Integrating The World-Wide Web and Multi-User Domains to Support Advanced Network-Based Learning Environments

Lee A. Newberg, Richard O. Rouse III, and John A. Kruper
Biological Sciences Division Office of Academic Computing
The University of Chicago
Chicago, Illinois, USA
E-Mail: L-Newberg@UChicago.EDU

October 21, 1994

The World-Wide Web consists of over five million documents distributed throughout the world and organized by a web of hypertext links connecting related topics. Object-Oriented Multi-User Domains are text- and network-based applications that allow individuals to converse in real time with each other and to interact with user-defined simulation objects. As part of our efforts to develop advanced network-based learning tools, we describe how we have integrated these two highly powerful and popular Internet applications. Our integration merges the strengths of the World-Wide Web (font styles, images, hypermedia, and hypertext), with the strengths of Multi-User Domains (interactivity and easy construction of software controlled objects).

1 Introduction

1.1 The World-Wide Web

The World-Wide Web consists of over five million documents and a vast collection of database information distributed throughout the world and organized by a web of hypertext links connecting related topics. Reflecting the popularity and growth of this distributed information system, Web-based information is doubling every six months to a year. Much of the power of the Web is derived from its seamlessness and ease of use. By clicking the mouse a user can fetch a page of information from any computer worldwide without knowing anything about how a given page is located, generated, or retrieved. The Web is free and is accessible to anyone equipped with Web-client software such as *Phoenix*, *Mosaic*, *Netscape*, or *MacWeb*.

1.2 Multi-User Domains

Object-Oriented Multi-User Domain (MOO) software and related variations are network-based applications initially developed for role-playing games in which players interact with each other and a computer-created fantasy environment complete with a variety of simulated objects and characters. Originally named MUDs for Multi-User Dungeons, these virtual worlds blur the notion of identity, allowing human users to adopt alter-egos, move around, explore, and emote with real and virtual creatures, and play out rich role-playing scenarios and simulations.

More recently, MUDs and MOOs have been used to support distributed scientific or political communities and to function as adjuncts or even replacements to town hall meetings and scientific conferences and seminars. One example of a MOO used for academic purposes is BioMOO, administrated out of the Weizmann Institute in Israel. Users connect to BioMOO to meet, speak, and collaborate with colleagues distributed world-wide. Using the simulation and object-oriented features of MOOs, users can show and share data and even conduct virtual experiments. BioMOO has a number of different subject-specific rooms, including the Library, the Lounge, the Lab Wing, the Seminar Room, and the Mouse Colony room. Illustrating the variety of activities supported in these environments, users can in the Mouse Colony room anesthetize, dissect, and examine virtual mice, all the while speaking to and interacting with other scientists located in the room.

Any MOO is readily accessible to anyone with MOO-client software such as *tkMOO*, *Telnet*, or *Phoenix*.

1.3 Network-based Learning Environments

The development of global computer networks and the availability for students at all levels to access these networks provides new opportunities to create what educational researchers call a "community of practice" — a distributed collection of individuals collaborating and collectively working toward a shared set of goals. Network-based learning environments provide the computer-based tools to establish and support such communities.

Though still in their infancy, some notable Network-based learning projects exist. Efforts such as the Technical Education Research Center's (TERC) LabNET project, Northwestern University's CoVIS project, and the University of Kansas' Explorer Project all have made significant steps to bring the benefits of network-based knowledge to the individual or classroom-based computer.

Though these efforts demonstrate the potential of network-based learning, significant drawbacks still exist. For example, some projects require highly proprietary or platform-specific hardware and software, limiting access to the lucky few schools, teachers, and students that can afford the expensive components. Other projects utilize software that offers only minimal interactivity, or is difficult to use due to a "command-line" interface.

We wished to directly address these limitations and to develop a next generation suite of tools that could support and further advance network-based learning projects. Specifically, we set out to marry the many strengths of the World-Wide Web to those of Multi-User Domains.

In a seamlessly integrated Web/MOO environment, information available via the Web also includes the highly interactive information that originates in MOOs. The powerful configurability and interactivity provided by MOOs is no longer restricted to text-only and is instead free to use the full hypertext and hypermedia capabilities of the Web, including Web links, images, sounds, movies, fill-out forms, and of course, plain text.

One simple example of how this environment can help build a community of practice is seen in on-line discussion sessions. Here, users dispersed geographically can view a page displaying an ongoing problem-solving discussion for a class. To participate in the discussion, users need only type what they wish to say or ask, then hit the return key. Because the Web technologies are incorporated into the MOO, the discussion section page can in addition contain graphs of data, digital videos, and links to additional supporting information.

In another example, professors can use these environments to show slides to students in cyberspace classrooms using a MOO slide-projector object. A MOO command such as "`show next slide`" would send the slide image simultaneously to all students within the same room. The slide projector could also be programmed to send accompanying text; Professors could also have the option of supplementing that text with their own comments via the MOO `say` command.

