Dynamic Project Collaboration and Control Using the World Wide Web

by

Stuart S. Gaudet

Submitted to the Department of Civil and Environmental Engineering,
in partial fulfillment of the requirements for the degree of

Master of Science in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

October 1995

© 1995 Massachusetts Institute of Technology
All rights reserved

Author .................................................. Civil and Environmental Engineering
October 16, 1995

Certified by ........................................... Professor John R. Williams
Civil and Environmental Engineering
Thesis Supervisor

Accepted by ................................................... Professor Joseph Sussman
Chairman, Departmental Committee on Graduate Students
Civil and Environmental Engineering
Dynamic Project Collaboration and Control Using the World Wide Web

by

Stuart S. Gaudet

Submitted to the Department of Civil and Environmental Engineering on October 16, 1995, in partial fulfillment of the requirements for the Degree of Master of Science in Civil and Environmental Engineering

ABSTRACT

This objective of this project is to provide dynamic information service on the World Wide Web for use in project collaboration and control. It includes an interactive virtual reality environment for project cooperation and automatically distributes updated information via the World Wide Web.

The Web improves the Internet by providing easier use and industry standardization on the Internet. At the same time, the type of information it presents is limited to static files or snapshots of ongoing activity. Though requested information may change after delivery, the client must make periodic and un-prompted file reload requests to see them. For now, dynamic use of the Web requires dynamic client interaction.

This project's server permits dynamic information distribution on the Web by keeping open the communications links with all of its clients and then automatically updating their browsers. It combines the virtual reality environment of an "object oriented multi-user dungeon" (MOO) with the versatile information displays used in the World Wide Web. Dynamic collaboration and control is conducted inside the MOO server while actual project material development continues anywhere in the world on the Web. When conditions change and client notification is required, pertinent information is immediately delivered, without prompting, and without client interaction. Through the MOO, the Web becomes a powerful developmental tool.

Thesis Supervisor: Dr. John R. Williams

Title: Associate Professor of Civil and Environmental Engineering
This thesis is based on a relatively new source of information, the World Wide Web. Most of the background information in this thesis is found on the Web. An attempt has been made to document all the sources used in this project. In the draft document, which is on the World Wide Web, all critical sources are linked into the document as hypertext. In the smooth thesis, the URLs of the source data are credited as sources, with credit given to known authors as due and available.

There are two major problems with this type of documentation. First, information travels quite freely about the Internet. Intellectual property or originality is difficult to prove. There is no guarantee that the credited source is the origin of the information it contains. The second problem is in the transient nature of information on the Web. Nodes are created and deleted as quickly as users can type the words copy or delete. Though credited and in existence at publication, there is no guarantee the credited sources will exist (or even be applicable) as time passes.

There is little one can do to avoid the intellectual property problem. Plagiarized material looks as valid as original material once published on the Web. The transient nature problem will be circumvented through hard copy printouts. The author will maintain printed reference material that will be available upon request.
# Table of Contents

1. Introduction 7  
   1.1. Background 7  
   1.2. Objective 8  
   1.3. Approach 9  
2. The World Wide Web 10  
   2.1. World Wide Web History 10  
   2.2. HyperText Markup Language 11  
   2.3. HyperText Transfer Protocol 13  
      2.3.1. The Common Gateway Interface 14  
   2.4. World Wide Web Browsers 15  
      2.4.1. The Common Client Interface 16  
3. Object Oriented Multi-User Dungeons 19  
   3.1. MOO History 19  
   3.2. LambdaMOO 21  
      3.2.1. Connecting to a MOO 22  
      3.2.2. Moving through a MOO 23  
      3.2.3. Interaction between MOO users 25  
      3.2.4. Interaction with MOO things 26  
   3.3. MOO Overview 27  
4. Existing Web-MOO products 28  
   4.1. Overview 28  
   4.2. ChibaMOO - The Sprawl 28  
   4.3. BioMOO 29  
   4.4. Shortcomings of current products 30  
5. NinjaMOO 32  
   5.1. Background 32  
   5.2. NinjaMOO Communications 32  
      5.2.1. The HTTPd Server daemons 35  
      5.2.2. The moo_map CGI 36  
      5.2.3. The cgi_with_args CGI 38  
      5.2.4. BigBrother 39  
      5.2.5. The $cgilink object 40  
      5.2.6. The $webroom object 41  
      5.2.7. The $web_utils object 42  
      5.2.8. The $webber object 44  
      5.2.9. The $ccilink object 45  
      5.2.10. bigsis 46  
   5.3 "Webbing" through NinjaMOO 47  
   5.4. Project Collaboration with NinjaMOO 63  
6. Conclusions 66
The Project Project home page 119
The Project Project Timeline Web page 120
NinjaMOO Web Display of the library 121
NinjaMOO Goodbye Screen 122
Appendix D NinjaMOO Project Code 123
restart 124
httpd.conf 126
access.conf 129
moo_map.c 130
Imagemap configuration file and associated image maps 137
   First_Room.map 137
   bar.map 137
   iesl.map 138
   library.map 138
   lounge.map 138
cgi_with_args 139
Makefile for BigBrother 142
cgilink.c 143
ioutils.c 149
str_echo.c 151
err_sys.c 152
readline.c 153
ritten.c 155
my_cgi_defs.h 156
ioutils.h 158
The $link object 159
The $cgilink object 164
The $webroom object 168
The $web_utils object 183
The $webber object 187
The $ccilink object 203
Makefile for the bigsis program 204
ccilink.c 205
cci_utils.c 211
cci_utils.h 219
port.h 221
my_defs.h 222
The $webthing object 224
Chapter 1
Introduction

1.1. Background

The creators of the World Wide Web project define the World Wide Web as "the universe of network-accessible information."\(^1\) Users of the World Wide Web (W3, Web, or WWW) see the Web from many perspectives. For some it is a research tool. For others, it is a playground. Almost everyone who has used a Web browser acknowledges the fact that it is a wonderful tool for Internet access and browsing. Through the use of HyperText and standardized information display tools, Web browsers have made accessing information on the Internet as easy as using a mouse. Internet veterans, however, claim that the Web browsers and servers of today are not the final solution.

The primary complaint from veterans is that the information presented by Web browsers is static in nature. Dynamic information is accessible, but for the most part, Web browsers in and of themselves only present static information. Once a HyperText Web document is requested, it is presented by the Web browser and does not change. This is in contrast to the multitude of normal Internet products that are dynamics. There are games, communications programs, telecommunication products, weather tracking programs,

\(^1\)\url{http://www.w3.org/hypertext/WWW/WWW/}.\)
stock tickers, etc., Currently, Web users only get snapshots of these types of products or have to launch a separate, dynamic process on their own machine.

With World Wide Web browsers, communication with the document server ends with document delivery. Innovative developments are forthcoming; but for now, dynamic information display requires dynamic client interaction or a dynamic interface and separate viewer. Though changes may be occur in the Web, the client must request periodic and un-prompted updates to see them.

The Web is improving daily; and the two primary Web browsers (Mosaic and Netscape) now provide the means for servers to deliver periodic updates. By voluntarily holding open a port for communications from the server, these new Web browsers now permit unrequested information service. This project provides the first useful exploitation of this capability for multi-user interaction and collaboration.

1.2. Objective

This objective of this project is to provide dynamic information service on the World Wide Web for use in project collaboration and control. It is to combine the dynamic, user-interactive environment of an "object oriented multi-user dungeon" (MOO) with the versatile information displays/Web browsers used on the World Wide Web. Dynamic activity, communication, and collaboration is conducted
inside a MOO server while actual project development can continue anywhere in the multi-platform, multi-system World Wide Web. A project team should be able to discuss their project, point out developments, conduct research, and receive direction from the project manager(s) through dynamic Web interaction.

1.3. Approach

This thesis takes a background-to-specifics approach to describing the problem. Each critical element is described before the final product is integrated and demonstrated. The most critical elements of this project are the World Wide Web and an object-oriented multi-user dungeon. These element are described, then some existing combinations of the two are explored, and finally this project's implementation is demonstrated.

To maintain a consistent format while discussing examples of operational programs, the following text fonts are used:

- Active processes, MOO objects, and MOO verbs - Helvetica
- Arguments - Times
- Files - Palatino
Chapter 2
The World Wide Web

2.1. World Wide Web History

"The World Wide Web is the universe of network-accessible information" In essence, the Internet or Information Superhighway is the World Wide Web and vice versa. A common mistake is to think of the Web as a series of interlocking (or inter-linking) pages of information. In reality, the name was assigned to a pre-existing network of computers and the information they embody. The innovation implemented by the creators of the W3 project was in the means used to access that information.

The W3 project was officially proposed in March 1989 by Tim Berners-Lee and Robert Cailliau at the CERN (European Laboratory for Particle Physics) in Geneva, Switzerland. The idea was to apply the HyperText principles proposed by Ted Nelson in 1965, and observed in the Hypercard players of the day, to provide "a single user-interface to many large classes of stored information such as reports, notes, data-bases, computer documentation and on-line systems help." Frustrated by their inability to share information (or the clumsiness of successful sharing) across various computer platforms/systems within the CERN network, they proposed the use

---

1 http://www.w3.org/hypertext/WWW/WWW/.
2 For more information on HyperText, see http://www.w3.org/hypertext/WWW/WhatIs.html.
3 http://www.w3.org/hypertext/WWW/Proposal.html.
of HyperText to enable user-friendly information access and display. Once the concept was formalized, they realized it could be applied world-wide and the World Wide Web was born.¹

By the end of 1990, Web browsers were available on two different platforms that displayed HyperText and limited Internet information. Over the next year, the project was discussed at a number of conferences, and Web browsers were released to a limited audience for testing. In January 1992, the line mode browser version 1.1 was released for general consumption via anonymous ftp. Once the potential of the HyperText technique of Web browsing was realized by the Internet public, the Web concept and its products exploded across the Internet.²

2.2. HyperText Markup Language

According to Tim Berners-Lee,

"The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are platform independent. HTML documents are SGML documents with generic semantics that are appropriate for representing hypertext news, mail, documentation, and hypermedia; menus of options; database query results; simple structures documents

¹http://www.w3.org/hypertext/WWW/History.html.
²ibid.
with in-lined graphics; and hypertext views of existing bodies of information.¹

The HyperText Markup Language (HTML) is a versatile data presentation language based on the ISO Standard 8879:1986 Information Processing Text and Office Systems; Standard Generalized Markup Language. It has been used on the Web since 1990.² It now exists in three versions (1.0, 2.0 and 3.0), each building upon the previous version.

HTML documents are text files with "tags" that force special handling of the text or are references to information or documents that lie elsewhere. The text handling tags are used primarily for aesthetics. The tagged references are the real strength of the HyperText system. The embedded references can be evaluated and presented as part of the HTML document or can be highlighted references to information lying elsewhere in the internet.

These highlighted references are also called anchors, links or hyperlinks. By definition, these names do not share the same exact meaning, but conceptually they are similar. Valid references consist of a communications protocol (http, ftp, telnet, gopher, etc.) recognized by the Web browser and its system, and a URL of a file on the Internet that is accessible via that communication protocol. The

²Ibid.
beauty of the World Wide Web, is that it was designed to take advantage of most communications protocols, and current Web browsers are highly successful with the major protocols used today.

Users usually access the information represented by these highlighted references by pointing at the reference and clicking the mouse. This directs the Web browser to request the URL to which it refers via the communication protocol provided. There is no guarantee that the reference is valid, but if it is, the requested information should be returned and displayed by the Web browser or by the appropriate viewer for that information (e.g. an MPEG player for an MPEG file, a Postscript viewer for a Postscript file, etc.). If no viewer is available, the information can be saved to the system disk for future use.

2.3. HyperText Transfer Protocol

HyperText Transfer Protocol (HTTP) is the communications protocol used by World Wide Web clients and servers for information transfer of HTML documents across the internet.

"The HTTP protocol is based upon a request/response paradigm. A requesting program (termed a client) establishes a connection with a receiving program (termed a server) and sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible
body content. The server responds with a status line, including its protocol version and a success or error code, followed by a MIME-like message containing server information, entity metainformation, and possible body content."¹

Since HTTP uses TCP/IP communications to transfer this information across the Internet, it uses a rapid and proven technique for data transfer. It is a lightweight communications protocol that executes with the minimum of server and client resource utilization. The versatility added by HTTP lies primarily with the HTTP server itself. In addition to the "request/response paradigm" nature of the server, HTTP servers can also activate server based programs to generate special responses to client requests. These special programs are use the Common Gateway Interface standard.

2.3.1. The Common Gateway Interface

"The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers."² These programs are commonly called CGIs after the standards they meet. Because these programs are executed by the HTTPd server, they can generate information that is current at the

time of the request. This is in contrast to the nature of HTML documents which are static and are updated at the whim of their creators. This timeliness is helpful when searching ever-changing databases, when checking stock quotes, etc.

An HTTPd server is configured to execute programs residing in its /cgi-bin/ directory. When a CGI command is received, the server analyzes the command string and then executes the CGI with the appropriate arguments. Once started, a CGI carries out its function independently from the HTTPd server that activated it and acts upon the arguments that were passed. Functionally, a CGI can do anything the creator is able to program and willing to provide. Once a CGI has completed its operation, it returns (to the HTTPd server) a full HTML document or a URL reference to another document, and then dies.

2.4. World Wide Web browsers

Web browsers come in various shapes and sizes, but have a few basic functions in common. Functionally, they operate as both the HTTP client and as Web data displays. At a minimum they must present the text portion of HTML documents and provide for hyperlink requests. Most modern Web browsers will present textual information and embedded images and also highlight a document's HyperText links.

Modern Web browsers meet far more than just the basic requirements. Depending upon the type of information requested,
the Web browsers can display almost all of them in a HyperText format. If HTML is used, the Web browsers will interpret the HTML document and will display the information contained therein in a marked-up format. Much like a modern word processor, modern web browsers have a multitude of text formats, and international characters. In addition to the textual content, some browsers can display embedded images, tables, floating images with surrounding text, background images and colors, etc.

Modern Web browsers will also markup other communication protocol data in a HyperText format. An example of an anonymous ftp request is included in Appendix A. The browser interprets the ftp data as messages, directories, and files. It then interprets this data and displays the directories and files as links to the documents represented. This technique also works for gopher, usenet, and other communication protocols. As a result, Web browsers are also a much friendlier way to get around the Internet than the typical text-based technique.

Once the Web browser finishes marking up the Web document from HTML, ftp data, gopher data, etc., its display becomes a powerful data collection tool. HyperText links are highlighted in the display and can provide almost instantaneous access to the data they represent. When the user "clicks" on a link, the Web browser requests the selected URL via its communications protocol. If this URL exists and is accessible, the server of that URL will return the data and the requesting browser will display it in the appropriate format. The
user can then peruse the data presented and continue by selecting another of the proffered links.

The fact that the user must select a link re-emphasizes the static nature of most Web browsing available today. As mentioned in the Introduction, there are developments forthcoming that have the potential to enable a more dynamic environment. This project takes advantage of one of these developments, the NCSA Mosaic Common Client Interface.

2.4.1. The Common Client Interface

The Common Client Interface (CCI) is a recent development released during the Spring of 1995 with Mosaic BETA versions 2.5 and later. This interface allows external applications to use TCP/IP communications to control properly configured NCSA Mosaic browsers.\(^1\) CCIs can be used for conducting Web slide shows, for teaching on the web by controlling student browsers, etc. By keeping a port open for communications from the Web, the Web browser can be updated periodically by any CCI cognizant program that sends the correct packet to the correct port number. That means that a server can update the Web browser when information changes rather than the user having to request updated information periodically and unprompted.

\(^1\)National Center for Supercomputing Applications, "NCSA Mosaic Common Client Interface", available on-line at http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/CCI/cci-spec.html
CCIs are the future for dynamic World Wide Web usage. Through this technique, companies can make announcements to clients, periodic stock quotes can be sent to browsers, weather updates can be sent to news stations, traffic updates can be sent to radio stations. The key development is ability to broadcast over the Web instead of waiting for request for information.
Chapter 3
Object Oriented Multi-User Dungeons

3.1. MOO History

Computer based role-playing games have existed almost as long as college campus computers. One of the most popular forms of interactive role-playing games is the multi-user adventure (MUA) which is a game or environment where many players can interact or compete. Originally, these games attempted to simulate the "Dungeons and Dragons" experience for their players.\(^1\)

"Present day MUAs are all descendants of a single game known as MUD (Multi-User Dungeon...). Although there were early attempts to turn single-player adventures such as Colossal Cave and Zork into multi-player adventures, and there may have been attempts to write MUAs from scratch, these came to nothing or petered out. [MUD] was the first proper, workable multi-user adventure game."\(^2\)

"[MUD] was written by Roy Trubshaw and Richard Bartle at Essex University on a DECsystem-10 mainframe. Trubshaw began in Autumn 1979, and Bartle took over in Summer 1980. Initially, the game was playable only by students at the


\(^{2}\)Ibid.
university and guests using (what was then) EPSS. After a year or so, however, external players began to direct-dial from home using modems, and the game's popularity grew."¹

Hundreds of similar MUAs have appeared over the ensuing years as commercial and free products. As computer connectivity and complexity has increased, so have the capabilities of these games. The MUD, Object-Oriented or MOO is a logical descendant of the MUD, applying modern object-oriented techniques to the MUD concept.²

In the early 1990s, Pavel Curtis of Xerox PARC developed the LambdaMOO server which is accepted as the standard for MOOs today. The LambdaMOO "server has three jobs: accepting and managing network connections from the users, maintaining the database of rooms, objects, and users in a robust and persistent way, and interpreting programs written in the server's embedded programming language, called 'MOO.'"³ In its basic form, the LambdaMOO is just a simple virtual reality. It is the internal MOO programming that determines the true character of an individual MOO. Depending upon the creativity each MOO administrator, a LambdaMOO can be a role-playing game, a simple chat-room, a design studio, etc. They are all unique.

¹Bartle.
²Ibid.
The next step is to expand upon this virtual environment with more modern applications. The Social Virtual Reality Project at Xerox PARC is attempting to use the LambdaMOO server as a springboard to launch other, more advanced forms of virtual reality. The basic server is sufficient, but the text-only interface is antiquated. The project participants anticipate adding audio, video, and windows applications.¹ This will enable greater user interface speed and more powerful information distribution. It may have drawbacks in server loading and the decreased use of imagination, but it is a natural evolution of the programs.

3.2. LambdaMOO

To demonstrate the functionality of a typical MOO, a walk through the original LambdaMOO will be used as an example. The original LambdaMOO environment is available on-line at lambda.parc.xerox.com (13.2.116.36) on port number 8888. For this example, LambdaMOO was reached through an emacs interface.

3.2.1. Connecting to a MOO

To connect to a MOO, telnet to the appropriate port and wait for the prompt. For LambdaMOO at lambda.parc.xerox.com, the following welcome screen appears:1

```
* Welcome to LambdaMOO! *
**********************************************************

Running Version 1.7.8p4 of LambdaMOO

PLEASE NOTE:
LambdaMOO is a new kind of society, where thousands of people voluntarily come together from all over the world. What these people say or do may not always be to your liking; as when visiting any international city, it is wise to be careful who you associate with and what you say. The operators of LambdaMOO have provided the materials for the buildings of this community, but are not responsible for what is said or done in them. In particular, you must assume responsibility if you permit minors or others to access LambdaMOO through your facilities. The statements and viewpoints expressed here are not necessarily those of the wizards, Pavel Curtis, or the Xerox Corporation and those parties disclaim any responsibility for them.

For assistance either now or later, type 'help'.

The lag is approximately 7 seconds; there are 208 connected.
```

To connect as a guest, type "connect guest".

```
>connect guest
Okay,... guest is in use. Logging you in as `Ruddy_Guest'
*** Connected ***
```

LambdaMOO asks what type of environment the user would like to use. In this example, "noisy" is selected. LambdaMOO then describes the room.

---

1All MOO quotes are representative of actual interaction and information displayed while interacting with LambdaMOO on 10/1/95.
Would you like to start in a noisy or quiet environment? A noisy environment will place you where you can get help from others and converse; while a quiet environment will give you a quiet place to read help texts. [Please respond 'noisy' or 'quiet'.]

>noisy

The Coat Closet
The closet is a dark, cramped space. It appears to be very crowded in here; you keep bumping into what feels like coats, boots, and other people (apparently sleeping). One useful thing that you've discovered in your bumbling about is a metal doorknob set at waist level into what might be a door.

There is new news. Type `news` to read all news or `news new` to read just new news.

Type `@tutorial` for an introduction to basic MOOing. If you have not already done so, please type `help manners` and read the text carefully. It outlines the community standard of conduct, which each player is expected to follow while in LambdaMOO.

### 3.2.2. Moving through a MOO

One of the attractive features of MOO environments is having the ability to create, move through, and interact with a virtual reality. The environment is not limited to worldly dimensionality or behavior. The example character "Ruddy_Guest" moves from The Coat Closet to The Hot Tub. Ruddy_Guest is moved by entering the movement commands "out", "southeast", "east", and "tub".

>out

You open the closet door and leave the darkness for the living room, closing the door behind you so as not to wake the sleeping people inside.

The Living Room
It is very bright, open, and airy here, with large plate-glass windows looking southward over the pool to the gardens beyond. On the north wall, there is a rough stonework fireplace. The east and west walls are almost completely covered with large, well-stocked bookcases. An exit in the northwest corner leads to the kitchen and, in a more northerly direction, to the entrance hall. The door into the coat closet is at the north end of the east wall, and at the south end is a sliding glass door leading out onto a wooden deck. There are two sets of couches, one clustered around the fireplace and one with a view out the windows.
You see README for New MOOers, Welcome Poster, a fireplace, Cockatoo, The Birthday Machine, lag meter, and FeyrWulf here.
so_belle (asleep), FireBeard (distracted), The Wretched Lazarus, Tchinek (distracted),
Hammer, Basshead, Notman, Rosy_Guest, Blue_Eyes-2, Will_E, Ruddy_Guest,
Magenta_Guest, HD-Stanton (asleep), PeNuUM bUg, bergzabern, and Copper_Guest are here.

>southeast

The Deck
This is a wooden deck behind the main house, facing southward across the pool to the lush gardens beyond. To the west is the door into the living room and to the north is a sliding glass door into the master bedroom, with a door to a half bath in the western side of the north wall. At the east end of the deck is a large hot tub. How Californian... The sun bakes the wooden boards of the deck, warming the area.
You see Rube Goldberg contraption here.

>east

The Hot Tub Deck
This is the higher portion of the deck, built up around the hot tub. You may enter the tub from here. There is a rumble of machinery beneath you. The sun bakes the wooden boards of the deck, warming the area. The hot tub is uncovered.
You see rose trellis and Jakk here.
Maia* and Stunt_Gerbil (Did I miss something?) are here.

>tub
You enter the soothing waters of the hot tub.

The Hot Tub
The hot tub is made of molded fiberglass: on three sides a bench will seat five comfortably (and ten who are friendly), and on the fourth side there is contoured couch for one luxurious soak. There are two rubber mounted buttons here. You may push either the right or left button. The bright sunlight glinting off the water makes you squint. The underwater light is on. The bubbling jets are on.
You see thermometer, Hoodlum's Hovercraft, Hot Tub Bar, Hoodlum's_Mutt, and Max here.
K-Mart_Warrior (distracted), Lady_Kathrine, Infamous_Hoodlum (President of Ruddy_Guest's fan club), MoonDrop, SlowOne, Red_Guest, NIN, Ruddy_Guest,
Hammer, and Will_E are here.
Aaaahhhh! The water is at that perfect temperature where you can just lie in here forever.

While moving through the MOO, the user can interact with the environment, other users, and other MOO things. For this example, all interaction with other users was deleted to conserve space.

At this point, it is necessary to highlight the contrast between the text-only format of this MOO and the text and images format of the
MOO used in this project. By using a text-only format, the user reads the description and uses his imagination to create an image of the MOO spaces. By adding images, imagination becomes less of a factor and MOO spaces are limited by the imagination and drawing skills of the MOO space creators.

3.2.3. Interaction between MOO users

Interaction between users is one of the most popular and most useful features of MOOs. Each MOO has its own character when it comes to the interaction between users. MOOs use this interaction for a multitude of purposes including research cooperation, software development, and entertainment. LambdaMOO is used primarily for socialization. The following example is a simulation. Normal MOO activity is much more involved and difficult to follow since many players can be carrying on several conversations and a significant lag can occur between comments and responses.

```
Blue_Guest says, "Hi!"
>say "Hi. How are you?"
You say, "Hi. How are you?"
>emote scratches his head. 
Ruddy_Guest scratches his head
Blue_Guest says, "Ouch! Do you need some shampoo for that itch?"
>say "No thanks!"
>You say, "No thanks!"
>splash Blue_Guest
You splash Blue_Guest, and it wipes the water out of its eyes.
```

This example demonstrates three distinct verbs that are executable in The Hot Tub. The commands "say" and "emote" are executable in any MOO room. The verb "splash" is unique to this room (though it could
be added to any room). These commands allow users to simulate actual or imaginary behavior while interacting. Verbs like "splash" are added by the creators of MOO rooms and add a little humor and playfulness to the environment.

