can't think of a situation where it would be *un*acceptable to page someone, first, and ask if you may join em.

@sweep

The `@sweep` command checks a room for potential listening devices, (any object that has a `:tell` verb on it). If you own the room, you might `@move` or `@eject` the unwelcome item. If you don't own the room, you might elect to have your conversation somewhere else.

The flip side of `@sweep` is bugging and/or recording conversations when your doing so is not obvious to the participants. Similarly, teleporting silently into the midst of a conversation such that the participants don't realize you're there can also land you in the dog house.

General Awareness

People seem to have a natural tendency towards nosiness. I am continually surprised by the many ways that people find, on MOOs, to look in on and keep track of others' doings. Being aware of some of these may help you make more informed choices about where, when and how you do or say things of a potentially sensitive or confidential nature.

- logging – Most client programs permit people to record some or all of their MOOing session with the option of saving it to a file for later review. It is not unusual for people to do this, typically for their own reference or review at a later time. In my experience, it's the exception rather than the rule for logs to be published without permission, but you should be aware that *anything* you say or do in the presence of another *could* come back to haunt you one day.
- Anyone can tell your MOO age, i.e. how long it has been since you first connected.
- People can view your description even if they aren't in the same room with you.
- Anyone can `@audit` you and view your possessions.
- Whom you're `@gagging` or `@refusing` is publicly accessible.
- People can detect when you look at them. Some people have a message that broadcasts to the room when you look at them.
- It is possible to detect your checking on someone's connection status when you use `@who`.
- Feature Object owners can and do keep track of how often their various feature verbs are called, and theoretically could keep track of who calls them.
- Mailing list owners can detect when you `@read` or `@peek` at messages on a mailing list.
- Player class owners are in a position to snoop in a variety of ways, including intercepting pages that you send or receive, intercepting MOOmail that you send or receive, and listening to conversations. There is usually enough social pressure to keep player class owners from doing these things, but one should be aware of

what's theoretically possible, and understand that this is what's behind the admonition to trust the owner(s) of your player class parent and ancestors.

- Wizards can look at anything on the MOO, including properties and verbs that you have set to unreadable. Wizards can enter locked rooms. Wizards can view your forked tasks. They can (though traditionally do not) read your private MOOmail. Wizards have access to your registration email address(es) and all the site addresses from which you connect. Strictly speaking, if you don't trust the wizards of a particular MOO, then you shouldn't get an account there. It's an interesting paradox, then, that anyone gets an account anywhere, because there really isn't a practical way to assess a group of wizards' trustworthiness and integrity before the fact. As with many things in life, it's a calculated risk.

Theft and Trespass

Sometimes people steal things on a MOO, but in point of fact it is impossible to truly hide a thing's whereabouts. I generally think that complaints of people stealing things are overblown – most of the time you can just `@audit` yourself and `@move` an object you own back to where you want it to be. When you *can't* just `@move` an object you own to a location of your choosing, things get more interesting. Trespass is the opposite problem: It's possible for someone to move something into your inventory or a room you own and make it difficult to get rid of. (Imagine a pernicious "Kick me!" sign.)

@lock

In addition to using the @lock command to keep intruders out of a room, you can also `@lock` objects in place to prevent people from taking them.[12] See `help locking`.

Theft Prevention

Some player classes provide an option that prevents others from moving things out of your inventory. This sounds fine, but becomes problematic if you pick up something that I own, and then I can't move my own object back to where it belongs. (In my experience, this situation has usually turned out to be an oversight on the part of the player class author, and been corrected upon request.)

The way theft prevention works is that when an item leaves a player's inventory, a special verb on that player called `:exitfunc` is called, notifying the player object

[12] On LambdaMOO you can ask the housekeeper to return an item to a specified place when it is no longer in use. This technology enables people to borrow things without your having to keep constant track of them manually and repeatedly put them back where they belong, and is a nice alternative to locking things in place.

that the item is leaving. A theft-preventing `:exitfunc` verb would fork a task and move the object right back again. A civic-minded `:exitfunc` verb would probably check to see if the exit was initiated by the object's owner (and/or a legitimate housekeeping task) and, if so, permit the item to be moved.

The flip side of `@lock` and programmatic theft prevention is taking stuff that doesn't belong to you. Sometimes you just have to guess as to whether an item is one that anyone is welcome to borrow or one that someone will miss if you take it, but be advised that this can be an issue that annoys people.