An interactive graphical dissection program is yet another possibility. Content experts could design a MOO object to react to various dissection commands from a student. The output of the MOO would include the usual text but could also include graphics showing the results of the student's actions. Through the power of the Web, the student could send dissection commands by the click of a mouse. For instance, the student might click on an image of a frog at the location she desires to start the next cut.

2 Goals & Considerations

We wish that the integration of the Web and MOO technologies be as seamless as possible in that any protocol-independent mechanisms by which a user requests information from normal Web-server software must also function when that server is a MOO. Conversely, any presentation possibilities available to a regular Web page must also be available to those that are MOO generated. These design and implementation goals are reflected in *Phoenix*, the WYSIWYG Web editor/browser we have developed.

We wish that *Phoenix* be able to communicate with any MOO (or related piece of software) and not just those specifically designed to understand *Phoenix*. A primary feature of a Web browser is its ability to access information on the Internet even though it is presented in many formats and retrieved via several mechanisms. In keeping with this spirit we wish *Phoenix* to be able to handle all MOOs, not just a few.

To properly incorporate MOO technologies we must overcome the asynchronous communications and statelessness that are an inherent feature of the Web. As currently implemented, a Web client makes a connection to the computer acting as a Web server, makes a request for information, receives a reply, and then breaks the connection. This communication is *asynchronous* in that the connection is broken and there are no provisions for allowing subsequent communications that progress in real-time. The communication is also *stateless* in that the Web server does not remember any information about the requesting client or its configuration state once the connection is broken. The Web's asynchrony and statelessness make conversations difficult and must be overcome in a proper integration with MOO technologies.

Fill-out forms, available with many Web clients, currently provide the Web's best approximation of a synchronous conversation. A page may be generated based upon and as a response to an electronic form or search request, filled-out by a user and submitted to a remote computer by the Web client. Although this mechanism is highly useful as a tool for getting information from the user to tailor a response, it does not provide real interactivity. For instance, it still requires the user/client to initiate every transaction. There is no way for the remote computer to initiate a transfer of information, as would be necessary in the case when another person within the same cyberspace classroom "speaks." In a forms-based approach to communication, a user would not be able to hear what other people said in response to his last remark until *after* his next remark.

The Web's statelessness must be overcome as well. Currently, unless sophisticated software for page generation is used to keep state (*i.e.*, accounting information), a remote computer cannot "remember" previous requests from the user and is unable to adjust its responses accordingly. To take full advantage of our cyberspace classroom and its learning aids, the system must be able to remember and build upon the past.

3 Implementation

We designed the Web/MOO integration with the above goals and considerations in mind. The primary decisions involved:

- Uniform Resource Locators describing access to MOO-server information from Web clients;
- a protocol for communications between Web clients and MOO servers; and
- the presentation of the communications in *Phoenix*.

3.1 Uniform Resource Locators

In the Web, each Web page can be located by a *uniform resource locator* (URL). Generally, a URL specifies the computer from which to fetch a page, the network port to contact, the filename containing the page, and the name of the transfer protocol to be employed in requesting the page. (Not all of these fields are present for each protocol and some protocols have additional fields.) We are establishing a Domain Server Transfer Protocol (dstp) for URLs. It describes to Web clients where and how to fetch a piece of information from a MOO server.

It should be noted that as a typical user, one would not need to know the URL of the requested page. This technical information is stored and managed by the Web client and is invisibly accessed when the user clicks on a hypertext link. The URL template for access to a MOO is given by

```
dstp://computer:port/prompt(/connection)*[#command(:command)*]
```

where

- "(...)*" means zero or more occurrences of the contained string.
- "[...]" means zero or one occurrences of the contained string.
- computer is the computer running the MOO server.
- port is the network port on computer at which the MOO server is expecting requests.
- prompt is a string of the last few characters the MOO server sends when it is ready to receive a command. When the initial connection to the MOO server is being established, any following connection pair(s) are used instead.

- Each connection is a connectprompt=connectresponse pair used to establish a connection to the MOO server if no such connection yet exists. They are used in the order given. connectprompt is a string of the last few characters the MOO server sends when it is ready for the Web client to send connectresponse. If connectresponse is empty, the user is prompted for the value. If in the second or following connection, the connectprompt is empty then the connectresponse (or, if empty, the value supplied by the user) is sent immediately after the previous connectresponse.

  For instance, a connection of `/login%3A%20=/Password%3A%20=` where `%3A%20` is the URL encoding for a colon followed by a space requests a login name and password following the prompts "`login: `" and "`Password: `", respectively. A connection of `/Hello%3A%20=login%20/=/=%20/=` would send "`login `" followed by a login name, a space, and a password after the prompt "`Hello: `".

- A command is something users can type to the MOO server after logging in. These are to be sent to the MOO server, in order, once a connection has been established. One command followed by a line terminator is sent after each occurrence of prompt until they are exhausted. For example a command might move a user to a particular classroom; create, find, or destroy an object; or start a MOO-object slide projector. Generally, the MOO server will give feedback to the user for each command, even for those commands that "fail" (*e.g.*, a user who wishes to start a slide projector, but cannot because one is not in the room). With the feedback, the user has the option of responding with subsequent actions.