3.2.4. Interaction with MOO things

MOO programmers can create almost anything within the realms of imagination. In reality, most MOO creations are closer to reality than one might expect. This may be because it is difficult for more than one user to picture the figments of another's imagination. MOO programmers attach verbs to the rooms, players and things inside the MOO. Again, these actions often simulate real-world behavior. The following example shows Ruddy_Guest ordering a drink from the Hot Tub Bar in The Hot Tub.

> drink whiskey from bar

A whiskey, eh?
Alcoholic? [Enter 'yes' or 'no']

> yes

OK, but don't operate any heavy machinery til you're sober!
You take a whiskey from the Hot Tub Bar.
You take a drink from your whiskey.
You pick up your whiskey and take a sip.
You chug about half your whiskey.
You upend your whiskey and finish it off.
You begin to feel the effects of your whiskey.
Ruddy_Guest sways, and almost loses his balance.
Ruddy_Guest has a complex conversation with himself.
Ruddy_Guest begins to pick up random objects and put them on his head.
Ruddy_Guest mumbles something about cockatoos who can't keep secrets.
You recover from the effects of your whiskey.
Again, all of the irrelevant activity was deleted for conciseness. In this case, the MOO object is the Hot Tub Bar and the verb is order. After the verb is started, it forks a separate process to track the caller (Ruddy_Guest). Periodically the forked task makes a pre-set announcement makes an announcement about the caller's behavior. This type of object and verb is typical of the MOO programming seen in LambdaMOO.

3.3. MOO Overview

A MOO is only as good as its MOO programmers. As a vehicle for cooperation and collaboration, a MOO is a powerful tool. It is a lightweight, capable server that can coordinated connections and communications with hundreds of players anywhere in the Internet world. When properly programmed, a MOO can be used for almost any distributive communication purpose imaginable. For this project, a LambdaMOO package is designed to operate as an environment for project coordination and interaction. The capabilities of this tool have barely been tapped, but are revealed to be enormous.
Chapter 4
Existing Web-MOO Products

4.1. Overview

Combining the World Wide Web with a MOO is not an idea that is unique to this project. Others have tried to combine the two with mixed results. Two of the more successful Web-MOO environments will be discussed for comparison's sake. These are not the only two products, but they are the most extensive and professional Web-MOOs currently found on the World Wide Web. Examples of both are provided in Appendix A.

4.2. ChibaMOO - The Sprawl

ChibaMOO appears to have been the first truly successful Web-MOO to come on-line. It was opened by SenseMedia Publishing in June 1994. It herald's itself as "The World's First Public Access Web Server and Multi Media MOO"\(^1\) and as a "collaborative, interactive fiction designed specifically for the World Wide Web."\(^2\) Its Web browsing capability apparently comes from the internal generation of HTML documents describing each object in the MOO. Through this technique, the user can examine the contents of the MOO without actually participating in the MOO.

\(^{2}\) ibid.
ChibaMOO provides three distinct methods for browsing through the MOO. Users can use either a MOO interface program (like telnet or emacs), a Web browser, or both:

- When using a MOO interface program, the user operates within ChibaMOO in the standard text-only mode.

- When using a Web browser alone, a user does not exist in the MOO, it only examines the HTML documents generated by the MOO. It can examine most MOO objects through this method alone.

- As an authorized user, a user can use both interfaces, but gets more than just twice the capability. These users can actually move their character through the MOO via the Web browser and the MOO interface window keeps up with the activity and allows text-based participation with the MOO.

4.3. BioMOO

BioMOO is another Web-MOO product that was originally established as an environment for collaboration on biology research projects. It emulates the ChibaMOO access and browsing techniques, but also permits guests to use the combined Web browser and MOO interface method. BioMOO adds a couple improvements including thumbnail images of objects, customizable Web pages, and helpful tutorials.

\(^1\)http://bioinfo.weizmann.ac.il:8888/#auth
4.4 Shortcomings of Current Products

The primary shortcoming of these two Web-MOO products is the static nature of the Web browser displays they use. These products perpetuate the original problem addressed in this paper, the Web pages do not change unless the user asks for an update. This means that the Web and text-only displays can get out of synch. Users must watch both windows to stay completely in touch with activity in the MOO. This is a problem in three major areas.

- If the user moves from one room to the next through the text-only interface, the Web page does not change. This limits the user to single-room movement via the Web browser or makes moving a two-step effort where the user moves in the text-based mode (which is required for the @join and @teleport commands) and then looks around with the Web browser to see the Web description of the space.

- Secondly, if the Web browser display gets out of synch with the player's actual, the player can get disoriented. This can happen when the player leaves the room but does not update their browser. They can then attempt to use their Web browser to look at objects that they should no longer be able to see because their player is elsewhere. They may also attempt to move to a room that is accessible on the MOO page, but not to the actual player.
• Thirdly, the surroundings can change without the user noticing the change. When other players come and go, when object are dropped, etc., the user is not updated on their Web browser display. They must watch their text-based interface for these updates.

This thesis solves these problems and improves upon the concept.
Chapter 5
NinjaMOO

5.1. Background

The NinjaMOO project has its roots in the Spring 1995 Computer
Aided Engineering II (1.125) class taught by Professor John Williams
at the Massachusetts Institute of Technology. The Web-MOO course
focus was proposed by Adam Skwersky and was quickly adopted by
the class.\footnote{Skwersky, Adam, "My Proposal for 1.125", March 1995. On-line at:
http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/webpages/adams_web_moo_proposal.html}
By the end of the semester, a Web-browsable MOO was
up and running. This basic project was fine-tuned and expanded by
the author while developing this thesis. Some elements of the
project were re-created, and others were replaced. In the end, the
executable processes of the project were standardized and
consolidated on NinjaMOO while the documentation was moved to the
1.125 Athena locker.

5.2. NinjaMOO Communications

Enabling interaction between a constantly changing environment
(like a MOO) and a Web browser that normally displays static
information was a healthy challenge. The resulting NinjaMOO
product requires both a Web browser interface and a telnet interface
for complete participation, but all movement and visual browsing can
be conducted and observed wholly by the CCI-capable Web browser.

\footnotetext[1]{Skwersky, Adam, "My Proposal for 1.125", March 1995. On-line at:
http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/webpages/adams_web_moo_proposal.html}
To enable the Web-MOO interaction, a number of pieces of code were written to provide coordination between Web browsers and the MOO. The objective was to leave the original products (the Web browsers and LambdaMOO) alone while enabling communication between them. By doing so, we use the standards set by the Web browser and MOO programs rather than setting standards for Web browser and MOO code writers.

This coordination required some method of "handshaking" between the Web browsers and the MOO. Unfortunately, there was no single piece of code that could do everything at once. As a result, three new programs were produced from scratch, another was substantially modified, and a significant (read enormous) amount of programming was conducted inside the MOO.

A focal point of this project was to enable MOO interaction via the Web. To accomplish this interaction, Web browser requests must be sent into and interpreted by the MOO; the MOO must act upon the Web browser originated command; and an HTML document should be returned to the Web browser to advise the user of the outcome of their command. Currently there is no standard internet protocol for users to pass information in and out of MOOs other than through a telnet connection. Overcoming that barrier was a critical element of this project.
From the initiation of a Web browser request to the display of the result of that request on the Web browser, a lot happens. As diagrammed in Figure 1, the information is relayed into the MOO through the HTTPd server, two or three external (to the Web browser and MOO) processes, through a MOO-internal communications and command link object, and into and through the various objects and their verbs inside the MOO. The verb actions are recorded and an HTML document is drafted inside the MOO. That document and its
address are sent out of the MOO through another MOO-internal link into another external process that saves the HTML document and notifies the Web browser of its existence through a CCI. The Web browser then requests that document, the HTTPd server provides it, and the Web browser displays it. From the user's perspective, it simply appears that the command was executed and the result was returned to the Web browser. This process may appear to be extraordinarily complicated, but each and every step is required and will be explained over the course of this chapter.

5.2.1. The HTTPd server daemons

Web browsers take advantage of a number of communications protocols, HTTP being one of the primary protocols used on the World Wide Web today. In order to handle CGIs on ninja-turtle.mit.edu, NCSA HTTPd 1.4.2 was installed and configured to monitor port 8009. This program is configured to operate five server daemons (hereafter called HTTPd) that monitor in-coming HTTP requests and act upon them accordingly. For this project, only three types of HTTP requests are expected (though more are possible):

- A request for an HTML document. In this case the requested document is returned by the HTTPd if it exists (error message returned otherwise).

- A CGI request for the CGI called "moo_map" with accompanying arguments. In this case the HTTPd launches the moo_map CGI. The
moo_map CGI accepts the arguments, determines what command was intended by them, rewords the intended command as a URL (that is really a call to the CGI "cgi_with_args"), and returns that value to the HTTPd. The HTTPd relays that to the Web browser.

- A CGI request for the CGI called "cgi_with_args" with accompanying arguments. In this case the HTTPd launches cgi_with_args. cgi_with_args executes and a "Status: 204 No Response" is relayed from the CGI to requesting Web browser through the HTTPd server.

5.2.2. The moo_map CGI

The moo_map CGI is a modification of the imagemap ANSI C code provided with the NCSA HTTPd code. The moo_map CGI is usually called by an HTTPd with four critical elements in the call. For demonstration purposes, an example is given where the user clicks on the MOOdy Charles MOO room's image map, but not on an image "hot spot":

http://path/cgi-bin/moo_map/bar+104?180,170

This information is relayed by the HTTPd as the path info: bar+104 and the argument 180,170. The moo_map program breaks this down further into four arguments: the mapname "bar", the player number "104" (representing the first_guest MOO object), the x coordinate "180", and the y coordinate "170". The x and y coordinates are the Cartesian coordinates of the spot that was "clicked" on the imagemap. Error
messages are returned at this point if the wrong number of arguments are found.

The moo_map program then operates on this information. It checks the imagemap configuration file for the corresponding mapname (in this case "bar"). The file "imagemap.conf" includes the following information:

<table>
<thead>
<tr>
<th>First_Room</th>
<th>/mit/adam/httpd_1.4.2/imagemaps/First_Room.map</th>
</tr>
</thead>
<tbody>
<tr>
<td>bar</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/bar.map</td>
</tr>
<tr>
<td>library</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/library.map</td>
</tr>
<tr>
<td>lounge</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/lounge.map</td>
</tr>
<tr>
<td>iesl</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/iesl.map</td>
</tr>
</tbody>
</table>

If there is a configuration file for bar (which there is), then moo_map opens and reads that configuration file. The file "bar.map" includes the following information:

<table>
<thead>
<tr>
<th>default +look</th>
</tr>
</thead>
<tbody>
<tr>
<td>rect +out 10,72 106,237</td>
</tr>
<tr>
<td>rect +ask+waiter+for+tequila 187,98 207,142</td>
</tr>
<tr>
<td>circle +ask+waiter+for+brandy 231,127 240,139</td>
</tr>
<tr>
<td>rect +ask+waiter+for+coffee 303,102 331,142</td>
</tr>
<tr>
<td>rect +menu 320,9 372,52</td>
</tr>
<tr>
<td>rect +ask+waiter+for+water 275,124 297,141</td>
</tr>
<tr>
<td>rect +ask+waiter+for+beer 155,54 329,89</td>
</tr>
</tbody>
</table>

moo_map then compares the coordinates that were passed as arguments with the "hot spot" areas specified in the configuration file. If there is a correlating command, moo_map uses it, otherwise moo_map uses the default command.

Once the command has been determined (default for this example) moo_map puts the pieces together into a URL for the Web browser to
call. In this case, it puts together the URL: http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+look, which is returned to the Web browser via the HTTPd. The Web browser then makes a call to that URL. As with all good CGIs, moo_map dies once it has completed its mission.

5.2.3. The cgi_with_args CGI

cgi_with_args is a perl script written to handle movement and visual browsing requests that are intended for the MOO, but sent by a Web browser. When a request is sent from the Web browser, an HTTPd launches cgi_with_args with the command arguments passed by the Web browser. An example (from the previous moo_map example) cgi_with_args call follows:

http://ninja-turtle.mit.edu/cgi-bin/cgi_with_args?104+look

The HTTPd breaks down the command and calls cgi_with_args with two arguments {104, look}. The cgi_with_args process turns that into the single command string "command 104 look" which is passed to BigBrother for handling. cgi_with_args then tells the Web browser (through the HTTPd) that nothing happened (Status: 204 No Response). Although something does happen in the MOO and will be returned to the Web browser eventually, the CGI's work is complete and it dies quietly.
5.2.4. **BigBrother**

The **BigBrother** daemon is an internal communications ANSI C program that is used as a relay from the `cgi_with_args` CGI into the MOO. Its basic function is to accept an incoming command string, break it into its operative parts and pass it on to the MOO.

On start-up, **BigBrother** first opens up a local socket on port number 7787 for direct communications with the MOO. The second half of that socket linkage is activated by opening the `$cgilink` object inside the MOO. The MOO `$cgilink` object is opened with the `"@open $cgilink"` command. This completes the connection between **BigBrother** and the MOO which is maintained ad infinitum. This step is never repeated unless **BigBrother** is terminated and has to be restarted.

After the `$cgilink` is opened, **BigBrother** opens a socket and listens on port number 9801 for commands to be sent from the `cgi_with_args` CGI. It holds that socket open until a message comes through. This is **BigBrother**'s normal condition and is the state it is in when the example message "command 104 look" is relayed.

Once `cgi_with_args` is called and relays a message, **BigBrother** is finally used as intended - as a messenger. For example, when the incoming message "command 104 look" is received, **BigBrother** reads it and breaks it down into the strings "104" and "look". These strings and the termination string " " are then written to the MOO socket where the MOO internal link object `$cgilink` picks up the action. The "command"
portion of the message is used as convention, in case another type of
message is used in the future. For the time being, the "command"
portion of the message is ignored.

After the message is passed to $cgilink, BigBrother closes the socket
with cgi_with_args, opens another socket on port 9801, and returns to
its normal state - listening for commands from the next run of
cgi_with_args.

5.2.5. The $cgilink object

The $cgilink object is a child of the $link object in the MOO database.
It provides a communications connection with BigBrother and
initializes MOO execution of the commands it receives. Tracking a
MOO command from entrance through complete execution is a
painful process. For the example used through this demonstration, a
brief highlight of its execution will be given to demonstrate the
versatility of the MOO and its object-oriented character.

MOO Step 1: Each argument {"104", "look", " "} that is passed by
BigBrother in the on-going example is read one-at-a-time in the
$cgilink:work() verb through the $cgilink:conn object.

MOO Step 2: Once each argument is completely read, the
$cgilink:work() verb forks, the $cgilink:handle_input() verb is called,
and the input argument is passed.
MOO Step 3: The $cgilink:handle_input() verb analyzes the argument that was passed. If it is the "" argument, it calls the $cgilink:parse_packet() verb, otherwise it appends it to the $cgilink:packet_buf list. In this example, the $cgilink:packet_buf list eventually reads {"command": "104", "look"} (where the "command" argument is the initial (and default) value of the $cgilink:packet_buf list) and the $cgilink:parse_packet() verb is called.

MOO Step 4: The $cgilink:parse_packet() verb breaks the $cgilink:packet_buf list down into a targeted object "#104" and a list of commands called commandstr, in this case {"look"}. In most cases, it then calls the do_command() verb on the object representing the room where the targeted object is located. In NinjaMOO this room is a $webroom object.

At this point, the $cgilink's work is essentially complete for this particular command as passed in from the Web. $cgilink continues to monitor its connection for further commands.

5.2.6. The $webroom object

The $webroom object carries out numerous functions. As a child of the $room object, it performs all of the normal $room functions, but it also performs the additional functions necessary to provide good HTML descriptions of itself. This is accomplished through numerous verbs, which can be found in Appendix C. Its primary additional function is to present itself as a useful Web page with links that
permit movement, visual browsing, and some limited command execution. It tracks player movement in and out of the room and updates all of the $webber’s Web browsers accordingly.

A quick demonstration of the $webroom functionality will be presented as a continuation of the example that has been used to track a Web command in and out of the MOO. MOO Step 5 continues the verb execution where the $cgilink object left off.

MOO Step 5: The $webroom:do_command() analyzes the commands passed by $cgilink:parse_packet(). If the command is found to be valid for the targeted object, that object then executes that command. In this case, the command "look" causes the $webroom looks at itself with the $webroom:look_self() verb.

MOO Step 6: The $webroom:look_self() verb puts together a description of the room. That description is sent as text to the user's telnet window and is written as an HTML document on the user's player object (in this case #104 or first_guest) in the property $webber.htmldoc. The $webroom:look_self() verb makes extensive use of various verbs on the $web_utils object to generate and send this Web description.

5.2.7. The $web_utils object

This object is used for centrally maintaining useful utilities for use in Web-MOO actions. Any of the other Web objects can use these
utilities for various functions. In the ongoing example, the $web_utils is used in four mid-level and several low-level functions in developing the description of the room. The mid-level verbs will be highlighted.

MOO Step 7: The first call to $web_utils in this case is the call to the $web_utils:httpbegin() verb. This verb starts generating the htmldoc property on the $webber (in this case #104). At this point htmldoc = "<html>"

MOO Step 8: In describing the room, the $webroom:look_self() verb makes several calls to $web_utils:httpwrite(). This verb adds input arguments to the $webber.htmldoc property. In this case, the verb is used along with some low-level $webroom verbs to add the image of the room, the room description, the exit calls, and lists the contents of the room.

MOO Step 9: After the basic description is completed, the quit button is added with the $web_utils:append_quit_button() verb.

MOO Step 10: The $web_utils:ccisend() verb then finishes the HTML document by adding the </html> tag and calls the $webber:sendhtml() verb. At this point, the $webroom.htmldoc property is now complete:
The sudden dimness makes your eyes hurt. When they finally adjust, you see a smoky bar. There are tables arrayed around the room in dark corners. There is a comparatively brightly lit bar in the center with all manner of bottles arrayed on it. Patrons sit talk smoke and drink. A bevy of waitresses and waiters wander about the room waiting for orders. At the bar the bartender scurries busily to fill drinks. A group of men sit at the bar intently watching a football game on TV and "commenting" on it. Are you thirsty? Ask for a drink...

Obvious exits: <A HREF="http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+out">out</A> to The First Room

You see <A HREF="http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+look+at+tiamo">tiamo</A> here.

5.2.8. The $webber object

The $webber object carries out numerous functions. As a child of the $player object, it performs all of the normal $player functions, but it also performs the additional functions necessary to allow Web-MOO interaction. Each $webber keeps track of important Web information in the form of properties on the object, like the location of the picture of the player (image_url), the player's Web browsing machine address (cci_machine), and developing Web pages (htmldoc). It also has the ability to describe itself ($webber:description_html() and $webber:look_self()), to start "webbing" ($webber:setccimachine()), to list its possessions ($webber:inventory()), and to send Web documents to its
user's Web browser ($webber:sendhtml()). A complete list of the $webber verbs and properties is provided in Appendix C.

MOO Step 11: The $webber:sendhtml() verb adds tags of "html", the internet address of the player's Web browser machine (which is attached to the player through the $webber:setccimachine() verb as the cci_machine property when they login to the MOO), and the player's object number (#104 in this example) to its htmldoc property. It then calls the $ccilink:send_packet() verb with htmldoc as the argument.

5.2.9. The $ccilink object

The $ccilink object is a child of the $link object in the MOO database. It provides a communications connection with bigsis and is the last Web-critical object used in the execution of a MOO command. It has very few unique properties when compared to the $link object and exists separately primarily for test purposes. Among its properties is the connection with bigsis.

MOO Step 12: The $ccilink:send_packet() verb is inherited from its parent $link object. It passes the argument htmldoc to the connection object $ccilink.conn that maintains the connection with the external process bigsis. All arguments are relayed directly through this object to bigsis.
Once all of the htmldoc has been passed to bigsis, the MOO wraps up its action on the original Web commands "104" and "look". From this point on, bigsis, httpd, and the Web browser take over the display of the Web document that was created in the MOO.

5.2.10. bigsis

The bigsis is an ANSI C program that is conceptually quite similar to BigBrother. Like BigBrother, it relays information from one program to another, but it has some additional responsibilities. It saves the HTML description file and acts as a CCI by relaying the HTML description file URL to the Web browser.

On start-up, bigsis first opens up a local socket on port number 7776 for direct communications with the MOO. The second half of that socket linkage is activated by opening the $ccilink object inside the MOO. The MOO $ccilink object is opened with the "@open $ccilink" command. As with BigBrother, this step is never repeated unless bigsis is terminated and has to be restarted.

When a description file is passed to bigsis by the $ccilink, bigsis first breaks it down into its key elements and acts upon them. It first identifies the file as an html file destined for a certain internet address and player number. It then saves the HTML file to a standard filename format. In the ongoing example the file is named moo104.html. All files are saved in the same format with only the player object number in the center to distinguish it.
big sis sends the URL of the new file to the designated internet address on port number 7775. As long as the player has configured his Web browser correctly, it should be listening to exactly that port.

5.3. "Webbing" through NinjaMOO

Once the communications problem was solved, NinjaMOO became a vehicle that serves truly dynamic documents. Through NinjaMOO, users can interact in a MOO based virtual reality that is now supplemented with a user friendly interface. Of course, the best way to experience the potential of NinjaMOO is through its use, but for demonstration purposes some simple examples are provided.
A user first enters NinjaMOO from the NinjaMOO home page. It can be entered by other means, but the home page provides the step-by-step procedures and easy access for logging into the NinjaMOO. Figure 2 illustrates the UNIX workstation display of a player connecting as a guest from the NinjaMOO home page. Note that the example figures are screen dumps from a full UNIX workstation. Larger (more legible) duplicate images can be found in Appendix C.
When the login is complete, The First Room page appears automatically in the Web browser. This is illustrated in Figure 3. At this stage, the player is fully immersed in NinjaMOO. At this point, the player can use the Web browser to move into the IESL Offices and The MOOdy Charles, to look at Acme and to quit. All other MOO commands can be passed via the telnet window.
IESL Offices

This is the suite leading to all the offices of everyone working on the 1.125 Project. You see numerous macs, suns, pc's, decs, and more macs. Several doorways lead to the various team-members' offices. If you are on the team and would like to add your office here, just use the @dig command to create your office. Type 'help @dig' to learn the syntax of the command.

Obvious exits: first to The First Room, east to Adam's Office, up to The Attic, climb to EP-3E with Stu at the controls (A P-3C picture was the best I could do), northeast to Dennis Shelden's office, lounge to Lounge/Conference Room, and southeast to southeast.

Figure 4

In this example, the user clicked on the IESL Offices door in The First Room image. This moves the player into the IESL offices. As shown in Figure 4, the Web browser and telnet window both describe the new room in the standard format. Again, all of the exits are highlighted as HyperText links that permit continued movement through NinjaMOO. Each of these links are actually formatted commands for the cgi_with_args CGI. In this case, the user selects the link for the Lounge/Conference Room.
The Lounge/Conference Room (Figure 5) is displayed in the standard format, with links for all of the exits and objects in the room. In this case, all of the objects in the room are also in the picture of the room. They can be examined by clicking on the link, clicking on the appropriate part of the image, or by typing in the "look at ..." command in the telnet window. Project_Project is of particular interest and can be seen on the table in the picture.
Figure 6

An example of the dynamic nature of this project is revealed when another player (named Acme - a second player controlled by the author) enters the room. Both the Web browser display and the telnet window are updated to reflect his arrival. It is significant to note that the user of the Web browser did not have to ask for an update, it was provided automatically. This is one example of the dynamic nature of the NinjaMOO.
As shown in the telnet window, some polite banter is exchanged between the player in this example and Acme. The example player decides to take a look at the Project_PROJECT and its basic Web description and MOO description are given in the appropriate windows. Project_PROJECT is just one of the generic MOO project objects. Project objects are all shown with the URLs of the project's home page, its leader's home page, and a timeline web page. These objects are really just starting points for tracking down information elsewhere in the Web.
The user selected the Project_Project home page which is shown in Figure 8. This page is not a product of NinjaMOO. The Project_Project object offers its URL, but at this point, the user is looking at a document produced by a team member and residing elsewhere on the World Wide Web. This document has links to other documents which are critical to the ongoing project. In this case, they are links to documents relating to the NinjaMOO project.
Figure 9

When the user selects the timeline link, the Web page shown in Figure 9 is revealed. Again, this page is not inside NinjaMOO. This demonstrates the power of this vehicle for project coordination. NinjaMOO is a centralized virtual reality with links to external reality. Users can meet and discuss points of interest while also demonstrating and integrating different elements of a team project.
As shown in the telnet window in Figure 10, the user asks Acme to "add a title to the room pages." Figures 3-7 reveal that the room and object descriptions are currently "untitled..." Acme says, "sure...," and takes care of the problem. The user then clicks on the "warp" link that is on his current room Web page description (in Figure 6) which takes him to the library. As shown in the "Title:" entry, room Web pages now have useful titles, in this case "library for first_guest". This is another demonstration of the ability of users to cooperate inside NinjaMOO for project efforts.
The user then selects "unwarp" to return to the Lounge/Conference Room. Its title is now user-friendly as well. Some words are exchanged with Acme, who decides it's time to go.
After Acme disconnects, the user takes a look at his sleeping character by selecting the "Acme" link. All players are described for the Web browsers and telnet windows in the formats shown in figure 12. The description includes the player's name, its description, its status, what it is carrying, and a link to return the user's gaze back into the room.
In figure 13, first_guest and Acme are in The MOOdy Charles. The Web browser is used by the first_guest character's user. Also in the room are tiamo, Generic WWW URL, MIT_URL, NinjaMOO_URL, and the player named shelden. Acme and the first_guest exchange some words and then Acme asks first_guest if it is O.K. to show him something. The first_guest says, "Yes."
Acme points out the contents of the object named "Generic WWW URL" to the first_guest with the command "show_url url to first_guest" ("url" is an alias for the "Generic WWW URL" object). The results are shown in Figure 14. The Generic WWW URL object is described in the Web browser display and includes a link to that object's corresponding URL. In the telnet window, it is described as well. Its description includes the URL of the object in a textual format in case the user is not using a correctly configured Web browser. Again, this demonstrates the dynamics of NinjaMOO. One user can actually
influence the Web browser of another. NinjaMOO ensures that the Web browser is updated for that user instead of by that user.