@eject

This command is for getting rid of things that are unresponsive to `drop` and/or `@move`. The syntax is `@eject <item> from <location>`, and you have to own `<location>` for it to work. (Typical usage would be `@eject <item> from me` or `@eject <item> from here`.) Successively forceful versions of this are `@eject!` and `@eject!!`, which provide the item being moved progressively less information about its being moved, thus giving it less opportunity to move itself right back again. If an item continues returning and `<location>` is a room, you might need to `@lock` your room against the item (`@lock <room> with !<item>`) and *then* `@eject` it.

@ban

This verb prevents a designated player or item from entering *any* room you own, as opposed to `@lock`, which only works on one individual room at a time.

@spurn

This command works like `@ban`, except it prevents a designated item from entering or re-entering your inventory. With a particularly pernicious object, it might be necessary to `@spurn` it before using `@eject`. (`@Spurn` is a relatively recent addition to the anti-trespass verbs available. MOOs based on a LambdaCore dated prior to 1998 probably don't have it.)

The flip side of `@eject`, `@ban`, and `@spurn` is trespass, either in person or with an object. Realize that if a player doesn't want to carry an object of yours or have it in a room e owns, then you are on risky territory if you try to impose your presence or program an object's "stickiness".

Sexual Advances

Some people who MOO like to engage in conversations with overt sexual content, either to shock people in public rooms, or privately, in order to become sexually aroused. If you aren't expecting it, receiving a sexually explicit page can be very disconcerting. So, first, just be mindful that such an event might occur. Second, be aware that most people who ask for MOOsex are looking for a *consenting* partner. A politely paged, "Not interested, thanks anyway," is enough to deter the vast majority of people cruising for action. If someone persists despite your declining politely, your best course of action is probably to utilize the noise abatement procedures detailed above. If you do decide to accept someone's invitation, be prudent: In particular, consider very carefully before giving anyone any real-life information about yourself.

If you are the person looking for a sexual conversation or encounter, you should be prepared (and willing) to take "No" for an answer, and be aware that not everyone is amenable to being approached in this way. I recommend politeness first, lasciviousness later.

Harassment

Different people have different thresholds of annoyance beyond which they feel harassed. Each MOO has its own policies and procedures for dealing with harassment or alleged harassment. On some MOOs, one is advised to contact a wizard. On other MOOs, the wizards explicitly state that they expect players to manage on their own. It is an inconvenient fact that there *are* people who seem to derive enjoyment from annoying others. If you are on a MOO whose wizards encourage you to notify them if someone is harassing you, by all means contact them. If you are on a MOO whose wizards have a more hands-off policy, understand that no amount of programmatic effort can prevent every instance of harassment, particularly a perpetrator's first attempt to harass a particular target. While an assault is never the victim's fault, there are some things that you can do to reduce the chances of being selected as a target, and to minimize the likelihood of repeated occurrences.

- Try asking the person to desist. Sometimes supposed "harassment" is truly unintentional, and it's a shame to escalate something that's just a simple misunderstanding.
- Realize that perpetrators want you to react. This is why they attack people and not inanimate objects. The more distress you show, the more you reward them for their offensive behavior. This is not to say that you shouldn't be upset, but that a flamboyant display of your distress to the perpetrator is likely to generate more unwanted attention from em. The calmer you can manage to seem in the face of a textual attack, the sooner the perpetrator will give up and go pick on someone else. The most effective response is no response at all, as if you had already `@gagged` whoever or whatever was generating the unwelcome text.
- If you feel panicked, `@quit` immediately. This is equivalent to simply hanging up the telephone if you receive an obscene phone call. Letting yourself be hounded

off the MOO may be unpalatable, but in most cases it is effective. In other instances, it may be sufficient simply to relocate to another room.

- Check `@who` *before* you connect. (Most MOOS permit this, though some do not). Return when the person bothering you isn't connected emself, then `@gag`, `@ban`, and `@refuse all from <em> for <several> years`.
- Do everything you can to rise above it, stay out of the fray, and move on. As my mother used to say, "Don't get in a pissing contest with a skunk."

Outing

MOOs vary in their privacy policies, but a good rule of thumb is that if a MOO does not in some way programmatically identify the typists behind the characters (e.g. by incorporating site information into guest descriptions or providing an automated registry of players' email addresses), then it is impolite to disclose information about another player's offline identity or particulars without that person's permission. Unfortunately, there is no program one could write to prevent this from occurring. Your best defense, as always, is to keep private information private, and not disclose it to others.

Summary

MOOers vary in their cultural and behavioral norms and expectations, and MOOs vary in the kinds of interactions that are encouraged and tolerated. Getting along is something of a balancing act. There are few hard and fast rules, but it's hard to go wrong if you treat others with respect, then learn a few techniques to cope effectively with the few troublemakers that are out there. Good luck.