When the first_guest user selects the Generic WWW URL link in NinjaMOO's description of the Generic WWW URL, it is shown the HTML document in Figure 15. This URL could point to any object that the programmer felt was relevant. This particular object points to a description of what the URL objects are and how to use them. This object is useful for cooperation on projects. Users can share interesting and relevant URLs with other users who may or may not
be in NinjaMOO at that time. By leaving a URL behind, they advertise it to other users.

![Figure 16](image)

Finally, in a separate session, the first_guest user decides to log out of NinjaMOO. He does so by hitting the quit button shown in Figure 16. He could also log out by typing "@quit" in his telnet window.
Disconnecting from NinjaMOO kills the telnet window and returns the HTML document shown above.

5.4. Project Collaboration with NinjaMOO

By taking advantage of the communications potential of the LambdaMOO product along with the capabilities of the World Wide Web, users can now discuss, control, and develop coordinated products as project teams that are co-located only in the virtual reality of NinjaMOO. As long as the team members have Internet
access, they can work together cooperatively and also independently. NinjaMOO facilitates collaboration by providing a user-friendly cooperative environment. It is still up to the team members to use smart project management and development techniques.

The development of NinjaMOO itself provides an excellent example of the power of MOO-based development. Before the MOO server came on line, the core developers were using email and zwrite as collaboration vehicles. These vehicles were hardly optimum for real communication. Once the MOO server was launched, communication was greatly facilitated, though still limited by the typing capabilities of the team. Instead of attempting to relay ideas through email, team members passed and discussed ideas with all the on-line team members simultaneously. Developers immediately addressed their questions about semantics and specifics.

Cooperation was critical, as four team members worked together to configure the original communications linkage and MOO environment. Before NinjaMOO, team members passed standards and techniques in person or through uncoordinated communications methods. NinjaMOO enabled rapid, on-the-fly, decision making. Team members discussed changes in standards, protocol, and capabilities in the real-time virtual reality of the MOO. Team members spent their most productive development time by far when all four members were on-line and working together. As the more advanced capabilities of NinjaMOO emerged, even more rapid and efficient cooperation was enabled.
With the addition of the World Wide Web element, static information sharing was greatly facilitated as well. Where once team members passed basic discussion issues as text and ongoing development information through email messages, they then started using HTML files as a vehicle of choice. They made detail decisions in the MOO and recorded them in the Web. When they had to reference standards, they did, while they continued to operate simultaneously in the MOO. Finally, the Web and the MOO became almost synonymous and communication began to involve files, images, and HyperText in addition to words. Developers can now share enormous amounts of data easily and intelligently.

The developmental potential of NinjaMOO was recognized through its creation. As NinjaMOO became increasingly powerful, so did the collaboration capabilities of the developers. By working in the MOO, the developers could be physically located anywhere in the world while still cooperating effectively on a single coordinated project. Its true potential is still untested, since it has never been used from start to finish (in its current state) in the development of a coordinated product, but that may be the next step in the NinjaMOO story.
Chapter 6
Conclusions

The objective of this thesis was to provide dynamic information service on the World Wide Web for use in project collaboration and control. To achieve this goal, the dynamic, interactive environment of a MOO was combined with World Wide Wed browsers. NinjaMOO users act and communicate inside the NinjaMOO server while continuing actual project development anywhere in the realm of the World Wide Web. A project team can discuss the project, point out developments, conduct research, and receive direction from the project manager(s) through dynamic Web interaction.

Most popular real-time communications programs have the potential to interact with the Web in a fashion similar to the Web-MOO combination. Before long dynamic Web servers could become the norm rather than the exception, but for now, NinjaMOO stands alone. In the meantime, it is more than just a convenient solution to the static Web problem:

- It is based on popular, tested, and free Internet technology.
- Because of the nature of Web, the project-unique software should operate on any machine that can run an HTTPd Web server.
- Developers can easily customize the MOO database to reflect the character of the team, without affecting the Web-MOO capability.
• Several teams can use the software simultaneously and without distraction. MOO rooms are effectively virtual conference rooms.

NinjaMOO's biggest shortcoming is that it has very little security protection. By reading this paper, the average computer user can figure out how to control not only its own MOO character, but any MOO character. The entire MOO is vulnerable if a capable hacker decides to do any damage. Security is a significant problem, but exceeds the scope of this project. For now, administrators can circumvent it by controlling access to the MOO and the HTML files it generates. Future development will require the elimination of this security problem.

NinjaMOO is not a perfect product. It does not solve all the Web's problems. It is an example of the possibilities developers can explore with the new dynamic capabilities of current Web browsers. It does unleash some of the power of interactive Web cooperation. By enabling dynamic activity in the Web, NinjaMOO not only meets the objectives of this thesis, it enhances the types of services providers can put on the Web. They can use NinjaMOO as a game, as a teaching tool, as an improved chat-room, etc. It may be cliché, but its possibilities are limitless. Ultimately, and for the purposes of this thesis, NinjaMOO is completely successful. It provides a robust, user-friendly, dynamic environment for collaboration and control in the World Wide Web.
Appendix A
World Wide Web Examples
Netscape Navigator 1.1 Beta 3

Please read the release notes appropriate to the version you wish to download:

- MS Windows
- Macintosh
- UNIX

The software is available for evaluation and academic/not-for-profit use (governmental organizations do not fall into the not-for-profit category in this program). You can download the exportable version via anonymous FTP from ftp.netscape.com or ftp2.netscape.com.

A U.S.-only version with very high grade security is also available.

Find out more about Netscape at info@mcom.com.

Copyright © 1995 Netscape Communications Corporation
Typical Web Page With Footnoted Links

Netscape 1.1 Beta 3

NETSCAPE NAVIGATOR 1.1 BETA 3

Please read the release notes appropriate to the version you wish to download:

- MS Windows®
- Macintosh®
- Unix®

The software is available for evaluation and academic/not-for-profit use (governmental organizations do not fall into the not-for-profit category in this program). You can download the exportable version via anonymous FTP from ftp.netscape.com or ftp2.netscape.com.

A U.S.-only version with very high grade security is also available.

Find out more about Netscape at info@mcom.com.

Copyright © 1995 Netscape Communications Corporation

http://www.netscape.com/eng/beta/nn1.1b3/index.html Tuesday, 25-Apr-95 00:03:48 GMT
<TITLE>Netscape 1.1 Beta 3</TITLE>

<TITLE>Netscape 1.1 Beta 3</TITLE>

<CENTER>
<A HREF="/index.html">
<IMG BORDER=0 SRC="/comprod/images/comprod_ban.gif" WIDTH=468 HEIGHT=32></A>

<H3>
<FONT SIZE=+2>N</FONT>ETSCAPE
<FONT SIZE=+2>N</FONT>AVIGATOR 1.1
<FONT SIZE=+2>B</FONT>ETA .3</FONT>
</H3>
</CENTER>

<P>
Please read the release notes appropriate to the version you wish to download:

<UL>
<LI><A href="/assist/relnotes/windows-1.1b3.html">MS Windows</A>
<LI><A href="/assist/relnotes/mac-1.1b3.html">Macintosh</A>
<LI><A href="/assist/relnotes/unix-1.1b3.html">Unix</A>
</UL>

The software is available for evaluation and academic/not-for-profit use
(governmental organizations do not fall into the not-for-profit category in
this program).
You can download the <B>exportable version</B> via anonymous FTP from
<A HREF="ftp://ftp.netscape.com/netscapel.1b3/">ftp.netscape.com</A> or

A <B>U.S.-only version with very high grade security</B> is also <A HREF="/eng/beta/nnl.1b3/Get-US-Version.html">available</A>.

<HR SIZE=4>

<P>

<TITLE>Netscape 1.1 Beta 3</TITLE>

<CENTER>
<A HREF="/misc/bottom.map">
<IMG SRC="/images/navigation_bar.gif" ISMAP BORDER=0 WIDTH=468 HEIGHT=32></A>

<P>

<HR SIZE=4>

</CENTER>

<P>

Find out more about Netscape at <A HREF="mailto:info@mcom.com">info@mcom.com</A>.

Copyright &copy; 1995 Netscape Communications Corporation
</P>
Web browser display of ftp result

*Mosaic* displays anonymous ftp results as user links to directories and files. These links actually represent ftp requests to "cd" (change directory) or "get" respectively.
HTML view of ftp result

The Web browser automatically converts ftp results to HTML.

<H1>FTP Directory /</H1>
<DL>
<DD><A HREF="/"> <IMG SRC="internal-gopher-menu"> .</A>
<DD><A HREF="../"> <IMG SRC="internal-gopher-menu"> ..</A>
<DD><A HREF="/betas.obsolete"> <IMG SRC="internal-gopher-menu"> betas.obsolete</A>
<DD><A HREF="/bin"> <IMG SRC="internal-gopher-menu"> bin</A>
<DD><A HREF="/etc"> <IMG SRC="internal-gopher-menu"> etc</A>
<DD><A HREF="/incoming"> <IMG SRC="internal-gopher-menu"> incoming</A>
<DD><A HREF="/netscape"> <IMG SRC="internal-gopher-menu"> netscape</A>
<DD><A HREF="/private"> <IMG SRC="internal-gopher-menu"> private</A>
<DD><A HREF="/pub"> <IMG SRC="internal-gopher-menu"> pub</A>
<DD><A HREF="/unsupported"> <IMG SRC="internal-gopher-menu"> unsupported</A>
</DL>
Appendix B
Web-MOO Project Examples
VRTV! - The SenseMedia Surfer: The ChibaMOO Papers: Web+MOO=WOO!

SenseMedia Publishing proudly announces The Worlds First Public Access Web Server and Multi Media MOO. There is presently a lot of interest in the World Wide Web. The emergence of reliable versions of WWW clients such as Mosaic for Macintosh, Windows and Unix as well as the proliferation of personal PPP and SLIP dialup internet connections is causing the WWW as well as the internet to grow at an explosive rate. In another part of the internet exist the multi user domains (MUDS). MUDS are often shunned as a simple games that threaten countless academic careers. MUD has grown up and one of the most exciting is MOO (MUD Object Oriented.)

MOO is no longer a game. MOO is a complete virtual society where individuals socialize and interact as well as collaborate to build and create things using a really simple yet powerful object oriented programming language. Recently, there has been a lot of effort to combine the media rich content of the World Wide Web with the dynamic and interactive environment of MOO, creating WOO (Webbed MOO.)

Public Access Web Server

SenseMedia Publishing opened "ChibaMOO - The Sprawl" to the public in June of 1994. The Sprawl is one of the first significant pieces of collaborative, interactive fiction designed specifically for the World Wide Web. More than five hundred people, from all over the world are creating a fictional cyberspace of over one thousand rooms, and several thousands of objects. The Sprawl is constantly changing and growing, adding roughly seven new participants a day. The Sprawl makes publishing on the World Wide Web as accessible to grammar school kids as it is to the largest of media mega corporations.

The Sprawl is based on the same core of code that LambdaMOO defined, enhanced to support WOO Transaction Protocol (WTP.) WTP enhances the standard MOO login mechanism to support HTTP requests. WTP means that every object, room and player description is automagically published on the world wide web. Every object may have a real HTML description and verbs can dynamically create HTML to be returned on the fly. This makes publishing on the World Wide Web as easy as creating and describing an object on the MOO. Browsing The Sprawl is as easy as clicking on objects you want to see, or exits to places you want to go. WOO objects may even contain inlined images and audio. Anyone is welcome to login to The Sprawl and browse or publish their own home page.
ChibaMOO

ChibaMOO consists of three areas, "ChibaMOO" – the internal development system at SenseMedia, "The Sprawl" – on the outskirts of Chiba and "The World." The Sprawl is based on a cyberscape of streets and paths through a virtual city and wilderness. Anyone may create a character here and publish their own home page or build whatever they like to be connected to one of the streets. The World, in contrast, is based on reality, and anyone can log in and describe their favorite place, or somewhere they have been.

All three WOOs feature full web support, as well as interesting developments in MOO to MOO networking and are publicly browsable with a WWW client. The Sprawl and The World are open for the public to log in with TELNET or another MUD client. SenseMedia Publishing will soon be opening Chiba.U dedicated exclusively to teachers and classes of students K–Ph.D.

Multi Media MOO

One of the most exciting aspects of the WOO technology is that it adds the rich content and diverse media types of the World Wide Web to what was previously a text only virtual reality. MOO is designed from the start to support group socializing and collaboration and provides an excellent platform from which to launch interactive World Wide Web applications. Rooms on The Sprawl can have real pictures hanging on the wall, windows to current satellite images and play digitized music.

The Sprawl provides some powerful tools for collaborative web browsing and publishing including a group hotlist where anyone can browse for cool web sites, or file a cool web site for others. Cruising through the WOO is done by clicking on exits in Mosaic. The WOO server magically moves your character in your mud client! Explore the web with your friends, hold user groups, take field trips or publish a class newspaper on the Web for the whole world to see. WOO technology aims to let all the individuals browsing the web alone start interacting and browsing the web together.

Other Web/MOO Systems

Other efforts to further integrate WWW and MOO services are ongoing at Jay’s House MOO, CardiffMOO, Hypertext Hotel and WaxWeb. Recently Hypertext Hotel and WaxWeb have integrated WTP into their system.

WAXweb is the hypermedia version of David Blair's film, "WAX, or the discovery of television among the bees". This MOO–based server, developed as a result of

http://sensemedia.net/about

Printed Mon Oct 2 23:10:59 1995
collaboration between David Blair and Tom Meyer, dynamically formats the 560 MPEG clips from the film, 2000 stills, the text and audio in English, German, French and Japanese, as well as additional material contributed by worldwide collaborators. Forms-based authoring tools let anyone add comments, links, and nodes to the hypermedia, while the MOO lets you talk with other readers or build whole new sections as you walk through the film.

**Conclusion**

WOO is cool because it opens up WWW publishing to a whole new level of internet users. WOO combines HTML publishing with MOO programming to create a dynamic, interactive, publishing system simple enough yet powerful enough for professionals and kids alike. woo woo!

- Try the following URLs with your WWW client:
  - The SenseMedia Surfer – http://sensemedia.net/
  - The Sprawl – http://sensemedia.net/sprawl/

- Try logging in with telnet or your MUD client to:
  - The Sprawl – sprawl.sensemedia.net 7777
  - The World – world.sensemedia.net 1234
  - WaxWeb – bug.village.virginia.edu 7777

**Credits**

The WOO concept was conceived at Picosof Systems, Santa Cruz, CA by Samuel Latt Epstein and Jay Campbell. Paul Kautz, Robert Armstrong and the Wizards of ChibaMOO are responsible for the continued and excellent development of this collaborative integrated media development effort. Rocker, Edge, Ashley, Paul, Crag, Fabuley, NetBlazer, Neuro, et al. Special thanks to Jay Carlson and the folks at JHM as well as Andrew Wilson of CardiffMOO.

Icons: SenseMedia (C) Picosof, Neuromancer (C) W. Gibson, Computer Lib (C) T. Nelson, Wax (C) D. Blair. In no way should the autograph be considered an endorsement, its just cool.

Picosof Systems, est. 1986 is a media and software development company and Node #0 for SenseMedia, providing flat rate unlimited/no idle PPP or SLIP to Santa Cruz, CA for $20/month as well as unique and affordable solutions for WWW publishing. SenseMedia can be reached at 408.335.9400.

Samuel Latt Epstein
Director, Technology – Picosof Systems

http://sensemedia.net/about

Printed Mon Oct 2 23:10:59 1995
The Sprawl Home Page

Sprawl Home Page

welcome to chibamoo - the sprawl, the worlds first public access web server and multi media moo! the sprawl is a collaborative hyperarchical integrated media environment (chime) residing on a silicon graphics media server at sensemedia publishing, santa cruz, ca. the sprawl is a virtual community where all users are given the ability to extend the cyberscape in an unrestricted manner, creating a huge and sprawling virtual reality.

web + moo = woo chibamoo - the sprawl is the result of the woo transaction protocol (wtp) project developed at picosof systems, santa cruz, ca. woo combines the interactivity of text based mos with the media rich content of the world wide web. creating objects which interact with the web, such as a program that lets you see who’s on the moo, is extremely easy to do using woo. You may browse our help system if you are interested in learning how to use and program the moo. Or you may view the
official moo programmer’s manual.

**how to** to connect to the moo, telnet to the sprawl and then type `connect guest`. or, you may browse the moo using your web client. if you already have a sprawl character, you may connect to the authorization system, so you can interact with the moo via the web, and telnet simultaneously.

**web** some of the more interesting web objects in the sprawl:

**objects**
- sprawl home pages
- chiba mailing lists
- sprawl web statistics
- object browser
- chiba graffiti wall
- eye of horus
- collaborative fiction center
- cyberscope
- chiba top ten quota hogs
- chibamoo notebook
- generic object browsers
- moo mail web gateway
- snaggletooth’s curiosity
- shoppe
- arizona’s better object browser

the sprawl has received a netscape communications seal of approval!

*sprawl@sensemedia.net*  
gfx/layout by pnambic
The lights of Chiba City twinkle in the south east distance. Street lights intermittently glow, off to the north. The smell of inexpensive liquor and the whine of cathode ray tubes can be heard coming out of Webster’s Bar and URL. Just over a big tree-covered hill to the south lies the Village Square of Connections, a nearly-complete experimental educational environment. Don’t forget to check out the Visitor’s Center.

Contents: Owen, Snarf, Guest, and Redwood_Guest.

Exits: north (to 100 Chiba Blvd), down (to Webster’s Bar and URL), west (to 100 Pico Ave), Center (to Visitor’s Center), venue (to The SenseMedia Venue), and prog (to The Programmitorium).

[help][who][ChibaMOO][The Sprawl] - SenseMedia Network - smn - info@sensemedia.net
BioMOO Web Gateway Page

BioMOO Web Gateway Page

220-

BioMOO’s Web Gateway Page

If you are already familiar with this system, use one of these quick connect options:

- **Authentication system**
- **Webpass system**
- **Continue anonymously** (via HTTP/0.9)

BioMOO’s multimedia interface lets you see what’s in BioMOO as well as providing an intuitive mechanism for navigating this virtual space and exploring the objects within it. We suggest you open two windows: a web window for perceiving multimedia information, and a telnet window for live conversation with other BioMOO users. To do this, follow the directions for the **Authentication system** (the preferred method), or the **Webpass system** (if your web browser doesn’t support authentication), as described below. If you can’t or don’t choose to open two internet windows at once, you can still use some features of the web interface by continuing anonymously. In the following instructions, items in angle brackets should be replaced with the indicated information, omitting the brackets.

**Important note:** If you are using an outdated web client, it is very possible that some of BioMOO’s web features may not work for you. You might not even be able to connect at all. BioMOO’s web system works with any HTML 2.0 compliant client. It has been tested with NCSA Mosaic and the Netscape Navigator in both Mac and PC versions. There is a useful web resource page that will test if your web browser is the most current version.

**Authentication System**

If your web browser supports it, the authentication system is the easiest to use and most secure way to open a multimedia window into BioMOO. Follow these steps:

1. Establish a real-time text connection to BioMOO using either `telnet` or a specialized client program (if you have one)
2. In the telnet/client window, enter BioMOO as follows:
   1. If you have a registered character at BioMOO, connect using "connect <name> <password>" where `<name>` is your character’s name, and `<password>` is your password (omit the angle brackets).
   2. If you are instead visiting BioMOO as a guest (with a temporary character) connect using "guest <name> <password>" where `<name>` will be your temporary character’s name, and `<password>` will be the password you enter for the web authentication system in the next step (make one up).
3. Then open a multimedia window by activating the authentication system with

   http://bioinfo.weizmann.ac.il:8000
   #webpass
   #auth
   http://bioinfo.weizmann.ac.il:8888/#auth
   http://bioinfo.weizmann.ac.il:8000
   http://bioinfo.weizmann.ac.il:8888/#auth
   http://www.city.net/checkup.cgi
   telnet://bioinfo.weizmann.ac.il:8888
   http://www.math.okstate.edu/~jds/mudfaq-p2.html
   http://bioinfo.weizmann.ac.il:8000

   Printed Mon Oct 2 23:10:24 1995

82
4. Welcome to BioMOO! Feel free to take a look around.

Notice that you will have TWO windows open into BioMOO. One (using telnet or a specialized client) is a real-time, text-only window, and the other (using the web browser) is a multimedia web window. The text window will automatically update when you move through BioMOO using the web window. You can select the "Look Around" button at the top of the web window to refresh that view if you move using the text window instead.

Webpass System

If your web browser doesn’t support authentication, you can connect using the webpass system. To do this:

1. Establish a real-time text connection to BioMOO using either telnet or a specialized client program (if you have one)
2. In the telnet/client window, enter BioMOO as follows:
   1. If you have a registered character at BioMOO, connect using "connect <name> <password>" where <name> is your character’s name, and <password> is your password (omit the angle brackets).
   2. If you are instead visiting BioMOO as a guest (with a temporary character) connect using "guest <name>" where <name> will be your temporary character’s name.
3. Establish your webpass by typing "@webpass" to be given a temporary webpass, or "@webpass <webpass>" to specify one of your own (omit the angle brackets). If you have a registered character, you can set this once and it will be the same the next time you come back.
4. Connect to the multimedia interface by entering your webpass in the field below and pressing "return" or "enter" on your keyboard. If you enter your webpass incorrectly, you will return to this page.
5. Welcome to BioMOO! Feel free to take a look around.

Anonymous Session

If you don’t want to open parallel real-time text and multimedia web windows, or can’t, you can open a web window only and continue anonymously instead. If your web browser supports HTTP/1.0, use the authentication system. Access for HTTP/0.9 web browsers is also supported. Note that not all BioMOO features are available to
BioMOO Web Gateway Page
users without a parallel telnet/client connection, and access is also limited by current usage.

A separate page of instructions is available, which includes some screen captures of sample sessions.

Go to the top of this page.
Return to BioMOO's home page.

Please report any problems to Gustavo Glusman at Gustavo@bioinformatics.ac.il.

http://www.cco.caltech.edu/~mercer/htmls/ConnectingByWeb.html
http://bioinfo.weizmann.ac.il:70/ls/biomo
http://bioinformatics.weizmann.ac.il:70/th/Gustavo/Glusman
mailto:Gustavo@bioinformatics.weizmann.ac.il

Details of BioMOO Lounge Web Display

Thud [guest] in BioMOO (Lounge)

Go to the list of applications.

Look around | Hide map | [Central Room] [Who's in BioMOO?] [Customise-page/
yourself] [objects] | Focus | [Help]

You're at The Lounge.

You see a list of users with bad email addresses and a Guest Book.

Richelieu (idle 1h), Biff [guest] (idle 8m), and Dude [guest] (idle 5m) are here.

Obvious exits: south (to the Central Room), tutorial (to Basic Tutorial Start), and web (to WebMOO Demonstration and Introduction).

The Lounge

A large, silent, dimly illuminated room with lots of people snoring their real lives away...
A door to the south leads out to the Central Room. (type 'south' to exit the Lounge; omit the quotes when you enter the command).
A big sign here reads: Type 'tutorial' to learn the basics of MOO, or if you have a web browser available, type 'web' for a web tutorial.

[Look around] [Top of page]

http://bioinfo.weizmann.ac.il:8000/applist
http://bioinfo.weizmann.ac.il:8000/view#focus
http://bioinfo.weizmann.ac.il:8000/view/11+map+
http://bioinfo.weizmann.ac.il:8000/view/4002_central_room#focus
http://bioinfo.weizmann.ac.il:8000/view/who
http://bioinfo.weizmann.ac.il:8000/view/vct
http://bioinfo.weizmann.ac.il:8000/view/char
http://bioinfo.weizmann.ac.il:8000/view/objbrowser
http://bioinfo.weizmann.ac.il:8000/view/3131#focus
http://bioinfo.weizmann.ac.il:8000/view/3007#focus
http://bioinfo.weizmann.ac.il:8000/view/261#focus
http://bioinfo.weizmann.ac.il:8000/view/107#focus
http://bioinfo.weizmann.ac.il:8000/view/397#focus
http://bioinfo.weizmann.ac.il:8000/view/3684#focus
http://bioinfo.weizmann.ac.il:8000/view/155#focus
http://bioinfo.weizmann.ac.il:8000/view/238#focus
http://bioinfo.weizmann.ac.il:8000/view/159#focus
http://bioinfo.weizmann.ac.il:8000/view/2199#focus
http://bioinfo.weizmann.ac.il:8000/view/4695#focus
#header
http://bioinfo.weizmann.ac.il:8000/view/browsers_or_edit/browse/11
http://bioinfo.weizmann.ac.il:8000/view/#focus
#header
http://bioinfo.weizmann.ac.il:8000/
Details of BioMOO Central Room Web Display

Thud [guest] in BioMOO (Central Room)

Go to the list of applications:

[Customise page]

You are at The Central Room.

You see the Research Directory, a public bulletin board, a Jobs/Postdocs bulletin board, a Lost and Found Box, a road-sign pointing to the special interests room, a tour dispenser (td), and an arrow pointing south to the zoo.

Biff [guest] is here.

Obvious exits: west (to the Foyer), north (to the Lounge), east (to the EMBnet Cafe), northeast (to the Seminar Room), southeast (to the Lab Wing), northwest (to the BioMOO Central Library), southwest (to the Special Interest Area (sia)), and south (to the Bio Center Office Tower Lobby).

The Central Room

A very large, circular room, its ceiling a transparent dome through which sunlight streams in.

A large archway leads west into a foyer.

There are doors leading in all directions, labelled with tasteful signs.
Appendix C
NinjaMOO Examples
NinjaMOO entry page

NinjaMOO

Houston, we have a problem... Two weeks of moo code was lost recently when the server's computer was shutdown for maintenance. Unfortunately, the routine backups hadn't happened due to insufficient disk space. So, if it looks like we've taken a step backwards, I apologize. I'M WORKIN' ON IT!! SS: 9/15/95

Hi! Welcome to NinjaMOO. This MOO is a local experiment in making the World Wide Web a more dynamic environment with multi-user interaction while still using current World Wide Web Browsers such as Mosaic and Netscape. Unfortunately the capabilities of these two browsers are starting to diverge. You may find that these pages are best viewed with Netscape, but to use our OCL system you will need to use Mosaic 2.6 or better. Primary points of contact are Stu Gaudet and Adam Skwarsky. Stu continues to work on the project, but Adam gave us the proposal that led to this project and is a great source of web-moo information.

Step 1: Open Mosaic 2.6 (or newer)

If you haven't already done so, open a Mosaic 2.6 (or newer) browser. For Athena customers, that can be done by going to an Athena prompt and typing at the `%` prompt:

```
add outland:Mosaic-BETA &
```

NinjaMOO entry page
The First Room

This is the entrance point to Ninja-MOO. You see a door (iels) leading to the IESL Suite. If you have not already done so, visit the Suite and dig your own office from there.

Obvious exits: iels to IESL Offices and bar to The MOOdy Charles

Acme is here.
IESL Offices

This is the suite leading to all the offices of everyone working on the 1.125 Project. You see numerous macs, suns, pc's, desks and more macs. Several doorways lead to the various team-members' offices. If you are on the team and would like to add your office here, just use the @dig command to create your office. Type 'help @dig' to learn the syntax of the command.

Obvious exits: first to The First Room, east to Adam's Office, up to The Attic, climb to EP-3E with Stu at the controls (A P-3C picture was the best I could do), northwest to Dennis Sheldon's office, lounge to Lounge/Conference Room, and test the testeden.
NinjaMOO Lounge/Conference Room

You see a spacious lounge that could be doubled as a conference area. Now all it needs is some furniture.

Obvious exits: out to IESL Offices and warp to library

You see chair, couch, lazyboy, loungers, and Project Project here.
NinjaMOO Lounge/Conference Room after Acme arrives
The Project Project
Project Home Page: Stu Gaudet
Project Due Date: 10/6/95

Turn your eyes back to the room.
The Project Project

This project is a follow-on to the Spring '95 L.125 class project on the integration of the World Wide Web and a standard MOO. Since MOOs are used for various purposes, including project collaboration and the exchange of ideas, the combination of the dynamics of a MOO and the World-Wide access and "user-friendliness" of World-Wide Web browsing can result in a powerful tool for intellectual cooperation and project management for users around the world.

This project is endeavoring to achieve several objectives:
- Wrap the NinjaMOO Project
  - Stabilize the basic functionality of NinjaMOO
- Document NinjaMOO
- Expand NinjaMOO to include project collaboration functionality
- Demonstrate the usefulness of the whole thing
- Project Timeline

This page was constructed by Stu Gaudet. If you have any questions about this page or project, feel free to write Stu at: ssgaudet@mit.edu

Last modified 10/5/95.
Project Project Timeline

The timeline for this project is pretty brief:

- Project Due to Prof Williams - 10/6/95
- Project Due to CEE Department 10/13/95

Back to the project homepage

This template was designed by Stu Gaudet. If you have any questions about this page or project, feel free to write: stegaudet@mit.edu

Last modified 10/5/95.
You say, "Right."  
"I'm going to go to library to check it out.
You say, "I'm going to go to library to check it out,"
library
This tiny little library is used to catalog projects that are 
under development. If you are webbing, click on the project icon or 
highlighted name. Otherwise, feel free to look at them through the normal 
methods. Take the project of your choice into the lounge while you are 
working on it, if you like. The AC in here is good for books, but not for 
cyber-noids.

Obvious exits: warp to Lounge/Conference Room
You see proj1, proj2, proj3, and proj4 here.
Lounge/Conference Room
You see a spacious lounge that could be doubled as a 
conference area. Now all it needs is some furniture.
Obvious exits: out to IESL offices and warp to library
You see chair, couch, lazyboy, lounger, and Project:Project here.

A name is here.
"Perfect!"
You say, "Perfect!"
Room says, "Thanks, well, I'm going to take off now.
"Bye"
You say, "Bye"
NinjaMOO display of the Acme character description

Acme
You see a wizard who chooses not to reveal its true appearance. He is sleeping.
Carrying:
backpack
link
unwarp

Generic_Project
cgilink

Turn your eyes back to the room.
The MOOdy Charles

The sudden dimness makes your eyes hurt. When they finally adjust, you see a smoky bar. There are tables arrayed around the room in dark corners. There is a comparatively brightly lit bar in the center with all manner of bottles arrayed on it. Patrons sit talk smoke and drink. A bevy of waitresses and walters wander about the room waiting for orders. At the bar the bartender scurries busily to fill drinks. A group of men sit at the bar intently watching a football game on TV and "commenting" on it. Are you thirsty? Ask for a drink...

Obvious exits: out to The First Room
Demonstration of show_url verb with a generic URL object
Generic WWW URL

This object can be tossed around and shown to other players for project coordination. By showing this to another player, they can click on the URL and see what you are up to on the Web.

Feel free to @create more urls at your leisure. For example:

@create url named MY_URL

You should probably give it a new description, too. Try:
@describe MY_URL as 'Here's my home page'

All urls will default to this page as their actual web url, so you will have to change the actual web url on each moo url you create. To do so, try:
@set MY_URL web_url to "http://......you get the picture...".

You can then show the url to another user as follows:
'show_url MY_URL to other_user'

Now, Use your browser 'back' button to look back into NinjaMoo or type 'look' or in the telnet window.

This page created by Stu Gaudet.
Last modified 9/9/95.
Lounge/Conference Room

You see a spacious lounge that could be doubled as a conference area. Now all it needs is some furniture.

Obvious exits: Out to IESL Offices and up to Library.

You see: Chair, couch, library, lounge, and Project/Project here.

Actor is here.
Demonstration of quit button usage

Take it easy. Hope you had fun!

Return to the NinjaMOO home page
Hi! Welcome to Ninja MOO. This MOO is a local experiment in making the World Wide Web a more dynamic environment with multi-user interaction while still using current World Wide Web Browsers such as Mosaic and Netscape. Unfortunately the capabilities of these two browsers are starting to diverge. You may find that these pages are best viewed with Netscape, but to use our CCI system you will need to use Mosaic 2.6 or better. Primary points of contact are Stu Gaudet and Adam Skwersky. Stu continues to work on the project, but Adam was one of the driving forces behind the success of this project and is a great source of web-moo information.

Step 1: Open Mosaic 2.6 (or newer)

If you haven’t already done so, open a Mosaic 2.6 (or newer) browser. For Athena customers, that can be done by going to an Athena prompt and typing at the ‘%’ prompt:

```
add outland;Mosaic-BETA&
```

Step 2: Listen for CCI commands

On the top left hand side of the new Mosaic window, there is a File menu. In that menu, select 'CCI...'. At the prompt, type in a CCI Port Address of 7775. Hit the Accept Requests radio button. Then hit the OK button.

Step 3: Establish a telnet link

Until a means to display ongoing activity without changing pages is developed, you will have to telnet into the MOO to login and to follow truly dynamic activity. Future use will hopefully be independent of this telnet interface. Now, telnet into NinjaMOO, through this link or by telnet to ninja-turtle.mit.edu:7777.

Step 4: Connect your character

If you are reading this on the same browser that is accepting CCIIs, be warned. After you login to the MOO, this page will change to show you the first room of the MOO. You can always back up to this page later if you so desire. Now, if you already have a NinjaMOO web player, connect in the normal fashion, otherwise, type ‘connect guest’ at the telnet prompt.

Step 5: Look around

1 telnet://ninja-turtle.mit.edu:7777/
2 http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
3 http://home.netscape.com
4 http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/help-on-version-2.6.html
6 http://web.mit.edu/afs/athena.mit.edu/user/us/askwersk/Public/www/adam_home.html
7 telnet://ninja-turtle.mit.edu:7777/

http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/webpages/moo_homindly, 10-Sep-95 18:03:35 GMT
Most of your standard MOO commands can be typed into the telnet window. There are also hypertext links available for moving around and looking at objects. Most of your exploration can be done via the Mosaic browser. More advanced commands still require telnet window entry. Most changes of state will also be displayed on the CCI-listening window. Monitor the telnet window for dynamic activity. Now, type in 'look' in the telnet window, and hit return. Oh yeah, while you are at it, stop by the MOOdy Charles and ask the waitress for a beer!

For project bodies, here are some more facts. If you are having problems with the CGI or CCI links, this is where you'll find some answers.

This project is part of the MIT Computer Aided Engineering II (1.125) class taught by Prof. John R. Williams

This page was constructed by Stu Gaudet and was last modified 9/9/95.
The First Room

This is the entrance point to Ninja-MOO. You see a door (iesl) leading to the IESL Suite. If you have not already done so, visit the Suite and @dig your own office from there.

Obvious exits: iesl→IESL Offices and bar→The MOOdy Charles

Oops, I’m lost. HELP!

QUIT NinjaMOO

---

1 http://ninja-turtle.mit.edu:8009/cgi-bin/moo_map/First_Room+104
2 http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+iesl
3 http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+bar
4 http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+look
5 http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?104+quit
The MOOdy Charles

The sudden dimness makes your eyes hurt. When they finally adjust, you see a smoky bar. There are tables arrayed around the room in dark corners. There is a comparatively brightly lit bar in the center with all manner of bottles arrayed on it. Patrons sit talk smoke and drink. A bevy of waitresses and waiters wander about the room waiting for orders. At the bar the bartender scurries busily to fill drinks. A group of men sit at the bar intently watching a football game on TV and "commenting" on it. Are you thirsty? Ask for a drink...

Obvious exits: out to The First Room

You see tiamo, Generic WWW URL, MIT_URL, and NinjaMOO_URL here.

shelden, Acme, and sixth_guest_Guest are here.

Oops, I'm lost. HELP!
Details of NinjaMOO Web Display of Acme's description

Acme
You see a wizard who chooses not to reveal its true appearance.
He is awake, but has been staring off into space for a minute.
Carrying:
backpack
unwarp
Generic_Project
Poop Room

link
ccilink
gilink

Turn your eyes back to the room.
NinjaMOO

This is the NinjaMOO home page. Check out this URL for some more information.

Turn your eyes back to the room.
Details of NinjaMOO Web Display of sixth_guest's description

sixth_guest’s Guest

By definition, guests appear nondescript.
It is awake and looks alert.

Turn your eyes back to the room.
Details of NinjaMOO Web Display of the MIT_URL object

NinjaMOO Web Display of MIT_URL for first_guest

Acme wants you to take a look at the following object:

MIT's home page URL
Check out this URL for some more information.

Turn your eyes back to the room.
Details of MIT Home Page

The MIT Home Page

Massachusetts Institute of Technology

MIT is an independent, coeducational university located in Cambridge, Massachusetts. For informal information see the MIT Student Information Processing Board's SIPB WWW Server. Our first campus information system was TechInfo.

Spotlight on MIT pages of special interest to new students:

- Undergraduate Education and Student Affairs Office (UESA)
- Beginner's Guide to Athena
- Educational Uses of the World Wide Web at MIT

General Information
Student, faculty, and staff online directory, including homepage URLs. MIT facts, news, visitor information and how to apply to MIT.

Academics and Research
Browse through information on MIT schools, departments, research centers, labs, and programs. Access the MIT Libraries. Read course requirements and the academic calendar, too.

Administration & Services
The Alumni Office, Career Services, and other offices offering services to the MIT community. MIT is currently reengineering its administrative processes.

Computing at MIT
Academic Computing Services, PGP, the W3 Consortium, help lines, exploring the Internet, and other computing services.

Publications
Online publications, newspapers and newsletters, documents, and how to get theses, working papers, and technical reports published at MIT.

Campus Activities & New England
Information on MIT groups, organizations, and events in the Cambridge–Boston area. There's always something to do!

http://web.mit.edu/
Services offered by web.mit.edu

What's new on this server, documents and software to help you publish on the Web, guidelines for appropriate use of MIT's CWIS, forms support and other features.

MIT
77 Massachusetts Avenue
Cambridge, MA 02139-4307
(617) 253 1000

Mail comments and link requests to web-request@mit.edu

Last modified: $Date: 95/08/29 17:36:45 $
Details of NinjaMOO Web Display of IESL Offices

IESL Offices

This is the suite leading to all the offices of everyone working on the 1.125 Project. You see numerous macs, suns, pc’s, decs and more macs. Several doorways lead to the various team–members’ offices. If you are on the team and would like to add your office here, just use the @dig command do create your office. Type ‘help @dig’ to learn the syntax of the command.

Obvious exits: first to The First Room, east to Adam’s Office, up to The Attic, climb to EP–3E with Stu at the controls (A P–3C picture was the best I could do), northwest to Dennis Shelden’s office, lounge to Lounge/Conference Room, and testden to testden

Oops, I’m lost. HELP!

http://ninja-turtle.mit.edu:8009/cgi-bin/moomap/iesl+104
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+first
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+east
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+climb
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+up
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+northwest
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+northwest
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+testden
http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-with-args?104+quit

http://ninja-turtle.mit.edu:8009/webhtml/moo104.html

Sun, 10 Sep 1995 18:41:01 GMT
Adam’s Office

Despite its small size, Adam’s Office seems to take up a lot of space. Perhaps this is because he keeps his room nice and neat?? (NOT!) You notice a small spiral staircase in the northeast corner of the room. You wonder where it leads to.

Obvious exits: iesl to IESL Offices, code to A Place for Coding, and down to Deep Dark Pit

Hacker and Adam are here.

Oops, I’m lost. HELP.*

* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+iesl
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+code
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+down
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+look+at+Hacker
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+look+at+Adam
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+look
* http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args?04+quit

Details of NinjaMOO Web Display of the Lounge/Conference Room

Lounge/Conference Room

You see a spacious lounge that could be doubled as a conference area. Now all it needs is some furniture.

Obvious exits: out to IESL Offices and warp to library

You see chair, couch, lazyboy, loungers, and Project Project here.

Oops, I'm lost. HELP!
The Project Project

Project Home Page

Project Leader: Stu Gaudet Home Page
Project Due Date: 9/18/95 Timeline

Turn your eyes back to the room.
The Project Project

This project is a follow-on to the Spring '95 1.125 class project on the integration of the World Wide Web and a standard MOO. Since MOOs are used for various purposes, including project collaboration and the exchange of ideas, the combination of the dynamics of a MOO and the World-Wide access and "user-friendliness" of World-Wide Web browsing can result in a powerful tool for intellectual cooperation and project management for users around the world.

This project is endeavoring to achieve several objectives:

- Wrap up the NinjaMOO project
  - Stabilize the basic functionality of NinjaMOO
  - Document NinjaMOO
  - Package NinjaMOO for general distribution
- Expand NinjaMOO to include project collaboration functionality
- Demonstrate the usefulness of the whole thing
- Document everything for public consumption

This page was constructed by Stu Gaudet. If you have any questions about this page or project, feel free to write Stu at: ssgaudet@mit.edu

Last modified 8/2/95.
The timeline for this project is pretty brief:

- Project Due to Prof Williams - 9/18/95
- Project Due to CEE Department 9/25/95

Back to the project homepage

This template was designed by Stu Gaudet: if you have any questions about this page or project, feel free to write: ssgaudet@mit.edu

Last modified 9/10/95.
Details of NinjaMOO Web Display of the library

**NinjaMOO Web Display of library for first_guest**

This tiny little library is used to catalog projects that are under development. If you are webbing, click on the project icon or highlighted name. Otherwise, feel free to look at them through the normal methods. Take the project of your choice into the lounge while you are working on it, if you like. The AC in here is good for books, but not for cyber-noids.

Obvious exits: unwarp 'to Lounge/Conference Room

You see proj1, proj2, proj3, and proj4 here.

Oops, I'm lost. HELP!

---

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+unwarp

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+look+at+proj1

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+look+at+proj2

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+look+at+proj3

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+look+at+proj4

http://ninja-turtle.mit.edu:8009/cgi-bin/cgi-withargs?I04+quit

http://ninja-turtle.mit.edu:8009/webhtml/moo104.html

Sun, 10 Sep 1995 18:47:59 GMT
Details of NinjaMOO Goodbye Screen

NinjaMOO Goodbye Screen

Take it easy. Hope you had fun!

Return to the NinjaMOO home page.

1 http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/webpages/moo_home.html

http://ninja-turtle.mit.edu:8009/webhtml/moo104.html
Appendix D
NinjaMOO Project Code

Includes code for the most important communications programs and internal MOO database objects.
restart

Used to restart NinjaMOO with restart webmoo command

#!/bin/csh -f

# Copyright (c) 1992 Xerox Corporation. All rights reserved.
# Portions of this code were written by Stephen White, aka ghond.
# Use and copying of this software and preparation of derivative
# works based upon this software are permitted. Any distribution of
# this software or derivative works must comply with all applicable
# United States export control laws. This software is made available
# AS IS, and Xerox Corporation makes no warranty about the
# software, its performance or its conformity to any specification.
# Any person obtaining a copy of this software is requested to send
# their name and post office or electronic mail address to:
# Pavel Curtis
# Xerox PARC
# 3333 Coyote Hill Rd.
# Palo Alto, CA 94304
# Pavel@Xerox.Com

if ($#argv < 1 || $#argv > 2) then
    echo 'Usage: restart dbase-prefix [port]'
    exit 1
endif

if (! -r $1.db) then
    echo "Unknown database: $1.db"
    exit 1
endif

if (-f $1.log) then
    cat $1.log >> $1.log.old
    rm $1.log
endif

if (-r $1.db.new) then
    mv $1.db $1.db.old
    mv $1.db.new $1.db
    rm -f $1.db.old.Z
    compress $1.db.old &
endif

if (-f $1.log) then
    cat $1.log >> $1.log.old
    rm $1.log
endif
echo `date`: RESTARTED >> $1.log
#unlimit descriptors
cd ./adams/
nohup ./bigsis 1 >> bigsis.log &
    cd ../siblings
    nohup ./BigBrother 7787 9801 >> bigbroth.log &
    cd ..
    nohup ./moo $1.db $1.db.new $2 >>& $1.log &

########################################################################
# $Log: restart,v $
# Revision 1.4 1992/10/23 23:15:21 pavel
# Added copyright notice.
#
# Revision 1.3 1992/08/18 00:34:52 pavel
# Added automatic compression of the .old database and the
# provision of a new log file for each activation, saving the old log file
# on the end of a .log.old file.
#
# Revision 1.2 1992/07/20 23:29:34 pavel
# Trying out an RCS change log in this file to see if it'll work properly.
#
########################################################################
httpd.conf

# This is the main server configuration file. It is best to
# leave the directives in this file in the order they are in, or
# things may not go the way you'd like. See URL
# http://hoohoo.ncsa.uiuc.edu/ for instructions.

# Do NOT simply read the instructions in here without understanding
# what they do, if you are unsure consult the online docs. You have
# been warned.

# NCSA httpd (comments, questions to httpd@ncsa.uiuc.edu)

# ServerType is either inetd, or standalone.

ServerType standalone

# If you are running from inetd, go to "ServerAdmin".

# Port: The port the standalone listens to. For ports < 1023, you will
# need httpd to be run as root initially.

Port 8009

# StartServers: The number of servers to launch at startup. Must be
# compiled without the NO_PASS compile option

StartServers 5

# MaxServers: The number of servers to launch until mimic'ing the
# 1.3 scheme (new server for each connection). These servers will
# stay around until the server is restarted. They will be reused as
# needed, however.
# See the documentation on hoohoo.ncsa.uiuc.edu for more
# information.

MaxServers 20

# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.

# User/Group: The name (or #number) of the user/group to run
# httpd as.
User nobody
Group #-1

# ServerAdmin: Your address, where problems with the server
# should be e-mailed.

ServerAdmin ssgaudet@mit.edu

# ServerRoot: The directory the server's config, error, and log files
# are kept in

ServerRoot /mit/adam/httpd_1.4.2/server_root

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.

ErrorLog logs/error_log

# TransferLog: The location of the transfer log file. If this does not start
# with /, ServerRoot is prepended to it.

TransferLog logs/access_log

# AgentLog: The location of the agent log file. If this does not start
# with /, ServerRoot is prepended to it.

AgentLog logs/agent_log

# RefererLog: The location of the referer log file. If this does not start
# with /, ServerRoot is prepended to it.

RefererLog logs/referer_log

# RefererIgnore: If you don't want to keep track of links from certain
# servers (like your own), place it here. If you want to log them all,
# keep this line commented.

#RefererIgnore servername

# PidFile: The file the server should log its pid to
PidFile logs/httpd.pid
# ServerName allows you to set a host name which is sent back to
# clients for your server if it's different than the one the program
# would get (i.e. use "www" instead of the host's real name).

# Note: You cannot just invent host names and hope they work. The
# name you define here must be a valid DNS name for your host. If
# you don't understand this, ask your network administrator.

ServerName ninja-turtle.mit.edu
access.conf

# access.conf: Global access configuration
# Online docs at http://hoohoo.ncsa.uiuc.edu/
# I suggest you consult them; this is important and confusing stuff.

# /usr/local/etc/httpd/ should be changed to whatever you set
# ServerRoot to.
<Directory /usr/local/etc/httpd/cgi-bin>
Options Indexes FollowSymLinks
</Directory>

# This should be changed to whatever you set DocumentRoot to.
<Directory /usr/local/etc/httpd/htdocs>
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", or "FollowSymLinks"
Options Indexes FollowSymLinks

# This controls which options the .htaccess files in directories can
# override. Can also be "None", or any combination of "Options",
# "FileInfo", "AuthConfig", and "Limit"
AllowOverride All

# Controls who can get stuff from this server.

<Limit GET>
order allow,deny
allow from all
</Limit>

</Directory>

# You may place any other directories you wish to have access
# information for after this one.
moo_map.c

/*
** moo_map.c 1.0 Stu Gaudet 7/29/95 is a simple modification of the
** imagemap.c program provided with the NCSA cgi-src package. The
** below comments document its development.
**
** mapper 1.2
** 7/26/93 Kevin Hughes, kevinh@pulua.hcc.hawaii.edu
** "macmartinized" polygon code copyright 1992 by Eric Haines,
** erich@eye.com All suggestions, help, etc. gratefully accepted!
**
** 1.1 : Better formatting, added better polygon code.
** 1.2 : Changed isnameO(), added config file specification.
**
** 11/13/93: Rob McCool, robm@ncsa.uiuc.edu
**
** 1.3 : Rewrote configuration stuff for NCSA /htbin script
**
** 12/05/93: Rob McCool, robm@ncsa.uiuc.edu
**
** 1.4 : Made CGI/1.0 compliant.
**
** 06/27/94: Chris Hyams, cgh@rice.edu
** Based on an idea by Rick Troth (troth@rice.edu)
**
** 1.5 : Imagemap configuration file in PATH_INFO. Backwards
** compatible.
**
** Old-style lookup in imagemap table:
** <a href="http://foo.edu/cgi-bin/imagemap/oldmap">
**
** New-style specification of mapfile relative to DocumentRoot:
** <a href="http://foo.edu/cgi-bin/imagemap/path/for/new.map">
**
** New-style specification of mapfile in user's public HTML
** directory:
** <a href="http://foo.edu/cgi-bin/imagemap/~username/path/
** for/new.map">
**
** 07/11/94: Craig Milo Rogers, Rogers@ISI.Edu
**
** 1.6 : Added "point" datatype: the nearest point wins. Overrides
** "default".
**
** 08/28/94: Carlos Varela, cvarela@ncsa.uiuc.edu
**
** 1.7 : Fixed bug: virtual URLs are now understood.
**     Better error reporting when not able to open configuration file.
**
** 03/07/95: Carlos Varela, cvarela@ncsa.uiuc.edu
**
** 1.8 : Fixed bug (strcat->sprintf) when reporting error.
**     Included getline() function from util.c in NCSA httpd
**
* /

#include <stdio.h>
#include <string.h>
#if !defined(pyr) && !defined(NO_STDLIB_H)
#include <stdlib.h>
#else
#include <sys/types.h>
#include <ctype.h>
char *getenv();
#endif
#include <sys/types.h>
#include <sys/stat.h>

#define CONF_FILE
"/mit/adam/httpd_1.4.2/server_root/conf/imagemap.conf"

#define MAXLINE 500
#define MAXVERTS 100
#define X 0
#define Y 1
#define LF 10
#define CR 13

int isname(char);
int main(int argc, char **argv)
{
    char input[MAXLINE], *mapname, def[MAXLINE], conf[MAXLINE],
errstr[MAXLINE];
    char *playernum; /*moo_map addition*/
    double testpoint[2], pointarray[MAXVERTS][2];
    int i, j, k;
    FILE *fp;
    char *t;
    double dist, mindist;
    int sawpoint = 0;

    /* make sure the right number of arguments are coming into the
     * program*/
    if (argc != 2)
        servererr("Wrong number of arguments, client may not support
ISMAP.");

    /* get name of requested map */
    mapname=getenv("PATH_INFO");

    /* check to see if a mapname is provided */
    if((!mapname) II (!mapname[0]))
        servererr("No map name given. Please read the <A
HREF="http://hoohoo.ncsa.uiuc.edu/docs/setup/admin/Imagemap.ht
ml">instructions</A>.<P>");

        printf("The following mapname including playernum was used:
%sn", mapname);

    /* the next two lines of code (not in comments) are moo_map
     * additions. the following line finds + which is followed by the
playernum in the moo_map configuration */

    playernum = strchr(mapname, '+');

    /* the following puts an endstring in mapname in the place of the
     * plus sign so that mapname is now chopped off to be only the
name of the map and then moves the playernum pointer to
point at the start of the playernum string. */

    *playernum++='\0';
mapname++; /* gets past the '?' */

if(!(t = strchr(argv[1],',')))
    servererr("Your client doesn't support image mapping properly.");
*t++ = '\0';
testpoint[X] = (double) atoi(argv[1]);
testpoint[Y] = (double) atoi(t);

/*
   * if the mapname contains a '/', it represents a unix path -
   * we get the translated path, and skip reading the configuration
   * file.
   */
if (strchr(mapname,'/')) {
    strcpy(conf,getenv("PATH_TRANSLATED"));
goto openconf;
}

if ((fp = fopen(CONF_FILE, "r")) == NULL){
    sprintf(errstr, "Couldn't open configuration file: %s", CONF_FILE);
    servererr(errstr);
}

while(!getline(input,MAXLINE,fp))) {
    char confname[MAXLINE];
    if(input[0] == '#' || (!input[0]))
        continue;
    for(i=0;isname(input[i]) && (input[i] != ':');i++)
        confname[i] = input[i];
    confname[i] = '\0';
    if(!strcmp(confname,mapname))
        goto found;
}

while(!feof(fp)) {
    struct stat sbuf;
    strcpy(conf,getenv("PATH_TRANSLATED"));
if (!stat(conf,&sbuf) && ((sbuf.st_mode & S_IFMT) == S_IFREG))
    goto openconf;
else
    servererr("Map not found in configuration file.");
}

found:
    fclose(fp);
    while(isspace(input[i]) || input[i] == ':') ++i;

    for(j=0;input[i] && isname(input[i]);++i,++j)
        conf[j] = input[i];
    conf[j] = '\0';

openconf:
    if(!(fp=fopen(conf,"r"))){
        sprintf(errstr, "Couldn't open configuration file: %s", conf);
        servererr(errstr);
    }

while(! (getline(input,MAXLINE,fp))) {
    char type[MAXLINE];
    char url[MAXLINE];
    char num[10];

    if((input[0] == '#' || (!input[0]))
        continue;

        type[0] = '\0';url[0] = '\0';

    for(i=0;isname(input[i]) && (input[i]);i++)
        type[i] = input[i];
    type[i] = '\0';

    while(isspace(input[i])) ++i;
    for(j=0;input[i] && isname(input[i]);++i,++j)
        url[j] = input[i];
    url[j] = '\0';

    if(!strcmp(type,"default") && !sawpoint) {
        strcpy(def,url);
        continue;
    }
k=0;
while (input[i]) {
    while (isspace(input[i]) || input[i] == ',')
        i++;
    j = 0;
    while (isdigit(input[i]))
        num[j++] = input[i++];
    num[j] = '\0';
    if (num[0] != '\0')
        pointarray[k][X] = (double) atoi(num);
    else
        break;
    while (isspace(input[i]) || input[i] == ',')
        i++;
    j = 0;
    while (isdigit(input[i]))
        servererr("Missing y value.");
}
pointarray[k][X] = -1;
if(!strcmp(type,"poly"))
    if(pointinpoly(testpoint,pointarray))
        sendmesg(url,playernum);
if(!strcmp(type,"circle"))
    if(pointincircle(testpoint,pointarray))
        sendmesg(url,playernum);
if(![Y]);
/* If this is the first point, or the nearest, set the default. */
if ((!sawpoint) || (dist < mindist)) {
    mindist = dist;
    strcpy(def,url);
}
sawpoint++;
}
if(def[0])
    sendmesg(def,playernum);
servererr("No default specified.");

sendmesg(char *url, char *playernum):
%" , getenv("SERVER_NAME"),
getenv("SERVER_PORT"));
printf("%s%c%c",called_url,10,10);
printf("This document has moved <A
HREF="%s">here</A>%c",url,10);
exit(1);
}

int pointinrect(double point[2], double coords[MAXVERTS][2])
{
    return ((point[X] >= coords[0][X] && point[X] <= coords[1][X]) &&
        (point[Y] >= coords[0][Y] && point[Y] <= coords[1][Y]))

    int crossings;
    double *p, *stop;
    double tx, ty, y;

    for (i = 0; pgon[i][X] != -1 && i < MAXVERTS; i++)
    {
        numverts = i;
        crossings = 0;
        tx = point[X];
        ty = point[Y];
        y = pgon[numverts - 1][Y];
        p = (double *) pgon + 1;
        if ((y >= ty) != (*p >= ty)) {
            if ((xflag0 = xflag0)
                crossings++;
        }
    else {
            crossings += (*(p - 3) - *(p - 2) - ty) *
                (*(p - 1) - *(p - 3)) / (*(p - *(p - 2))) >= tx;
        }
    }

    inside_flag = crossings & 0x01;
    return (inside_ s[i] = '\0';
        return (feof(f) ? 1 : 0);
    }
    ++i;
}
}
Imagemap configuration file and its associated image maps

imagemap.conf

<table>
<thead>
<tr>
<th>Room</th>
<th>Image Map Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>First_Room</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/First_Room.map</td>
</tr>
<tr>
<td>bar</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/bar.map</td>
</tr>
<tr>
<td>library</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/library.map</td>
</tr>
<tr>
<td>lounge</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/lounge.map</td>
</tr>
<tr>
<td>iesl</td>
<td>/mit/adam/httpd_1.4.2/imagemaps/iesl.map</td>
</tr>
</tbody>
</table>

First_Room.map

default +look

circle +iesl 66,112 75,129  
circle +bar 355,115 359,135  
rect +iesl 47,108 85,158  
rect +bar 335,113 374,164

bar.map

default +look

rect +out 10,72 106,237  
rect +ask+waiter+for+tequila 187,98 207,142  
circle +ask+waiter+for+brandy 231,127 240,139  
rect +ask+waiter+for+coffee 303,102 331,142  
rect +menu 320,9 372,52  
rect +ask+waiter+for+water 275,124 297,141  
rect +ask+waiter+for+beer 155,54 329,89
iesl.map

default +look
poly +first 345,87 344,189 381,225 382,72
rect +east 144,83 191,175
poly +up 86,78 87,197 116,175 115,83
poly +climb 130,1 163,34 317,34 352,2 133,2

library.map

default +look
poly +look+at+proj1 55,26 24,25 26,64 27,72 56,72
poly +look+at+proj2 75,51 60,54 60,70 112,74 112,55 97,55 97,42 75,43
poly +look+at+proj3 86,81 77,125 122,131 120,83
poly +look+at+proj4 109,173 108,160 98,162 97,143 81,140 65,144 63,159 50,162 50,171 101,177

lounge.map

default +look
poly +look+at+lounger 1,142 57,135 56,208 3,218
poly +look+at+chair 101,124 100,201 191,202 190,122 140,122 121,100
poly +look+at+lazyboy 194,89 192,139 215,163 283,153 287,110 263,103 262,86 196,88
rect +look+at+Project_Project 189,230 246,256
cgi_with_args

#!/usr/bin/perl

# This perl script was written to interface between a standard
# WWW browser and NinjaMOO. This script accepts commands sent
# by a web browser, parses them and forwards the commands to
# BigBrother for further processing and submission to the MOO.
# It then sends a 'no response' status back to the
# web browser as the MOO processes the command. The intended
# MOO reaction comes via bigsis and a CCI after the command is
# fully processed and executed.
#
# This apparently clumsy arrangement is established because
# the best way to send commands in and out of the MOO is through
# permanent links, and CGI's are temporary. A
# command passed into the MOO may affect several objects and
# thus may require the ability to 'touch' several browsers,
# rather than just the one that made the cgi command. Finally,
# there is no good way for the MOO to realize it has finished
# processing a cgi command, since the MOO is really just
# passing arguments from verb to verb(s). It doesn't have
# an umpire that realizes everything is done and can then talk
# back to the cgi with one single response.
#
# * Redistribution and use in source and binary forms are permitted
# * provided that this paragraph is included. Any distribution of
# * this software or derivative works must comply with all applicable
# * United States export control laws.
# * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY
# * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT
# * LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY
# * AND FITNESS FOR A PARTICULAR PURPOSE.
# * Any person obtaining a copy of this software is requested
# * to send their name and electronic mail address to:
# ssaudet@mit.edu
# This smooth version compiles code from various free sources.
# Author: Stu Gaudet ssaudet@mit.edu 8/28/95
#*************************************************************************
The following mumbo jumbo is a lot of rough TCP/IP code. Look elsewhere for documentation. The idea is to set up a socket with our BigBrother process for passing the command that came into this process.

```
$port = 9801;  # port for comms with BigBrother
$them = 'localhost';  # $them = 'localhost', literally

$AF_INET = 2;
$SOCK_STREAM = 1;
$SIG{'INT'}='dokill';

$sockaddr = 'S n a4 x8';

chop($hostname = `hostname`);

($name,$aliases,$proto)=getprotobynumber('tcp');
($name,$aliases,$port)=getservbyname($port,'tcp')
  unless $port =~ /\d+/;;

($name,$aliases,$type,$len,$thisaddr)=gethostbyname($hostname);
($name,$aliases,$type,$len,$thataddr)=gethostbyname($them);

$this = pack($sockaddr,$AF_INET,0,$thisaddr);
$that = pack($sockaddr,$AF_INET,$port,$thataddr);

# make the socket filehandle;

if(socket(S,$AF_INET,$SOCK_STREAM,$proto)) {
  #     print "socket ok\n";
} else {
  die $!;
}

if(bind(S,$this)) {
  #     print "bind ok\n";
} else {
  die $!;
}
```

# **********************************************************************************
# The following 'until' routine converts the arguments passed to
# the cgi in the form of a list @ARGV into a string called $cmd.
# For example, if the following call was made,
# 'cgi_with_args 1 2 3' (or from web browser 'cgi_with_args?1+2+3')
# @ARGV would be the list '1,2,3'
# The resulting $cmd would be '1 2 3'
# which is the individual elements from the @ARGV list with a space
# space between each.
#
# That command is then sent to BigBrother (or whatever process is
# monitoring port 9801) as 'command 1 2 3' followed by a newline.
#
# Then a 'No Response' is sent to the web browser that made the
# request
# to let it know that the cgi is done.

$cmd;

until ($index > $#ARGV) {
    if($cmd=="") {
        $cmd=$ARGV[$index];
    } else {
        $cmd=$cmd . " " . $ARGV[$index];
    }
    $index=$index+1;
}

if (connect(S,$that)) {
    select(S); $l=1; # send output to port 9801

    # now send 'command $cmd' to BigBrother (port 9801)
    print "command $cmd\n";

    select(STDOUT); # send output to STDOUT

    # tell browser to do nothing for now
    print "Status: 204 No Response\n\n";
}

#
Makefile for BigBrother

CC = gcc -g -Wall -Wwrite-strings
LIBRARIES = -lm -lnsl
OBJS = cgilink.o ioutils.o str_echo.o err_sys.o readline.o writen.o
COMPILE.c = $(CC) $(CFLAGS) $(CPPFLAGS) -c
HDRS = my_cgi_defs.h ioutils.h

BigBrother: $(OBJS)
    $(CC) $(CFLAGS) $(OBJS) $(LIBRARIES) -o $@

.SUFFIXES: .po

c.po: $(HDRS)
    $(COMPILE.c) -o $@ <$

142
/*cgilink.c  

* Author: Stu Gaudet  
* Date: Smooth copy 8/31/95  
* Purpose: To pass web browser requests from the browser into a MOO  
* Modified 9/1/95 by Adam Skwersky to include socket closing functions  
* Modified 9/6/95 by Stu Gaudet to include disclaimer verbage  
* This program is basically just a relay process. It listens to port 9801 for processes that want to pass commands into a MOO. In this format, it is specifically set up to take commands from the cgi called cgi_with_args used with the NinjaMOO project. It maintains a permanent link with the MOO (NinjaNOO) and accepts transient requests from cgi's that are triggered by MOO surfing browsers.  
* Program should be started like this:  
* BigBrother 7787 9801  
* meaning: work on the moo through port 7787 for the moo and listen to port 9801 for cgi's  
* Redistribution and use in source and binary forms are permitted provided that this paragraph is included. Any distribution of this software or derivative works must comply with all applicable United States export control laws.  
* THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  
* Any person obtaining a copy of this software is requested to send their name and post office or electronic mail address to:  
* Stu Gaudet  
* c/o MIT NROTC Unit  
* MIT Room 20E-125  
* Cambridge, Ma 02139-4307  
* ssgaudet@mit.edu  

*/
#include "my_cgi_defs.h"
#include <time.h>

extern printf();
extern bzero();
extern str_echo();
extern writeln();
extern clock();

void shut_down();     /* a routine to exit the program */
void cci_err(const char *);  /* a routine to print error and shutdown */
void OpenMOOSock(char *argv[]); /*a routine to open a socket with
                        the MOO*/
void OpenCGISock(char *argv[]); /* opens socket for CGIs */

int mooSock=-1, cgiSock=-1, servSock;

void filterCR(char *line); /* a routine to filter out carriage returns */

enum {
    ARG_HOSTNAME=0,
    ARG_MOO_HOSTPORT=1,
    ARG_CGI_HOSTPORT=2
};

void main(int argc, char *argv[])
{
    if (argc!=3){
        cci_err("Invalid number of arguments! Need two. Try: 'BigBrother
7787 9801'.\n");
    }
    OpenMOOSock(argv); /* Opens the socket with the MOO*/
    /*All of the actual work happens in OpenCGISock...
     *It was a lot easier than passing
     *a wad of variables and pointers around...
     */
    OpenCGISock(argv);
}
/**************************Filter out carriage returns***************************/
void filterCR(char *line){
    int count=0;
    
    /*
        * printf("in filterCR\n");
        * AM (\r) usually occurs just before a linefeed, \n so
        * just replace them with a space.
        */
    
    while (line[count]){
        if (line[count]=='\r')
            line[count]=' ';
            count++;
    }
}

/**************************Open socket with the moo***************************/
void OpenMOOSock(char *argv[]){

    /* initialize variables*/
    short moo_hostport = atoi(argv[ARG_MOO_HOSTPORT]); /* socket
    for MOO */
    struct sockaddr_in moo_serv_addr, moo_addr;
    int moo_len;
    char *moo_name=NULL;
    int loop_timer;
    int number_of_tries;

    /*
    * set up and wait for moo cgilink to open
    */
    if ((servSock = socket(AF_INET,SOCK_STREAM,0))<0)
        cci_err("Could not create a socket!\n");

    /*
    * bind local address so clients can send to us.
    */
bzero((char*) &moo_serv_addr, sizeof(moo_serv_addr));
moo_serv_addr.sin_family = AF_INET;
moo_serv_addr.sin_port = htons(moo_hostport);
moo_serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);

/* Big Brother tries to bind a few times before it gives up. */
/* This is in case it crashes and needs to be restarted. This frees
 * you up from retrying until the socket clears. */

loop_timer = clock();
number_of_tries = 0;

while (number_of_tries++< 10){
    if (bind(servSock,(struct sockaddr *) &moo_serv_addr,
            sizeof(moo_serv_addr)) < 0)
    {
        printf("BigBrother cannot bind local address on attempt 
#%d.\n",number_of_tries
    );
        while (((clock()-loop_timer)<2000000){})
            loop_timer = clock();
    } else break;
}

if (number_of_tries==11){
    if (bind(servSock,(struct sockaddr *) &moo_serv_addr,
            sizeof(moo_serv_addr)) < 0)
        cci_err("Cannot bind local address\n");
}

/* Big Brother should have bound with a socket or given up by this point */

/* Now listen to socket. BB does not wait here, it waits at 
accept(xmmxmxm) in if below.*/
listen(servSock,5);
printf("\nMoo Socket is open! CGIlink now listening for the MOO to 
connect.\n");
moo_len = sizeof(moo_addr);
    if ((mooSock = accept(servSock,(struct sockaddr *)
            &moo_addr,&moo_len))<0)
cci_err("Cannot accept a connection!
");
moo_name = inet_ntoa(moo_addr.sin_addr);
printf("Got the MOO connection from hostname and port: %s and %d\n", moo_name, moo_addr.sin_port);
}

/**********************************************************/

void OpenCGISock(char *argv[]){
#define MAXCHARS 160
    short cgi_hostport = atoi(argv[ARG_CGI_HOSTPORT]);
    struct sockaddr_in cgi_serv_addr, cgi_addr;
    int cgi_len;
    char *cgi_name=NULL;
    char mess[MAXCHARS];
    char *comm=NULL;
    char buf[MAXCHARS];
    char quit_command[4] = "quit";
    char QUIT_command[4] = "QUIT";
    char kill_big_brother_command[7] = "kill bb";
    char *what_to_do=NULL;
    if ((servSock = socket(AF_INET,SOCK_STREAM,0))<0)
        cci_err("Could not create a socket!\n");
    bzero((char*) &cgi_serv_addr, sizeof(cgi_serv_addr));
    cgi_serv_addr.sin_family = AF_INET;
    cgi_serv_addr.sin_port = htons(cgi_hostport);
    cgi_serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(servSock,(struct sockaddr *) &cgi_serv_addr, sizeof(cgi_serv_addr)) < 0)
        cci_err("BigBrother cannot bind local address.\n");
    listen(servSock,5);
    cgi_len = sizeof(cgi_addr);
    while(1){
        if ((cgiSock = accept(servSock,(struct sockaddr *) &cgi_addr,&cgi_len))<0)
            cci_err("Cannot accept a connection!\n");
        cgi_name = inet_ntoa(cgi_addr.sin_addr);
        printf("Got a CGI from the following host ann');
        *comm=0;
        printf("The CGI originated message was: %s\n",mess);
        if (!(comm = (char *) strchr(mess,' '))) {
            printf("Recieved a bad packet\n");
        }
    }
goto beginning;
}
comm++;
*(comm-1)=0;
if (!(what_to_do = (char *) strchr(comm,' '))){
    printf("Recieved a bad packet\n");
    goto beginning;
}
what_to_do++;
*(what_to_do-1)=0
exit(0);
#include "my_defs.h"

int readn(int fd, char *ptr, int nbytes)
{
    int nleft, nread;
    nleft = nbytes;
    while (nleft > 0) {
        nread = read(fd, ptr, nleft);
        if (nread < 0)
            return (nread); /* error, return < 0 */
        else if (nread == 0)
            break; /* EOF */
        nleft -= nread;
        ptr += nread;
    }
    return (nbytes - nleft); /* return >= 0 */
}

int writen(int fd, char *ptr, int nbytes)
{
    int nleft, nwritten;

    nleft = nbytes;
    while (nleft > 0) {

nwritten = write(fd,ptr,nleft);
if (nwritten<=0)
    return(nwritten); /*error*/
nleft -= nwritten;
ptr += nwritten;
}
return(nbytes-nleft);
}

int readline(int fd, char *ptr, int maxlen){
    int n,rc;
    char c;
    for(n=1;n<maxlen;n++){
        if((rc=read(fd,&c, 1))== 1) {
            *ptr++=c;
            if (c=='\n')
                break;
        } else if (rc==0){
            if(n==1)
                return(0); /*EOF, no data read*/
            else
                break;
        } else
            return(-1); /* error */
    }
    *ptr = 0; /*terminate ptr*/
    return(n);
}

int writeline(int fd, char *ptr, int len){
    static char NL = '\n';
    int nwritten = written(fd,ptr,len);
    if (nwritten<0)
        return (nwritten); /*some error occurred*/
    /*assume all len chars have been written
    so now add a newline*/
    if ((nwritten = write(fd,&NL,1))<0)
        return (nwritten); /*some error occurred*/
    else
        return (nwritten + len);
}
# Redistribution and use in source and binary forms are permitted
# provided that this paragraph is included. Any distribution of
# this software or derivative works must comply with all applicable
# United States export control laws.
# THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT
# LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY
# AND FITNESS FOR A PARTICULAR PURPOSE.
# Any person obtaining a copy of this software is requested
# to send their name and post office or electronic mail address to:
# Stu Gaudet
c/o MIT NROTC Unit
MIT Room 20E-125
Cambridge, Ma 02139-4307
ssgaudet@mit.edu

#define MAXLINE 512
extern readline();
extern printf();
extern err_dump();
extern writen();
void str_echo(sockfd,message)
    int sockfd;
    char *message;
{
    int num_of_chars_in_msg;
    num_of_chars_in_msg = readline(sockfd, message, MAXLINE);
    printf("In str_echo.c.... After readline, num_of_chars_in_msg =
%d\n",num_of_chars_in_msg);
    if (num_of_chars_in_msg == 0)
        return;
    else if (num_of_chars_in_msg < 0)
        err_dump(":num_of_chars_in_msg was less than zero: str_echo:
readline error");
    if (writen(sockfd, message, num_of_chars_in_msg) !=
num_of_chars_in_msg)
        err_dump(":str_echo: writen error");
    printf("Line read was: %s",message);
    return;
}
err_sys.c

Redistribution and use in source and binary forms are permitted provided that this paragraph is included. Any distribution of this software or derivative works must comply with all applicable United States export control laws.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Any person obtaining a copy of this software is requested to send their name and post office or electronic mail address to:
Stu Gaudet
c/o MIT NROTC Unit
MIT Room 20E-125
Cambridge, Ma 02139-4307
ssgaudet@mit.edu

#include <stdio.h>

extern printf();

void err_sys(message)
char* message;
{
    printf ("in err_sys\n");
    printf ("ERROR %s\n",message);
}

void err_dump(message)
char* message;
{
    printf ("in err_dump\n");
    printf ("ERROR %s\n",message);
}
extern read();

/*
 * read a line from a descriptor
 */
int readline(file_descriptor, ptr_to_char, max_line_length)
register int file_descriptor;
register char *ptr_to_char;
register int max_line_length;
{
    int count_variable, read_func_return;
    char char_read;

    for (count_variable = 1; count_variable < max_line_length; count_variable++)
    {
        if ( (read_func_return = read(file_descriptor, &char_read, 1)) == 1 )
            {
                *ptr_to_char++ = char_read;
            }
if (char_read == 'n')
    break;
}
else if (read_func_return == 0)
{
    if (count_variable == 1)
        return (0); /* EOF, no data read */
    else
        break; /* EOF, some data read */
}
else
    return (-1); /* error */

*ptr_to_char = 0;
return (count_variable);
}
extern write();
/*
 * write n bytes to a descriptor
 */
int writen(fd, ptr, nbytes)
register int fd;
register char *ptr;
register int nbytes;
{
    int nleft, nwritten;
    nleft = nbytes;
    while (nleft > 0)
    {
        nwritten = write(fd, ptr, nleft);
        if (nwritten < 0)
            return (nwritten); /* error return < 0 */
        nleft -= nwritten;
        ptr += nwritten;
    }
    return( nbytes - nleft);
}
my_cgi_defs.h

/******************** my_cgi_defs.h ********************
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that this paragraph is included. Any distribution of
 * this software or derivative works must comply with all applicable
 * United States export control laws.
 * THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT
 * LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 * Any person obtaining a copy of this software is requested
 * to send their name and post office or electronic mail address to:
 Stu Gaudet
 c/o MIT NROTC Unit
 MIT Room 20E-125
 Cambridge, Ma 02139-4307
 ssgaudet@mit.edu
 *
 *******************************************/

#ifndef _MY/cgiDefs_H_
#define _MY/cgiDefs_H_

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/time.h>
#include<arpa/inet.h>
#include"ioutils.h"
#define MOO_SERVER_PORT 7787
#define CGI_SERVER_PORT 7877
int socket(int domain, int type, short protocol);
int bind(int s, void *name, int namelen);
int connect(int s, void *addr, int addrlen);
int listen(int s, int qlen);
int accept(int s, void *addr, int *addrlen);
int close(int s);
int read(int s, char *buffer, unsigned buflen);
int readv(int s, struct iovec *iov, int count);
int recv(int s, void *buffer, int buflen, int flags);
int recvfrom(int s, void *buffer, int buflen, int flags, void *from, int *fromlen);
int recvmsg(int s, struct msghdr *msg, int flags);
int write(int s, const char *buffer, unsigned buflen);
int writev(int s, struct iovec *iov, int count);
int send(int s, void *buffer, int buflen, int flags);
int sendto(int s, void *buffer, int buflen, int flags, void *to, int tolen);
int sendmsg(int s, struct msghdr *msg, int flags);
int select(int width, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
#endif
#ifndef _IOUTILS_H_
#define _IOUTILS_H_  

int readn(int fd, char *ptr, int nbytes);  
int writen(int fd, char *ptr, int nbytes);  
int readline(int fd, char *ptr, int maxlen);  
int writeline(int fd, char *ptr, int len);  

#define writelinestring(fd,ptr) writeline((fd),(ptr),(strlen(ptr)))  

#endif
The $link object

Object ID: #158
Name: link
Parent: generic thing (#5)
Location: Acme (#157)
Owner: Adam (#2)
Flags:

Verb definitions:
  startup
  work
  handle_input
  @start @open
  @close
  parse_packet
  clear_packet
  @transmit
  transmit_packet
  @clearpack
  @setaddress
  description
  send_packet

Property definitions:
  current_packet
  conn
  port
  address
  packet_buf
  linked

Properties:
  key: 0
  aliases: {"link"}
  description: "This is a link object for connecting to external processes."

object_size: {0, 0}
  drop_failed_msg: "You can't seem to drop %t here."
  drop_succeeded_msg: "You drop %t."
  odrop_failed_msg: "tries to drop %t but fails!"
  odrop_succeeded_msg: "drops %t."
  otake_succeeded_msg: "picks up %t."
  otake_failed_msg: ""
  take_succeeded_msg: "You take %t."
  take_failed_msg: "You can't pick that up."
current_packet: 0
cconn: E_INVARG
port: 7778
    address: "ninja-turtle.mit.edu"
packet_buf: {"command"}
linked: 0

#158:"startup" this none none
1:  "Taken from help open_network_connection";
2:  "conn = open_network_connection(this.address, this.port);";
3:  conn = $network:open(this.address, this.port);
4:  this.location:announce_all("Found a conn, ",
$string_utils:print(conn));
5:  this.conn = conn;
6:  "line = read(this.conn);";
7:  this.linked = 1;
8:  this:work();
9:  this.linked = 0;

#158:"work" this none this
1:  while (1)
2:    if (typeof(line = read(this.conn)) != ERR)
3:      fork (0)
4:        this:handle_input(line);
5:      endfork
6:    elseif (line == EINVARG)
7:      this.location:announce_all("Darn! Lost the connection!");
8:      return;
9:    else
10:      this.location:announce_all(this.name, ":Strange Error: ",
$string_utils:print(line));
11:      return;
12:    endif
13:  endwhile
#158:"handle_input" this none this
1:  if (args[1] == "")
2:    this:parse_packet();
3:    this:clear_packet();
4:  else
5:    this.packet_buf = { @this.packet_buf, args[1]};
6:  endif

#158:"@start @open" this none none
1:  this:startup();

> @list #158:@close
#158:"@close" this none none
1:  $network:close(this.conn);

#158:"parse_packet" this none this
1:  this.location:announce_all("entering parse_packet");
2:  this.location:announce_all("packet is ");
3:  this.location:announce_all($string_utils:print(this.packetbuf));
4:  if (this.packet_buf[1] == "tell")
5:    tellobj = toobj(this.packet_buf[2]);
6:    tellobj:tell(tostr(this.packet_buf[3]));
7:  elseif (this.packet_buf[1] == "move")
8:    moveobj = toobj(this.packet_buf[2]);
9:    towhere = toobj(this.packet_buf[3]);
10:   move(moveobj, towhere);
11:  endif

#158:"clear_packet" this none this
1:  this.packet_buf = {};

161
#158:

```
"@transmit" any (on top of/on/onto/upon) this
1: "iobjstr will have the string to send over the link";
2: this.location:announce_all("Sending text: ", dobjstr);
3: notify(this.conn, dobjstr);
```

#158:

```
"transmit_packet" this none this
1: "Call it like this: transmit_packet(apacket); where ";
2: "apacket is a list of lines to send. This verb will ";
3: "send all the lines in apacket and it will append the";
4: "blank-line terminator.";
5: if (typeof(apacket) != LIST)
6: this.location:announce_all("Tried to send a packet that was not in list format.");
7: return;
8: endif
9: apacket = {@apacket, ""};
10: for line in (apacket)
11: notify(this.conn, line);
12: endfor
```

#158:

```
"@clearpack" this none none
1: this:clear_packet();
```

#158:

```
"@setaddress" this (at/to) any
1: "@setaddress <link> to anyaddress [anyport]";
2: address = $string_utils:words(iobjstr);
3: player:tell($string_utils:print(address));
4: totell = "";
5: if (length(address) >= 1)
6: this.address = address[1];
7: totell = "Address set to " + address[1];
8: if (length(address) >= 2)
9: this.port = tonum(address[2]);
10: totell = totell + " and port set to " + address[2];
11: else
12: totell = totell + " and port set to default.";
13: endif
14: endif
15: player:tell(totell ? totell l "Well, set it to what??");
```
#158:\"description\"  
   this none this
1:   prevdesc = pass(@args);
2:   toadd = this.name + " is " + (this.linked ? "connected to " +
      this.address + " on port " + tostr(this.port) + "." | "not connected to
      anything.");
3:   if (typeof(prevdesc) == LIST)
4:      prevdesc = (@prevdesc, toadd);
5:   elseif (typeof(prevdesc) == STR)
6:      prevdesc = {prevdesc, toadd};
7:   else
8:      prevdesc = toadd;
9:   endif
10:  return prevdesc;

#158:\"send_packet\"  
   this none this
1:   if (0)
2:      this.location: announce("Entering ccilink:send_packet");
3:      this.location: announce($string_utils: print(args));
4:   endif
5:   tosend = length(args) > 0 ? args[1] | this.packet_buf;
6:   if (tosend)
7:      for line in (typeof(tosend) != LIST ? {tosend} | tosend)
8:         notify(this.conn, tostr(line));
9:   endfor
10:  notify(this.conn, ":");
11:  if (length(args) == 0)
12:     this: clear_packet();
13:  endif
14:  return 1;
15: else
16:   return EINVARG;
17: endif

163
The $cgilink object

Object ID:  #163
Name:  cgilink
Parent:  link (#158)
Location:  Acme (#157)
Owner:  Adam (#2)
Flags:

Verb definitions:
- parse_packet
  @testcomm
- find_right_caller
  @testcomm

Properties:
- key: 0
- aliases: {"cgilink"}
  description: "This is a link object for connecting to external processes."
- object_size: {0, 0}
- drop_failed_msg: "You can't seem to drop %t here."
- drop_succeeded_msg: "You drop %t."
- odrop_failed_msg: "tries to drop %t but fails!"
- odrop_succeeded_msg: "drops %t."
- otake_succeeded_msg: "picks up %t."
- otake_failed_msg: ""
- take_succeeded_msg: "You take %t."
- take_failed_msg: "You can't pick that up."
- current_packet: 0
- conn: #4
- port: 7787
- address: "ninja-turtle.mit.edu"
- packet_buf: {}
- linked: 1

#163:"parse_packet"  this none this
1:  if (this.packet_buf[1] == "command")
2:    if (0)
3:       this.location:announce_all($string_utils:print(this.packet_buf));
4:  endif
5:  targetstr = this.packet_buf[2];
6:  targetobj = toobj(targetstr);
7: commandstr = this.packet_buf[3];
8: "need to sub " , "; , and " ; " ;
9: if (commandstr[1] == "\"")
10: commandstr = "say " + commandstr[2..length(commandstr)];
11: elseif (commandstr[1] == ":")
12: commandstr = "emote " + commandstr[2..length(commandstr)];
13: elseif (commandstr[1] == ";")
14: commandstr = "eval " + commandstr[2..length(commandstr)];
15: endif
16: if (is_player(targetobj) && valid(targetobj.location) &&
targetobj in
connected_players())
17: if ($webroom in $object_utils:ancestors(targetobj.location))
18: if (commandstr == "@quit")
19: this:clear_packet();
20: player = targetobj;
21: player:goodbye_screen();
22: boot_player(player);
23: else
24: targetobj.location:do_command(targetobj, commandstr);
25: endif
26: endif
27: endif
28: "What if user is not in a room? a bag?";
29: "Will either get EVERBNF or nothing will happen.";
30: "Maybe we should split work up between the two.";
31: else
32: pass(@args);
33: endif

#163:"@testcomm" any (with/using) this
1: this:clear_packet();
2: this:handle_input("command");
3: this:handle_input(tostr(player));
4: this:handle_input(dobjstr);
5: this:handle_input(""");
6: #2.location:announce_all("cgi:@testcomm: leaving");
#163: "find_right_caller" this (on top of/on/onto/upon) this
1: "rightcaller = find_right_caller(targetobj, dobj, prepstr, iobj, b[6], verb);"
2: "First check if targetobj has verb."
3: "If not, then check room player is in."
4: "If not, then check the direct object."
5: "If not, then check the indirect object.
6: "else, huh."
7: targetobj = args[1];
8: dobj = args[2];
9: prepstr = args[3];
10: iobj = args[4];
11: argset = args[5];
12: found = 0;
13: verb = args[6];
14: checklist = {targetobj, targetobj.location};
15: if (valid(dobj))
16: checklist = {@checklist, dobj};
17: endif
18: if (valid(iobj))
19: checklist = {@checklist, iobj};
20: endif
21: this.location:announce_all("Check list is: ",
$string_utils:print(checklist));
22: this.location:announce_all("argset list is: ",
$string_utils:print(argset));
23: for anobj in (checklist)
24: this.location:announce_all("now checking ", anobj.name);
25: if (!found)
26: for what in ({anobj, @$object_utils:ancestors(anobj)})
27: this.location:announce_all(" checking: ", what.name);
28: if (!found)
29: if ((argspec = verb_args(what, verb)) != E_VERBNF)
30: this.location:announce_all("argspec is ",
$string_utils:print(argspec));
32: if (argspec[2] in argset[2])
anobj == iobj))
34: "YES! FOUND IT!";
35: found = 1;
36: found_obj = anobj;

166
37:    endif
38:    endif
39:    endif
40:    else
41:        this.location:announce_all("Problem getting verb_args,
argspec
= ", $string_utils:print(argspec));
42:    endif
43:    endif
44:    endfor
45:    endif
46:    endfor
47:    if (found)
48:        return found_obj;
49:    else
50:        return $failed_match;
51:    endif
The $webroom object

Object ID: #154
Name: Generic Web Room
Parent: generic room (#3)
Location: Adam (#2)
Owner: Adam (#2)
Flags:
Verb definitions:
  init_for_core
  @testcomm
  find_right_caller
  do_command
  say
  look_self
  image_url
  look
  announce_web
  confunc
  enter_inform_webbers
  enterfunc
  exitfunc
  look_self2
  description_html
  html_tell_exits
  html_tell_contents
  @getimage
  @setimage
  look_back
Property definitions:
  image_url
  ismap
  mapname
Properties:
  key: 0
  aliases: {"Generic Web Room", "gwr"}
  description: ""
  object_size: {0, 0}
  who_location_msg: "%T"
  free_home: 0
  victim_ejection_msg: "You have been expelled from %i by %n."
  ejection_msg: "You expel %d from %i."
  oejection_msg: "%D is unceremoniously expelled from %i by %n."
residents: {}
free_entry: 1
entrances: {}
blessed_object: #-1
blessed_task: 0
exits: {}
dark: 0
ctype: 3
image_url: 
ismap: 0
mapname: ""

#154:"init_for_core" this none this
1: pass(@args);

#154:"@testcomm" any (with/using) this
1: targetstr = 
2: targetobj = toobj(targetstr);
3: commandstr = dobjstr;
4: "need to sub :, ;, and \" ";
5: #2.location:announce_all("Got ", commandstr);
6: b = $string_utils:parse_command(commandstr, targetobj);
7: verb = verbstr = b[1];
8: dobj = b[2][1];
9: dobjstr = b[2][2];
10: prep = b[3][1];
11: prepstr = b[3][2];
12: iobj = b[4][1];
13: iobjstr = b[4][2];
14: args = b[5][1];
15: argstr = b[5][2];
16: player = targetobj;
17: caller = targetobj.location;
18: rightcaller = this:find_right_caller(targetobj, dobj, prepstr, iobj, b[6], verb);
19: #2.location:announce_all("Found verb on: ", $string_utils:print(rightcaller));
20: if (valid(rightcaller))
21: rightcaller:(verb)(@args);
else
    #2.location:announce_all("@testcomm: verb is ", verbstr, " args are ",
    $string_utils:print(args));
#154:
24:  "targetobj.location:huh(@ args);"
25:  $command_utils:do_huh(verbstr, args);
26:  endif

#154:"find_right_caller"  none none none
1:  "rightcaller =
find_right_caller(targetobj,dobj,prepstr,iobj,b[6],verb);"
2:  "First check if targetobj has verb."
3:  "If not, then check room player is in."
4:  "If not, then check the direct object."
5:  "If not, then check the indirect object."
6:  "else, huh."
7:  targetobj = args[1];
8:  dobj = args[2];
9:  prepstr = args[3];
10: iobj = args[4];
11: argset = args[5];
12: found = 0;
13: verb = args[6];
14: checklist = {targetobj, targetobj.location};
15: if (valid(dobj))
16:  checklist = {@checklist, dobj};
17: endif
18: if (valid(iobj))
19:  checklist = {@checklist, iobj};
20: endif
21: for anobj in (checklist)
22:  if (!found)
23:     for what in ({anobj, @$object_utils:ancestors(anobj)})
24:        if (!found)
25:           if ((argspec = verb_args(what, verb)) != E_VERBNF)
27:                 if (argspec[2] in argset[2])
29:                       "YES! FOUND IT!";
30: found = 1;
31: found_obj = anobj;
32: endif
33: endif
34: endif
35: endif
36: endif
37: endfor
38: endif
39: endfor
40: if (found)
41: return found_obj;
42: else
43: return $failed_match;
44: endif

#154: "do_command" this none this
1: targetobj = toobj(args[1]);
2: commandstr = args[2];
3: "need to sub :, ;, and " ";
4: b = $string_utils:parse_command(commandstr, targetobj);
5: verb = verbstr = b[1];
6: "verb = "look";
7: "I use the above command to clear the buffer as necessary";
8: "After one run through, it filters out the problem with";
9: "tracebacks....";
10: "Please inform askwersk@mit.edu of any tracebacks."
11: "copy the entire traceback message and send it to me";
12: "Also, tell me what you were doing."
13: dobj = b[2][1];
14: dobjstr = b[2][2];
15: prep = b[3][1];
16: prepstr = b[3][2];
17: iobj = b[4][1];
18: iobjstr = b[4][2];
19: args = b[5][1];
20: argstr = b[5][2];
21: player = targetobj;
22: caller = targetobj.location;
23: rightcaller = this:find_right_caller(targetobj, dobj, prepstr, iobj, b[6], verb);
if (valid(rightcaller))
    player.is_web_command = 1;
    rightcaller:(verb)(@args);
    player.is_web_command = 0;
else
    "targetobj.location:huh(@args);"
endif

#154:"say" this any this
1: if (0 && $web_utils:is_webber(player))
2:   $web_utils:httpbegin(tonum(player));
3:   $web_utils:httpwrite(player:image_url());
4:   pass(@args);
5:   $web_utils:ccisend(tonum(player));
6: else
7:   pass(@args);
8: endif
9: if (0)
10: "deal with the player that did the command."
11: if (0 && $web_utils:is_webber(player))
12:   $web_utils:httpbegin(tonum(player));
13:   $web_utils:httpwrite(player:image_url());
14:   player:tell("You say, \", argstr, \\"");
15:   $web_utils:httpwrite(tonum(player), "You say, \\
      \" + argstr +\\
      "\"");
16:   $web_utils:ccisend(tonum(player));
17: endif
18: player:tell("You say, \\
      argstr, \\
      "\"");
19: this:announce_web(player.name, "says, \\
      argstr, "\"");
20: endif

#154:"look_self" this none this
1: if ($web_utils:is_webber(player))
2: "we can be sure player is a webber."
3: if (player:is_webbing())
4: "rebuild complete room display"
5: saved = player.recording;
6: player.recording = 1;
7:   pass(@args);
8:   player.recording = saved;
9:   savedtell = player:divert_tell_html();
10:  if (!savedtell)
11:    "was not already diverting, so lets do a diversion";
12:   $web_utils:httpbegin(tonum(player));
13:   "player:divert_tell_html(1);"
14:  if (this.ismap)
15:    $web_utils:httpwrite(tonum(player), "<a href="http://ninja-turtle.mit.edu:8009/cgi-bin/moo_map/" +
16:      this.mapname + "+"
17:      tostr(tonum(player)) + ">");
18:  else
19:    $web_utils:httpwrite(tonum(player), this:image_url());
20:  endif
21:  if (!(args && args[1]))
22:    $web_utils:httpwrite(tonum(player), this:description_html());
23:  endif
24:  if (player.show_exits)
25:    this:html_tell_exits();
26:  endif
27:  "player:divert_tell_html(1);this:tell_contents(setremove(this:contents (player), this.ctype));player:divert_tell_html(0);";
28:  $web_utils:httpwrite(tonum(player), "<p>");
29:  this:html_tell_contents(setremove(this:contents(), player), this.ctype);
30:  $web_utils:append_quit_button(tonum(player));
31:  $web_utils:ccisend(tonum(player));
32:  endif
33:  else
34:    pass(@args);
35:  endif
36:  else
37:    pass(@args);
38:  endif
1: return "<IMG HEIGHT=300 WIDTH=500 SRC="" + this.image_url + """" + (args && args[1] ? " ISMAP" | "") + ">

1: if (dobjstr == "" && !prepstr)
2: this:look_self();
3: elseif (dobjstr == "back")
4: this:look_back();
5: elseif (prepstr !="in" && prepstr !="on")
6: if (!dobjstr && prepstr == "at")
7: dobjstr = iobjstr;
8: iobjstr = "";
9: else
10: dobjstr = dobjstr + (prepstr && (dobjstr && " ") + prepstr);
11: dobjstr = dobjstr + (iobjstr && (dobjstr && " ") + iobjstr);
12: endif
13: dobj = this:match_object(dobjstr);
14: if (!$command_utils:object_match_failed(dobj, dobjstr))
15: if ($web_utils:is_webber(player) && !$web_utils:is_web_readable(dobj))
16: "if dobj is not a $webthing then still try to";
17: "get some cool HTML document.";
18: dobj:look_self();
19: $web_utils:httpbegin(tonum(player));
20: if ($webber.image_url)
21: $web_utils:httpwrite(tonum(player), $webber:imageurl());
22: endif
23: saved = player:divert_tell_html(1);
24: dobj:look_self();
25: player:divert_tell_html(saved);
26: $web_utils:append_look_back();
27: $web_utils:ccisend(tonum(player));
28: else
29: dobj:look_self();
30: endif
31: endif
32: elseif (!iobjstr)
33: player:tell(verb, " ", prepstr, " what?");
34: else
iobj = this:match_object(iobjstr);
if (!$command_utils:object_match_failed(iobj, iobjstr))
if (dobjstr == "")
iobj:look_self();
elseif ((thing = iobj:match(dobjstr)) == $failed_match)
player:tell("I don't see any ", dobjstr, " ", prepstr, " ",
iobj.name, ");
elseif (thing == $ambiguous_match)
player:tell("There are several things ", prepstr, " ", iobj.name, ");
one might call ", dobjstr, ");
else
thing:look_self();
endif
endif
endif

#154:"announce_web" this none this
this:announce(@args);
"now do the web part:";
image_url = $web_utils:is_webber(player) ? player:image_url() | $webber:image_url();
for dude in ($web_utils:all_webbers_in(this))
$web_utils:httpbegin(tonum(dude));
$web_utils:httpwrite(tonum(dude), image_url);
$web_utils:httpwrite(tonum(dude), @args);
$web_utils:ccisend(tonum(dude));
endfor
for dude in (setremove(this:contents(), player))
dude:tell(@args);
endfor

#154:"confunc" this none this
pass(@args);
if (args)
    entering = toobj(args[1]);
else
    entering = $nothing;
endif
webbers = {};
for w in (this.contents)
    if ($web_utils:is_webber(w))
        webbers = [@webbers, w];
    endif
endfor
if (entering != $nothing)
    webbers = $list_utils:setremove_all(webbers, entering);
endif
for w in (webbers)
    if (w in connected_players())
        player = w;
        diverting = w:divert_tell_html();
        if (!diverting)
            "wasn't already diverting, so lets do a diversion";
            $web_utils:httpbegin(tonum(player));
            "player:divert_tell_html(1);";
            if (this.ismap)
                $web_utils:httpwrite(tonum(player), "<a href=""http://ninja-turtle.mit.edu:8009/cgi-bin/moo_map/" + this.mapname + "+" + tostr(tonum(player)) + "">" + this:image_url(1) + "</a>");
            else
                $web_utils:httpwrite(tonum(player), this:image_url());
            endif
        $web_utils:httpwrite(tonum(player), "</h1>" + this:title() + "</p>");
        if (!(args && args[1]))
            $web_utils:httpwrite(tonum(player), this:description_html());
        endif
        if (player.show_exits)
            this:html_tell_exits();
        endif
    endif
$web_utils:httpwrite(tonum(n1sver), "<n>";
38:     this:html_tell_contents(setremove(this:contents(), player),
this.ctype);
39:     if (0)
40:         player:divert_tell_html(1);
41:     this:tell_contents(setremove(this:contents(), player),
this.ctype);
42:         player:divert_tell_html(0);
43:     endif
44:     $web_utils: append_quit_button(tonum(player));
45:     $web_utils:ccisend(tonum(player));
46:     else
47:         this:look_self(player.brief);
48:     endif
49:     endif
50: endfor

#154:"enterfunc" this none this
1:     pass(@args);
2:     this:enter_inform_webbers(args[1]);

#154:"exitfunc" this none this
1:     pass(@args);
2:     this:enter_inform_webbers(args[1]);

#154:"look_self2" this none this
1:     if ($web_utils:is_webber(player))
2:         "we can be sure player is a webber.";
3:     if (player:is_webbing())
4:         "rebuild complete room display";
5:     saved = player.recording;
6:     player.recording = 1;
7:     pass(@args);
8:     player.recording = saved;
9:     savedtell = player:divert_tell_html();
10:     if (!savedtell)
11:         "wasnt already diverting, so lets do a diversion";
12:     $web_utils:httpbegin(tonum(player));
player: divert_tell_html(1);
"$web_utils: httpwrite(tonum(player), this: title())";
$web_utils: httpwrite(tonum(player), this.description_html());
$web_utils: httpwrite(tonum(player), this.image_url());
$web_utils: httpwrite(tonum(player), "<p>");
pass(@args);
player: divert_tell_html(0);
$web_utils: ccisend(tonum(player));
endif
else
pass(@args);
endif
else
pass(@args);
endif

#154:"description_html" this none this
1: desc = this: description();
2: if (desc == "")
3: desc = "You see a web room without a description (look at the nice picture).";
4: endif
5: "this: announce_all($string_utils: print(desc))";
6: if (typeof(desc) == LIST)
7: desc = {"<b> ", @desc, "</b> "};
8: else
9: desc = {"<pre> ", desc, "</pre> "};
10: endif
11: return desc;

#154:"html_tell_exits" this none this
1: obexits_list = {};
2: for exit in (this.exits)
3: if (exit.obvious)
4: obexits_list = listappend(obexits_list, $web_utils: wrap_exit(exit) + " to " + exit.dest.name);
5: endif
6:   endfor
7:   if (obexits_list != { })
8:       $web_utils:httpwrite(tonum(player), "Obvious exits: " +
9:       $string_utils:english_list(obexits_list));
10:  endif
11:  if (0)
12:     "for debugging"
13:   this: announce_all("obvious exits: ");
14:  this: announce_all($string_utils:english_list(obexits_list));
15:  endif

#154:"html_tell_contents"  this none this
1: contents = args[1];
2: ctype = args[2];
3: if (!this.dark && contents != { })
4:     if (ctype == 0)
5:         player:tell("Contents:");
6:         for thing in (contents)
7:             $web_utils:httpwrite(tonum(player), " " +
$web_utils:wrap_look(player, thing:title()));
8:     endfor
9:     elseif (ctype == 1)
10:        for thing in (contents)
11:           if (is_player(thing))
12:               $web_utils:httpwrite(tonum(player),
$web_utils:wrap_look(player,
$string_Utils:pronoun_sub("%n %<is> here.", thing)));
13:           else
14:               $web_utils:httpwrite("You see ",
$web_utils:wrap_look(player,
thing:title()), " here.");
15:           endif
16:         endfor
17:     elseif (ctype == 2)
18:        $web_utils:httpwrite("You see ", $web_utils:wrap_look(player,
$string_utils:title_list(contents)), " here.");
19:     elseif (ctype == 3)
20:        players = things = { }
21: for x in (contents)
22:     if (is_player(x))
23:        players = {@players, x};
else
  things = { @things, x };
endif
endfor
if (things)
titles = $list_utils:map_verb(things, "title");
if (0)
  this:announce("titles are " + $string_utils:print(titles));
endif
htmltitles = $list_utils:map_arg(2, $web_utils, "wrap_look", player, titles);
if (0)
  this:announce("htmltitles are " + $string_utils:print(htmltitles));
endif
$web_utils:httpwrite(tonum(player), "You see " + $string_utils:english_list(htmltitles) + " here.<p>");
eendif
if (players)
titles = $list_utils:map_verb(players, "title");
i'm not sure why we need the next line. I mooched it from
$string_utils:title_listc";
titles[1] = players[1]:titlec();
htmltitles = $list_utils:map_arg(2, $web_utils, "wrap_look", player, titles);
if (0)
  this:announce("htmltitles are " + $string_utils:print(htmltitles));
endif
$web_utils:httpwrite(tonum(player), $string_utils:english_list(htmltitles) + (length(players) == 1 ? $string_utils:pronoun_sub(" %<is>" , players[1]) | " are") + " here.");
eendif
eendif

#154: "@getimage" none (on top of/on/onto/upon) this
1: player:tell("The image corresponding to this object is..."),
2: player:tell(this.image_url);
#154: "@setimage" any (on top of/on/onto/upon) this
1:   this.image_url = dobjstr;
2:   player.tell("You set the image_url to " + dobjstr + ".");

#154: "look_back" this none this
1:   if ($web_utils:is_webber(player))
2:     "we can be sure player is a webber.";
3:   if (player:is_webbing())
4:     "rebuild complete room display"
5:     saved = player.recording;
6:     player.recording = 1;
7:     player.recording = saved;
8:     savedtell = player:divert_tell_html();
9:     if (!savedtell)
10:    "was not already diverting, so lets do a diversion";
11:    $web_utils:httpbegin(tonum(player));
12:    "player:divert_tell_html(1);"
13:    if (this.ismap)
14:       $web_utils:httpwrite(tonum(player), "<a href="http://ninja-turtle.mit.edu:8009/cgi-bin/moo_map/" + this.mapname + "+" + tostr(tonum(player)) + "\">" + this:image_url(1) + "</a>");
15:      else
16:         $web_utils:httpwrite(tonum(player), this:image_url());
17:    endif
18:    $web_utils:httpwrite(tonum(player), "<p>");
19:    $web_utils:httpwrite(tonum(player), "<h1>" + this:title() + "</h1>");
20:    if (!!(args && args[1]))
21:       $web_utils:httpwrite(tonum(player), this:description_html());
22:    endif
23:    if (player.show_exits)
24:      this:html_tell_exits();
25:    endif
26: "player:divert_tell_html(1);this:tell_contents(setremove(this:contents(),player), this.ctype);player:divert_tell_html(0);";
27:    $web_utils:httpwrite(tonum(player), "<p>");
28:    $web_utils:httpwrite(tonum(player), this:image_url());
$web_utils:append_quit_button(tonum(player));
$web_utils:ccisend(tonum(player));
endif
else
    pass(@args);
endif
else
    this:look_self(@args);
endif
The $web_utils object

Object ID: #162
Name: Web Utilities
Parent: Generic Utilities Package (#79)
Location: Adam (#2)
Owner: Adam (#2)
Flags:
Verb definitions:
  init_for_core
  httpbegin
  ccisend
  gifwrite
  httpwrite
  is_webber
  all_webbers_in
  is_web_readable
  is_webthing
  init_webber
  wrap_exit
  cgi_ref
  wrap_look
  append_look_back
  wrap_href
  append_quit_button
  change_pass_code
Property definitions:
  use_builtin
  cgi_ref
  goodbye_html
Properties:
  key: 0
  aliases: {"Web Utilities", "web_utils"}
    description: "This is a placeholder parent for all the $..._utils packages, to more easily find them and manipulate them. At present this object defines no useful verbs or properties. (Filfre.)"
    object_size: {0, 0}
    help_msg: {"This is the Generic Utility Object. One presumes it should have text in it explaining the use of the utility object in question."}
    use_builtin: 0
    cgi_ref: "http://ninja-turtle.mit.edu:8009/cgi-bin/cgi_with_args"
goodbye_html: {"Take it easy. Hope you had fun!<p> <a href="http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/webpages/moo_home.html">Return to the NinjaMOO home page</a></html>"}

#162: "init_for_core" this none this
1: pass(@args);

#162: "httpbegin" this none this
1: if (this.use_builtin)
2: "httpbegin(@args);";
3: else
4: player.html_doc = {"<html>"};
5: endif

#162: "ccisend" this none this
1: if (this.use_builtin)
2: "ccisend(@args);";
3: else
4: $web_utils:httpwrite(tonum(player), "</html>");
5: player:sendhtml();
6: endif

#162: "gifwrite" this none this
1: "gifwrite(@args);";

#162: "httpwrite" this none this
1: if (this.use_builtin)
2: "httpwrite(@args);";
3: else
4: player:httpwrite_self(args[2]);
5: if (0)
6: player.location:announce(tostr(player) + " httpwrite: " + $string_utils:print(args[2]));
7: endif
8: endif
#162:"is_webber"  this none this
1:  if (length(args) > 0)
2:    OBJ = toobj(args[1]);
3:    return "is_webbing" in $object_utils:all_verbs(OBJ) > 0;
4:  else
5:    return 0;
6:  endif

#162:"all_webbers_in"  this none this
(That verb has not been programmed.)

#162:"is_web_readable"  this none this
1:  testobj = toobj(args[1]);
2:  return "image_url" in $object_utils:all_verbs(testobj);
3:  " || $web_utils:is_webthing(testobj)";

#162:"is_webthing"  this none this
(That verb has not been programmed.)

#162:"init_webber"  this none this
1:  if (0)
2:    this.location:announce("in init_webber, player is", args[1]);
3:  endif
4:  if ($web_utils:is_webber(args[1]))
5:    args[1]:init_web_myself();
6:  endif

#162:"wrap_exit"  this none this
1:  exit = args[1];
2:  if (0)
3:    player:tell("exit is " + exit.name);
4:  endif
5:  res = "<A HREF=" + $web_utils:cgi_ref() + "?" +
tostr(tonum(player)) + "+" + tostr(exit.name) + "">" + tostr(exit.name) +"</A>";
6:  return res;
7:  "%23";
#162: "cgi_ref" this none this
1:    return this.cgi_ref;

#162: "wrap_look" this none this
1:    if (0)
2:        player.location:announce("args in wrap_look " + $string_utils:print(args));
3:    endif
4:    thing = args[2];
5:    res = "<A HREF=" + $web_utils:cgi_ref() + "?" + tostr(tonum(player)) + "+" + "look+at+" + tostr(thing) + "\">" + tostr(thing) + "</A>";
6:    return res;

#162: "append_look_back" this none this
1:    $web_utils:httpwrite(tonum(player), "Turn your eyes <A HREF="" + $web_utils:cgi_ref() + "?" + tostr(tonum(player)) + "+look\">back</A> to the room.");

#162: "wrap_href" this none this
1:    link = args[1];
2:    link_name = args[2];
3:    stuff_to_send = "<a href = "" + link + "">" + link_name + "</a>";
4:    return stuff_to_send;

#162: "append_quit_button" this none this
1:    player = args[1];
2:    button_loc = "<IMG SRC = "/"http://web.mit.edu/afs/athena.mit.edu/course/1/1.125/moo/images/button_QUIT.gif\">";
3:    $web_utils:httpwrite(tonum(player), "<hr><A HREF="" + $web_utils:cgi_ref() + "?" + tostr(tonum(player)) + "@quit\">" + button_loc + "</A>");

#162: "change_pass_code" this (on top of/on/onto/upon) any
1:    caller.location:announce(random(32323));
The $webber object

Object ID: #149
Name: Generic Web Person
Parent: generic player (#6)
Location: Adam (#2)
Owner: Adam (#2)
Flags:
Verb definitions:
  init_for_core
tell
  record_event
tell_event
tell_lines
  erase_recording
get_events
@listevents @le
@eraseevents @ee
notify
@record
ignore_event
@ignore
notice_event
@notice
  description_html
@web
is_webbing
i inv*entory
look_self
imageURL
  httpwrite_self
sendhtml
  divert_tell_html
image_url
@setccimachine
init_web_myself
setccimachine
send_events
@setimage
@getimage
@listen
  goodbye_screen
Property definitions:
event_types
record_tell
recorded_events
recording
is_webbing
html_doc
cci_machine
is_web_command
divert_tell_html
event_types_used
image_url
auto_set_machine
mapname
pass_code
Properties:
key: 0
aliases: {"gwp"}
description: "A Generic web character."
object_size: {0, 0}
features: {}
  previous_connection: 0
mail_lists: {}
  email_address: ""
last_disconnect_time: 0
help: #181
more_msg: "*** More *** %n lines left. Do @more [restiflush] for more."
linetask: {0, 0}
linesleft: 0
linebuffer: {}
pagelen: 0
_mail_task: 0
owned_objects: {}
linelen: -79
current_folder: 1
all_connect_places: {}
last_connect_place: "?"
dict: {}
messages_going: {}
brief: 0
responsible: {}
lines: 0
page_absent_msg: "%N is not currently logged in."
#149:"init for core"  this none this
1:    pass();

#149:"tell"  this none this
1:    if ("is_webbing" in $object_utils:all_verbs(player))
2:        if (this.gaglist || this.paranoid)
3:            "Check the above first, default case, to save ticks. Paranoid gaggers are";
4:        "cost an extra three or so ticks by this, probably a net savings.";
5:        if (this:gag_p())
6:            return;
7:        endif
8:        if (this.paranoid == 1)
9:            $paranoid_db:add_data(this, {{@callers(), {player, "<cmd-line>", player}}}, args));
10:        elseif (this.paranoid == 2)
11:            z = this:whodunnit({@callers(), {player, "", player}},{this, $no_one}, { })[3];
12:            args = {"(" , z.name, " ", z, ") ", @args};
13:        endif
14:    endif
15:    "will have returned already if gagged"
16:    saved = this.recording;
17:    divert = this:divert_tell_html();
18:    if (!saved && !divert)
19:        this:record_event("tell_event", args);
20:    endif
21:    "dont let notify record the event, we already did";
if (divert)
  $web_utils:httpwrite(tonum(player), tostr(@args));
else
  this.recording = 1;
  this:notify(tostr(@args));
  this.recording = saved;
endif
else
  pass(@args);
endif

#149:"record_event"  this none this
1:  eventtype = args[1];
2:  prev_args = args[2];
3:  this.recording = 1;
4:  if (!(eventtype in this.event_types_used))
5:     this.recording = 0;
6:     return;
7:  else
8:     "  this:(eventtype)(@prev_args);
9:     this.recorded_events = { @this.recorded_events, {eventtype,
prev_args} }
10:    this.recording = 0;
11:   endif

#149:"tell_event"  this none this
1:  "This logs the tell_event";

#149:"tell_lines"  this none this
1:  if ($web_utils:is_webber(player))
2:    "enter this if player is a webber";
3:    lines = args[1];
4:  if (typeof(lines) != LIST)
5:    lines = {lines};
6:  endif
7:  if (this.gaglist || this.paranoid)
8: "Check the above first, default case, to save ticks. Paranoid
gaggers are cost an extra three or so ticks by this, probably a net
savings."
9: if (this:gag_p())
10: return;
11: endif
12: if (this.paranoid == 2)
13: z = this:whodunnit({@callers(), {player, "", player}}, {this, $no_one}, {3});
14: lines = {["start text by " + z.name + " (" + tostr(z) + ")"], @lines,
15: ["end text by " + z.name + " (" + tostr(z) + ")"],
16: $paranoid_db:add_data(this, {{@callers}, {player, "<cmd-
line>"}, lines});
17: endif
18: endif
19: "don't gather stats for now:
20: $list_utils:check_nonstring_tell_lines(lines)"
21: "will have returned already if gagged, so lets record the event"
22: saved = this.recording;
23: divert = this:divert_tell_html();
24: if (!saved && !divert)
25: this:record_event("tell_lines_event", args);
26: endif
27: if (divert)
28: $web_utils:httpwrite(tonum(player), lines);
29: else
30: this.recording = 1;
31: this:notify_lines(lines);
32: this.recording = saved;
33: endif
34: else
35: pass(@args);
36: endif

#149:"erase_recording" this none this
1: this.recorded_events = {};
#149: "get_events"  this none this
1:  "this verb will return a list of all events of the type";
2:  "given in the string or list of strings in args[1]";
3:  if (argtype = typeof(args[1]) == STR)
4:    targets = {args[1]};
5:  elseif (argtype == LIST)
6:    targets = args[1];
7:  else
8:    "strange argument passed";
9:  return E_TYPE;
10: endif
11: "must also check if they are valid events...does it matter??";
12: "why waste time, we already checked when we logged the
events";
13: captured = {};
14: len = length(this.recorded_events);
15: this.location:announce_all("length is ", len);
16: for i in [len..1]
17:    this.location:announce_all("searching element ", i);
18:    event = this.recorded_events[i];
19:    if (event[1] in targets)
20:      captured = {event, @captured};
21:      listdelete(this.recorded_events[i]);
22:    endif
23: endfor
24: return captured;

#149: "@listevents @le"  any any any
1:  "List events on an object.";
2:  "Usage: @listevents [quietly] [on web-person]";
3:  "Default is '@listevents on me'";
4:  if (prepstr == ")")
5:    target = this;
6:  elseif (prepstr == "on")
7:    "do we need this toobj()?";
8:    if ((target = toobj(this: my_matchobject(iobj str, caller.location)))
9:      return;
10: elseif (typeof(property_info(target, "recorded_events")) == ERR)
11:      $code_utils:showproperty(#57, "recorded_events");
13:    player:tell(tostr(target));
14:    player:tell("That is not a generic web person");
15:    return;
16:    endif
17:    endif
18:    if ("recording" in $object_utils:all_properties(player))
19:        saved = player.recording;
20:        player.recording = index(dobjstr, "quiet") > 0;
21:    endif
22:    eventlist = target.recorded_events;
23:    player:tell("Events in ", target.name);
24:    for anevent in (eventlist)
25:        player:tell($string_utils:print(anevent));
26:    endfor
27:    if ("recording" in $object_utils:all_properties(player))
28:        player.recording = saved;
29:    endif

#149:"@eraseevents @ee" any any any
  1:    if (prepstr == "")
  2:        target = this;
  3:    elseif (prepstr == "on")
  4:        "do we need this toobj()?";
  5:        if ((target = toobj(this:my_match_object(iobjstr, caller.location)))
 ==E_TYPE || target == $failed_match)
  6:            player:tell("Couldnt find ", iobjstr, " in here.");
  7:        return;
  8:    elseif (!"recorded_events" in $object_utils:all_properties(target))
  9:        "elseif (typeof(property_info(target,
10:            "recorded_events";
11:        ")==ERR)"
12:        $code_utils:show_property(target, "recorded_events");
13:        player:tell(tostr(target));
14:        player:tell("That is not a generic web person");
15:        return;
16:    endif
17:    endif
18:    if ("recording" in $object_utils:all_properties(player))
19:        player.recording = index(dobjstr, "quiet") > 0;
20:    endif
21:    player:tell("Erasing events on ", target.name, ", ", target, 
22:    target.recording ? "", quietly." | "."));
23:    this.recorded_events = {};
24:    player:tell("Erased all events");
25:    if ("recording" in $object_utils:all_properties(player))
26:       player.recording = 0;
27: endif

#149:"notify" none none none
1:    pass(@args);

#149:"@record" this any none
1:    "Start or stop event recording. ";
2:    "@record me on";
3:    "@record me off";
4:    "See also @listen.";
5:    target = caller.location:match_object(dobj str);
6:    caller:tell($string_utils:print(args));
7:    if (args[2] == "off")
8:       target.recording = 1;
9:    caller:tell("Event recording on ", target.name, " is OFF.");
10:   target:tell("Event recording on you is OFF.");
11: elseif (args[2] == "on")
12:   target.recording = 0;
13:   caller:tell("Event recording on ", target.name, " is ON.");
14:   target:tell("Event recording on you is ON.");
15: else
16:   caller:tell("Please specify ON or OFF.");
17: endif

#149:"ignore_event" this none this
1:    "Remove an event_type from the event_types list";
2:    "Usage: ignore_event(eventstr[,SAVEIT])";
3:    "Removes eventstr from event_list and if SAVEIT then place";
4:    "it into the event_types_unused list";
5:    eventstr = args[1];
6:    savit = length(args) == 2 ? tonum(args[2]) I 0;
7:   eventidx = eventstr in this.event_types;
8:   if (eventidx)
9:      this.event_types = listdelete(this.event_types, eventidx);
10:     if (saveit)
11:        this.event_types_unused =
12:           listappend(this.event_types_unused,
13:              eventstr);
14:     endif
15:   endif

#149:"@ignore" any (on top of/on/onto/upon) any
1:   "set_task_perms(caller);"
2:   player:tell($string_utils:print(args));
3:   if ((len = length(args)) == 3)
4:      eventstr = dobjstr;
5:   else (target = this.location:match_object(iobjstr)) ==
6:      $failed_match)
7:      player:tell("Could not find ", iobjstr, ".");
8:      return;
9:   endif
10:  elseif (len == 1)
11:     eventstr = args[1];
12:     target = this;
13:  else
14:     player:tell("Wrong number of arguments to @ignore");
15:     player:tell("Usage: @ignore <eventstr> on <webplayer>");
16:     return;
17:  endif
18:  if ($web_utils:is_webber(target))
19:     target:ignore_event(eventstr, 1);
20:  else
21:     player:tell("Sorry, but ", target.name, " is not a web player.");
22:  endif
#149:"notice_event"  this none this
  1:  "Add an event_type to the event_types list";
  2:  "Usage: watch_event(eventstr[,SAVEIT])";
  3:  "Places eventstr into event_list and removes it from";
  4:  "from the event_types_unused list if it is there";
  5:  set_task_perms(caller);
  6:  player:tell($string_utils:print(args));
  7:  eventstr = args[1];
  8:  eventidx = eventstr in this.event_types_unused;
  9:  if (eventidx)
 10:     this.event_types_unused = listdelete(this.event_types_unused, eventidx);
 11: endif
 12: this.event_types = listappend(this.event_types, eventstr);

#149:"@notice"  any (on top of/on/onto/upon) any
  1:  player:tell($string_utils:print(args));
  2:  set_task_perms(caller);
  3:  if ((len = length(args)) == 3)
  4:    eventstr = dobjstr;
  5:    if ((target = this.location:match_object(iobjstr)) == $failed_match)
  6:      player:tell("Could not find ", iobjstr, ".");
  7:      return;
  8:    endif
  9:  elseif (len == 1)
 10:    eventstr = args[1];
 11:    target = this;
 12:  else
 13:    player:tell("Wrong number of arguments to @watch");
 14:    player:tell("Usage: @watch <eventstr> on <webplayer>");
 15:    return;
 16:  endif
 17:  if ("notice_event" in $object_utils:all_verbs(target))
 18:    player:tell(target, " ", eventstr);
 19:    target:notice_event(eventstr);
 20:  else
 21:    player:tell("Sorry, but ", target.name, " is not a web player.");
 22:  endif
```plaintext
#149: "description_html" this none this
1:   lines = this: description();
2:   if (lines)
3:       if (typeof(lines) == LIST)
4:           lines = "{<pre>, @lines, "</pre>}";
5:       else
6:           lines = "<pre>" + tostr(lines) + "</pre>";
7:       endif
8:   else
9:       lines = "<pre>" + "You see nothing special." + "</pre>";
10:  endif
11:  return lines;

#149: "@web" this any none
1:   if (prepstr == "on")
2:       if (this:is_webbing())
3:           this: tell("Webbing already on.");
4:           return;
5:       else
6:           this: tell("Webbing now on.");
7:           this:is_webbing = 1;
8:           return;
9:       endif
10:  elseif (prepstr == "off")
11:    if (this:is_webbing())
12:      this: tell("Webbing now off.");
13:    this:is_webbing = 0;
14:      return;
15:    else
16:      this: tell("Webbing already off.");
17:      return;
18:    endif
19:  endif

#149: "is_webbing" this none this
1:   return this.is_webbing;
```
#149:"i inv*entory" this none this
1:  $web_utils:httpbegin(tonum(player));
2:  pass(@args);
3:  $web_utils:ccisend(tonum(player), player.cci_machine);

#149:"look_self" this none this
1:  if ($web_utils:is_webber(player))
2:    "we can be sure player is a webber."
3:    if (player:is_webbing())
4:      "Dont let it record any tell events"
5:        saved = player.recording;
6:        player.recording = 1;
7:        "do normal player:look_self"
8:        pass(@args);
9:        player.recording = saved;
10:    "Now do the web version:"
11:    $web_utils:httpbegin(tonum(player));
12:    $web_utils:httpwrite(tonum(player), this:image_url());
13:    savedtell = player:divert_tell_html(1);
14:    pass(@args);
15:    player:divert_tell_html(savedtell);
16:    this:announce(player.name + " looking at ", this.name);
17:    $web_utils:append_look_back();
18:    $web_utils:ccisend(tonum(player));
19:    else
20:      pass(@args);
21:    endif
22:  else
23:    pass(@args);
24:  endif

#149:"imageURL" this none this
1:  $web_utils:httpwrite(tonum(player), tostr("<IMG HEIGHT=200 WIDTH=200 SRC="Obj", tonum(this), ".gif">"));
#149:"httpwrite_self" this none this
1:   towrite = typeof(args[1]) == LIST ? args[1] | {args[1]};
2:   player.htmldoc = {@player.htmldoc, @towrite};

#149:"sendhtml" this none this
1:   if (this.htmldoc)
2:     this.htmldoc = {"html", this.cci_machine, tonum(this),
@this.htmldoc};
3:     $ccilink:send_packet(this.htmldoc);
4:   endif

#149:"divert_tell_html" this none this
1:   saved = this.divert_tell_html;
2:   if (length(args) > 0)
3:     new = this.divert_tell_html = tonum(args[1]);
4:     if (new && !saved)
5:       "switching from 0 to 1";
6:     $web_utils:httpwrite(tonum(this), "<pre>");
7:   elseif (!new && saved)
8:     "switching from 1 to 0";
9:     $web_utils:httpwrite(tonum(this), "</pre>");
10: endif
11: endif
12: return saved;

#149:"image_url" this none this
1:   return "<IMG SRC="" + this.image_url + "">;"

#149:"@setccimachine" none none none
1:   this:setccimachine();
#149: "init_web_myself" this none this
1: if (this.auto_set_machine)
2:   this: setccimachine();
3: endif
4: this: divert_tell_html(0);
5: this.is_webbing = 1;
6: this.is_web_command = 0;
7: this.htmldoc = {};

#149: "setccimachine" this none this
1: this.cci_machine = $string_utils:words(this.last_connect_place)[3];

#149: "send_events" this none this
1: "go through the vent list from front to back and construct a list of lines";
2: "to send to ccilink-bigisis";
3: lines = {};
4: for anevent in (this.recorded_events)
5:   estring = anevent[2];
6:   if (anevent[1] == "tell_event")
7:     lines = { &lines, $string_utils:from_list(estring) };
8:   elseif (anevent[2] == "tell_lines_event")
9:     this.location: announce("In send_events, estring = " +
$string_utils: print(estring));
10:    lines = { &lines, @$list_utils: map_arg($string_utils, "from_list",
estring) };
11:   endif
12: endfor
13: if (lines)
14:   lines = { "tell_stuff", this.cci_machine, tonum(this), @lines };
15:   $ccilink: send_packet(lines);
16: this.recorded_events = {};
17: endif
#149: "@setimage" any (on top of/on/onto/upon) this
   1:  this.image_url = dobjstr;
   2:  player:tell("You set the image_url to " + dobjstr + ".");

#149: "@getimage" none (on top of/on/onto/upon) this
   1:  player:tell("The image corresponding to this object is...");
   2:  player:tell(this.image_url);

#149: "@listen" none none none
   1:  this:send_events();

#149: "goodbye_screen" this none this
   1:  this.htmldoc = $web_utils.goodbye_html;
   2:  this:sendhtml();

202
The $ccilink object

Object ID: #159
Name: ccilink
Parent: link (#158)
Location: Acme (#157)
Owner: Adam (#2)
Flags:
Verb definitions:
   handle_input
   init_for_core
   testhtml
Properties:
   key: 0
   aliases: {"ccilink", "cci"}
   description: "This is a link object for connecting to external processes."
   object_size: {0, 0}
   drop_failed_msg: "You can't seem to drop %t here."
   drop_succeeded_msg: "You drop %t."
   odrop_failed_msg: "tries to drop %t but fails!"
   odrop_succeeded_msg: "drops %t."
   otake_succeeded_msg: "picks up %t."
   otake_failed_msg: ""
   take_succeeded_msg: "You take %t."
   take_failed_msg: "You can't pick that up."
   current_packet: 0
   conn: #-3
   port: 7776
   address: "ninja-turtle"
   packet_buf: {}
   linked: 1

#159:"handle_input"     this none this
   1: pass(args);

#159:"init_for_core"    this none this
   1: pass(@args);

#159:"testhtml"        this none none
   1: this.packet_buf = {"html", "m16-034-19.mit.edu", "123",
                         "<html>","<pre>", "hello", "</pre>", "</html>", " "};
   2: this:send_packet();
Makefile for the bigsis program

CC = gcc -g -Wall -Wwrite-strings
LIBRARIES = -lm -lnsl -L./libcci/ -lcci
CCIOBJJS = ccilink.o ioutils.o cci_utils.o
CGIOBJJS = cgilink.o ioutils.o
COMPILE.c = $(CC) $(CFLAGS) $(CPPFLAGS) -c
HDRS = my_defs.h ioutils.h

bigsis: $(CCIJOBJS)
    $(CC) $(CFLAGS) $(CCIJOBJS) $(LIBRARIES) -o @$
cgilink: $(CGIJOBJS)
    $(CC) $(CFLAGS) $(CGIJOBJS) $(LIBRARIES) -o @$
.SUFFIXES: .po

c.po: $(HDRS)
    $(COMPILE.c) -o @$ <$

204
ccilink.c

/*************************** ccilink.c*****************************/

* Redistribution and use in source and binary forms are permitted
* provided that this paragraph is included. Any distribution of
* this software or derivative works must comply with all applicable
* United States export control laws.
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY
* EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT
* LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY
* AND FITNESS FOR A PARTICULAR PURPOSE.
*
* Any person obtaining a copy of this software is requested
* to send their name and electronic mail address to:

    ssgaudet@mit.edu     or     askwersk@mit.edu

*

*************************************************************************/

/* BIG SISTER CODE           */
/* CALL IT LIKE THIS:         */
/* bigsis 1 7776 7775         */
/* the 1 means try to connect to* /
/* mosaics. If you use 0, then  */
/* cci will not try to make any */
/* mosaic network connections   */
/* the second parameter is the */
/* port number that big sister */
/* connects to moo server with */
/* the third number is the port */
/* that cci sends to mosaic on. */

#include "my_defs.h"
#include "cci_utils.h"
extern printf();
extern fprintf();
extern fclose();
extern bzero();
extern rename();
void cci_err(const char *);
void makeSocks(int argc, char *argv[]);
void doWork(void);
void readpacket(void);
void performCCI(void);
void filterCR(char *line);
void filterN(char *line);

/*declared in cci_utils.c*/
extern char ss_host_name[];
extern long int ss_port_num;
extern MCCIPort ss_cci_port;
int mooSock=-1,servSock;
int do_urls=0;

char objnum[16];
void readln(char *buf);
enum {
    ARG_DO_URL=1,
    ARG_HOSTPORT,
    ARG_CCI.PORT,
};

void main(int argc, char *argv[])
{
    printf("argc = %d\n",argc);
    if (argc<2)
        cci_err("Invalid number of arguments! Need at least 1.\n");
    printf("args are \n%s\n%s\n",argv[0],argv[1]);
    do_urls = atoi(argv[ARG_DO_URL]);
    printf(do_urls ? "doing urls\n" : "not doing urls\n");
    ss_port_num=(argc>3) ? atoi(argv[ARG_CCI.PORT]) : 7775;
    printf("CCI Port is %ld.\n",ss_port_num);
    makeSocks(argc,argv);
    doWork();
}

void makeSocks(int argc,char *argv[]){
    short hostport = (argc>2) ? atoi(argv[ARG_HOSTPORT]):7776;
    struct sockaddr_in serv_addr,moo_addr;
    int moo_len;
    char *name=NULL;
printf("Here's what I got: hostport = %d\n", hostport);
if ((servSock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    cci_err("Could not create a socket!\n");
/*
 * bind local address so clients can send to us.
 */
bzero((char*) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(hostport);
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);

if (bind(servSock, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
    cci_err("Cannot bind local address!");
listen(servSock, 5);
printf("CCIlink now listening\n");

moo_len = sizeof(moo_addr);
if ((mooSock = accept(servSock, (struct sockaddr *)
    &moo_addr, &moo_len)) < 0)
    cci_err("Cannot accept a connection!\n");
printf("Got a connection!\n");
printf("hostname and port are %ld and %d\n",
    ntohl(moo_addr.sin_addr.s_addr),
    ntohs(moo_addr.sin_port));
name = inet_ntoa(moo_addr.sin_addr);
printf("hostname and port are %s and %d\n", name,
    moo_addr.sin_port);
}

void filterCR(char *line) {
    int count = 0;
    /*^M \r usually occurs just before a linefeed, \n so
    * just replace them with a space.*/
    while (line[count]) {
        if (line[count] == '\r')
            line[count] = '\ ';
        count++;
    }
}
#define BUF_SIZE 4096
char buf[BUF_SIZE];
char packtype[32];
char clientaddr[128];

#define PACK_TEMP_FILE "pack.temp"
void readpacket(void){
    char aline[512];
    int i;
    int amtread,loc=0;
    int done = 0;
    FILE *pack=NULL;
    buf[0]=\0';
    if (readline(mooSock,packtype,80)<0)
        cci_err("Error trying to read in readpacket\n");
    printf("got packtype: %s",packtype);
    if (readline(mooSock,clientaddr, 80)<0)
        cci_err("Error trying to read in clientaddr\n");
    printf("Got clientaddress");
    filterCR(clientaddr);
    i=strlen(clientaddr);
    i--;
    while (((clientaddr[i]==')') || (clientaddr[i]==')') || (clientaddr[i]==')'))
        if (((clientaddr[i]==')') || (clientaddr[i]==')') || (clientaddr[i]==')'))
            printf("ni=%d c = %c",i,clientaddr[i]);
            clientaddr[i]=0;
            i--;
        }
    strcpy(ss_host_name,clientaddr);
    printf("ss_host_name = :%s\n",clientaddr);
    if (readline(mooSock,objnum,16)<0)
        cci_err("Error getting object number\n");
    filterCR(objnum);
    {
        char *temp=NULL;
        if (temp=strchr(objnum,' '))
            *temp=0;
        printf("object number is %s\n",objnum);
    }
    if(!(pack = fopen(PACK_TEMP_FILE,"w"))
        cci_err("Couldnt open file in readpacket\n");
}
while (!done)
    if ((amtread = readline(mooSock, aline, 512)) <= 0) {
        if (pack) fclose(pack);
        cci_err("Error trying to read in readpacket\n");
    } else if ((amtread == 2) && (aline[0] == '\n')) {
        printf("now done\n");
        done = 1;
    } else {
        aline[amtread] = '\0';
        printf("received : %s", aline);
        /* need to filter out ^M's, I think they are \r's */
        filterCR(aline);
        fprintf(pack, aline);
    }
    printf("read %d bytes\n", amtread);
}
fclose(pack);

#define HTML_FILE "moo"
void performCCI(void)
{
    char filename[64] = "moo";
    char filename2[192] = "";
    printf("entering performCCI\n");
    strcat(filename, objnum); /* should probably be the object number */
    if (!strncmp(packtype, "html", 4)) {
        strcat(filename, ".html");
        printf("renameing to %s\n", filename);
        printf("result of rename: %d\n", rename(PACK_TEMP_FILE, filename));
        printf("file renamed\n");
        strcat(filename2, HTML_BASE);
        strcat(filename2, filename);
        printf("Showing URL %s\n", filename2);
        if (do_urls)
            ss_show_next_url(filename2);
        MCCIDisconnect(ss_cci_port);
    }
}
void doWork(void) {
    while(1) {
        readpacket();
        performCCI();
    }
}

void readln(char *buf) {
    int loc=0;
    if (((buf[loc++]=getchar())=='\n'))
        loc=0;
    while ((buf[loc++]=getchar())!='\n');
    buf[--loc]='\0';
}

void cci_err(const char *string) {
    printf("\%s",string);
    exit(0);
}

#include <stdio.h>    /* C includes */
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "cci_utils.h"    /* CCI includes */

#define VERSION "1.0"
#define STR_LEN 255

MCCIPort ss_cci_port;
char ss_script_file[STR_LEN], ss_host_name[STR_LEN];
char *cmd_name;
long int ss_port_num=0;
int ss_no_sleep=0, ss_last_sleep=0, quiet_flag=0;

void ss_default_init()
{
    strcpy(ss_host_name, "ml-142-12.mit.edu");
    ss_port_num = 7776;
}

void ss_display_usage()
{
    printf("\nCCI Slide Show %s\n", VERSION);
    printf("Usage: %s options\n where options is one or more
of:\n", cmd_name);
    printf("-file followed by the name of a text file that contains
one\n");
    printf(" instruction per line for the CCI-compliant
browser\n");
    printf("-host followed by the host name of the machine running
the browser\n");
    printf("-port followed by the port number on which the browser is
listening\n");
    printf(" for CCI messages\n");
    printf("-url followed by a single URL to be fetched and
displayed\n");
    printf("-quiet optional flag to disable run-time console
messages\n");
    /*exit(0);*/
}
int ss_make_cci_connection() /* return 0 on success, 1 or 2 on
failure */
{
    if (strcmp(ss_host_name, "default\%")==0)
    {
        printf("\n%s: Error, use -host to specify a host machine.\n", cmd_name);
        ss_display_usage();
    }
    if (ss_port_num==0)
    {
        printf("\n%s: Error, use -port to specify a port number.\n", cmd_name);
        ss_display_usage();
    }
    if (quietflag==0)
        printf("CCI Slide Show: Connecting to %s at port %d\n", ss_host_name, ss_port_num);
    if (!ss_cci_port)
    {
        printf("\n%s: Error, could not connect to machine %s at port
%d.\n", cmd_name, ss_host_name, ss_port_num);
        ss_display_usage();
    }
}

ss_sleep(int sleep_time)
{
    if (quiet_flag==0)
        printf("CCI Slide Show: Sleeping for %d seconds.\n", sleep_time);
    if (sleep_time<0)
        sleep_time=0;
    sleep(sleep_time);
    ss_last_sleep=sleep_time;
}
int ss_show_next_url(char *url) /* return 0 on success, or 1 on
failure */
{
    if (ss_no_sleep==1)
        ss_sleep(ss_last_sleep);
    /* we might need to store a table of ss_cci_port's if we cannot */
    /* send to other mosaics. */
    if (!MCCIIsConnected(ss_cci_port))
        ss_make_cci_connection();
    if (quiet_flag==0)
        printf("CCI Slide Show: Getting URL %s\n", url);
    if (MCCIIsConnected(ss_cci_port))
    {
        MCCIGet(ss_cci_port, url, MCCI_DEFAULT, MCCI_ABSOLUTE, 0);
        return 0;
    }
    else
        return 1;
}

void ss_process_cmd(char s[STR_LEN])
{
    char *cmd, *var;
    int free_flag=0;

    if ( (s[0]=='#') || (s[0]=='!') || (s[0]=='\n') ) /* # denotes a comment */
        return;
    cmd=strtok(s, " ");
    var=strtok(NULL, "");

    if (cmd==NULL)
        return;
    if (strlen(cmd)==0)
        return;
    if (var==NULL) /* special case, only a URL on this line */
    {
        var=strdup(cmd);
        cmd=strdup("url");
        free_flag=1;
    }
    else
        free_flag=0;
}
if (var[strlen(var)-1]=='\n')
    var[strlen(var)-1]=NULL;

if (strcmp(cmd, "url") == 0) /* url */
{
    if (ss_show_next_url(var)==1)
        printf("CCI Slide Show: Error, no connection.\n");
    ss_no_sleep=1;
}

if (strcmp(cmd, "sleep") == 0) /* sleep */
{
    ss_sleep(atoi(var));
    ss_no_sleep=0;
}

if (strcmp(cmd, "host") == 0) /* host */
{
    if (MCCIIsConnected(ss_cci_port))
        MCCIDisconnect(ss_cci_port);
    strcpy(ss_host_name, var);
}

if (strcmp(cmd, "port") == 0) /* port */
{
    if (MCCIIsConnected(ss_cci_port))
        MCCIDisconnect(ss_cci_port);
    ss_port_num = atoi(var);
}

if (strcmp(cmd, "quiet") == 0)
    if (strcmp(var, "on") == 0)
        quiet_flag=1;
    else
        quiet_flag=0;

if (strcmp(cmd, "exec") == 0)
    system(var);
if (free_flag==1)
{
    free(cmd);
    free(var);
}
return;

int ss_slide_show() /* return 0 if file opens, 1 on failure */
{
    FILE *fp;
    char s[STR_LEN];
    if ( (fp=fopen(ss_script_file, "r")) == NULL)
        return 1;
    else
    {
        while (fgets(s, STR_LEN, fp)!=NULL)
            ss_process_cmd(s);
        fclose(fp);
        MCCIDisconnect(ss_cci_port);
        return 0;
    }
}

int ss_handle_args(int argc, char *argv[])
{
    int x=0, flag=0;
    if (argc==1)
        ss_display_usage();
    while (x<argc-1)
    {
        x++;
        if ( (strcmp(argv[x], "-file")==0) ||       /* -file */
            (strcmp(argv[x], "-f")==0) )
            if (argc == x+1)
                ss_display_usage();
            else
                strcpy(ss_scriptfile, argv[x+1]);
}
if ( (strcmp(argv[x], "-help")==0) || (strcmp(argv[x], "-h")==0) )
    ss_display_usage();

if ( (strcmp(argv[x], "-version")==0) || (strcmp(argv[x], "-v")==0) )
    ss_display_usage();

if (strcmp(argv[x], "-host")==0)
    if (argc == x+1)
        ss_display_usage();
    else
        strcpy(ss_host_name, argv[x+1]);

if (strcmp(argv[x], "-port")==0)
    if (argc == x+1)
        ss_display_usage();
    else
        ss_port_num = atoi(argv[x+1]);

if (strcmp(argv[x], "-url")==0)
    if (argc == x+1)
        ss_display_usage();
    else
    {
        ss_process_cmd(argv[x+1]);
        flag=1;
    }

if (strcmp(argv[x], "-quiet")==0)
    quiet_flag=1;
}
return flag;

#endif MOO
static int URLfor(char *file_name, Var arg)
{
    int the_case;
    the_case = 0;
    switch(arg.type) {
    case TYPE_STR:
        strcpy(file_name,
            "http://m42.mit.edu:8080/htdocrcat(file_name, ".gif >");
        the_case = 1;
        break;
    case TYPE_NUM:
        sprintf(file_name, "<IMG SRC =
            http://m42.mit.edu:8080/htdocs/webMOO/resource/Obj%u.gif >", (int) arg.v.num);
        the_case = 1;
        break;
    default:
        the_case = 0;
        break;
    }
    return the_case;
}

static int HTMLfor(char *file_name, Var arg)
{
    int thedaMOO Access httpwrite(text) *****/
    static package
    httpbegin(Var arglist, Byte next, void *vdata, Objid progr)
    {
        FILE *the_file;
        char file_name[256];
        Var r;

        r.type = TYPE_NUM;
        r.v.num = HTMLfor(file_name, arglist.v.list[1]);
        if (r.v.num== 0)
            {free_var(arglist);
             return make_error_pack(E_TYPE);} 
        the_file = fopen(file_name, "w");
fcst);
    return make_var_pack(r);
}

switch(arglist.v.list[1].type) {
    case TYPE_LIST:
        r.type = TYPE_NUM;
        r.v.num = arglist.v.list[1].v.list[0].v.num;
        break;
    case TYPE_STR:
        r.type = TYPE_NUM;
        r.v.num = strlen(arglist.v.list[1].v.str);
        break;
    default:
        free_var(arglist);
        return make_error_pack(E_TYPE);
    ion("gifwrite", 1, 1, gifwrite, TYPE_ANY);
    (void) register_function("httpwrite", 2, 2, httpwrite, TYPE_ANY, TYPE_STR);
}

#endif
#define MCCI_MAX_RETURN_TEXT 1024

#define MCCI_OK 0
#define MCCI_FAIL 1
#define MCCI_OUTOFMEMORY 2
#define MCCI_REQUEST_FAIL 3
#define MCCI_NETWORK_ERROR 4
#define MCCI_DEFAULT 0
#define MCCI_OUTPUT_NONE 100
#define MCCI_OUTPUT_CURRENT 101
#define MCCI_OUTPUT_NEW 102
#define MCCI_ABSOLUTE 110
#define MCCI_RELATIVE 111
#define MCCI_S_GET "GET"
#define MCCI_S_DISPLAY "DISPLAY"
#define MCCI_S_DISCONNECT "DISCONNECT"
#define MCCI_S_QUIT "QUIT"
#define MCCI_S_SEND "SEND"
#define MCCI_S_OUTPUT "OUTPUT"
#define MCCI_S_ANCHOR "ANCHOR"
#define MCCI_S_TO "TO"
#define MCCI_S_STOP "STOP"
#define MCCI_S_CURRENT "CURRENT"
#define MCCI_S_NEW "NEW"
#define MCCI_S_NONE "NONE"
#define MCCI_S_HEADER "HEADER"

#define MCCIR_OK 200
#define MCCIR_GET_OK 210
#define MCCIR_DISPLAY_OK 211
#define MCCIR_DISCONNECT_OK 212
#define MCCIR_QUIT_OK 213
#define MCCIR_SEND_OUTPUT_OK 214
#define MCCIR_SEND_STOP_OK 215
#define MCCIR_SEND_ANCHOR_OK 216
#define MCCIR_SEND_A_STOP_OK 217
#define MCCIR_ANCHOR_INFO 301
#define MCCIR_SEND_DATA_OUTPUT 302
#define MCCIR_UNRECOGNIZED 401 /* what's this? */
#define MCCIR_ERROR 402 /* does not follow protocol */
#define MCCIR_REQ_FAILED 500 /* request failed */
#define MCCIR_GET_FAILED 501 /* request failed */

#include "port.h"
typedef PortDescriptor *MCCIPort;
#ifndef PORT_DOT_H_BEEN_INCLUDED_BEFORE

#define PORTBUFFERSIZE 1024

typedef struct {
    int socketFD;
    char *serverAddress;
    int serverTCPPort;
    int connected;

    int numInBuffer;
    char buffer[PORTBUFFERSIZE *2 +2];

    /* callBack(void *callBackData) */
    void (*callBack) (); /* called when connection dropped*/
    void *callBackData;
} PortDescriptor;

#define PORT_DOT_H_BEEN_INCLUDED_BEFORE
#endif
Redistribution and use in source and binary forms are permitted provided that this paragraph is included. Any distribution of this software or derivative works must comply with all applicable United States export control laws.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Any person obtaining a copy of this software is requested to send their name and electronic mail address to:

ssgaudet@mit.edu or askwersk@mit.edu

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/time.h>
#include<arpa/inet.h>
#include"ioutils.h"

/*
#define CGI_SERVER_PORT 7778*/
/*
#define CCI_SERVER_PORT 7776*/

int socket(int domain, int type, short protocol);
int bind(int s, void *name, int namelen);
int connect(int s, void *addr, int addrlen);
int listen(int s, int qlen);
int accept(int s, void *addr, int *addrlen);
int close(int s);
int read(int s, char *buffer, unsigned buflen);
int readv(int s, struct iovec *iov, int count);
int recv(int s, void *buffer, int buflen, int flags);
int recvfrom(int s, void *buffer, int buflen, int flags, void *from, int *fromlen);
int recvmsg(int s, struct msghdr *msg, int flags);
int write(int s, const char *buffer, unsigned buflen);
int writev(int s, struct iovec *iov, int count);
int send(int s, void *buffer, int buflen, int flags);
int sendto (int s, void *buffer, int buflen, int flags, void *to, int tolen);
int sendmsg(int s, struct msghdr *msg, int flags);
int select(int width, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
#endif
The $webthing object

Object ID: #160
Name: webthing
Parent: generic thing (#5)
Location: Adam (#2)
Owner: Adam (#2)

Flags:
Verb definitions:
  init_for_core
  look_self
  image_url
  @setimage
  @getimage

Property definitions:
  image_url

Properties:
  key: 0
  aliases: {"webthing"}
  description: ""
  object_size: {0, 0}
  drop_failed_msg: "You can't seem to drop %t here."
  drop_succeeded_msg: "You drop %t."
  odrop_failed_msg: "tries to drop %t but fails!"
  odrop_succeeded_msg: "drops %t."
  otake_succeeded_msg: "picks up %t."
  otake_failed_msg: ""
  take_succeeded_msg: "You take %t."
  take_failed_msg: "You can't pick that up."
  image_url: ""

#160:"init_for_core" this none this
1: pass(@args);

#160:"look_self" this none this
1: if ("is_webbing" in $object_utils:all_verbs(player))
2: "we can be sure player is a webber."
3: if (player:is_webbing())
4: "Dont let it record any tell events";
5:     saved = player.recording;
6:     player.recording = 1;
7:   "do normal player:look_self";
8:     pass(@args);
9:     player.recording = saved;
10:  "Now do the web version:"
11:     $web_utils:httpbegin(tonum(player));
12:     $web_utils:httpwrite(tonum(player), this:image_url());
13:     $web_utils:httpwrite(tonum(player), "<p>");
14:     savedtell = player:divert_tell_html(1);
15:     pass(@args);
16:     player:divert_tell_html(savedtell);
17:     $web_utils:append_look_back();
18:     $web_utils:ccisend(tonum(player));
19:     else
20:     pass(@args);
21:     endif
22:     else
23:     pass(@args);
24:     endif

#160:"image_url"    this none this
1: return "<IMG SRC="" + this.image_url + "">";

#160:"@setimage"    any (on top of/on/onto/upon) this
1:   this.image_url = dobjstr;
2:   player:tell("You set the image_url to " + dobjstr + ".");

#160:"@getimage"    none (on top of/on/onto/upon) this
1:   player:tell("The image corresponding to this object is...");
2:   player:tell(this.image_url);