  Certain types of Web pages known as ISMAPs, ISINDEXes or FORMs cause strings to be appended to URLs. In such cases, the appended string is appended to the last command. This changes the actual command name unless the command ends with a space or other separator recognized by the MOO server. This appending mechanism would enable user-customizable manipulations of a MOO.

This syntax was chosen because it describes where and how to access any information that a MOO server might provide. The computer/port pair uniquely identifies the MOO server. The connection pairs are sufficiently general to allow automated login to any MOO server encountered. The simplicity of the command syntax allows any command to be submitted to the MOO server. Finally, information from the user can be supplied to the MOO server by appending to the URL in any of the ways normally employed to send information to Web servers.

3.2 Protocol

To ensure that *Phoenix* is able to communicate with existing MOO servers, we have to keep the protocol simple. We use *Telnet*, the software used for remote logins. Note that the connection is not broken after the initial response of the MOO server, but is held open until explicitly closed with a MOO command, which may be a URL, or after a specified time-out period has expired. This means that at any given time, *Phoenix* may have open connections to several MOOs. A drawback to using *Telnet* is that the automated format negotiation mechanism of http is lost. However, this is unavoidable since we wish to support existing MOOs that do not understand this mechanism.

Commands from the Web client are exactly the same as users (or the MOO client software) would type to a MOO server. MOO servers wishing to present more than plain text would support commands that allow users (or the Web client software) to specify presentation formats. For example, the command "`set MIME=True`", which typically would be specified in the connection part of a URL, would be a request that the MOO server use MIME rather than plain text. In such cases, the response from the MOO is interpreted according to the Multipurpose Internet Mail Extension (MIME) standard. A MOO server that wishes to present hypertext using the HyperText Markup Language (HTML), for example, may issue a "`Content-type: text/html`" line. If no content type is specified it is assumed that the MOO server is sending plain text. Finally, the MOO server may switch among MIME types by proper use of the MIME type `multipart/mixed`.

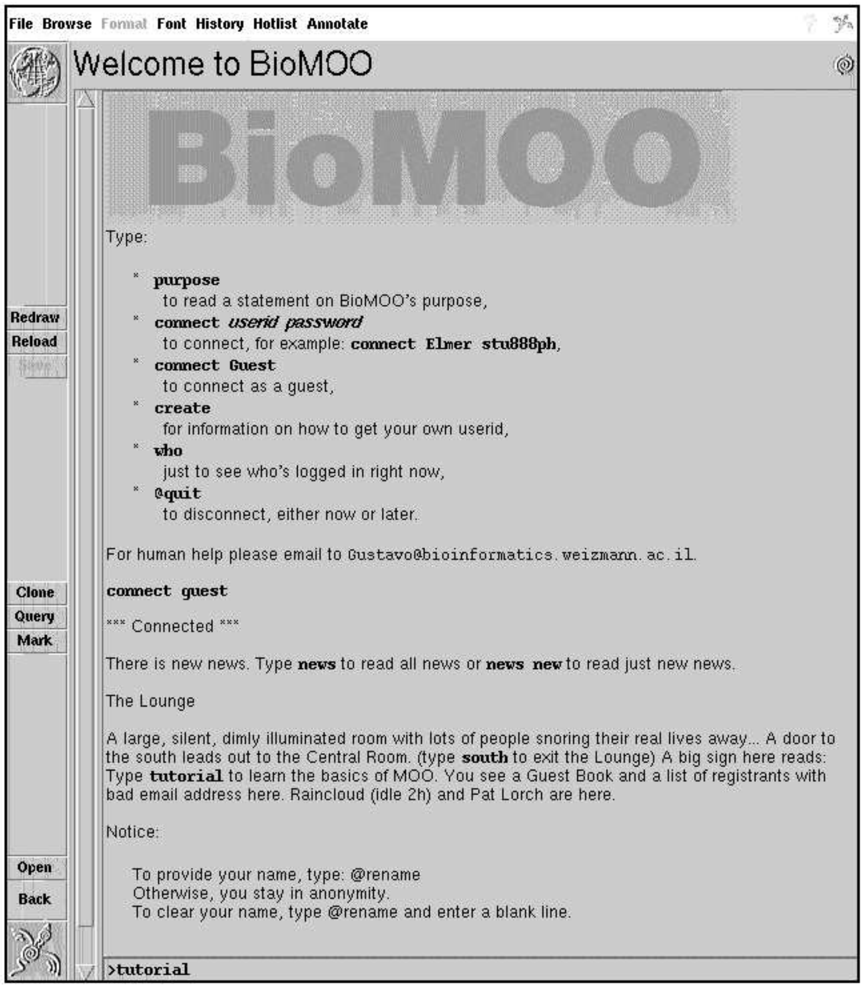
In addition to the above protocol considerations, the Web client should record communications from each MOO server even if the client is not currently displaying the corresponding pages.

The choice of *Telnet* and MIME, two widely implemented standards, ensures easy migration of the protocol to many hardware/operating-system platforms. The system that runs the MOO server may be of any type and need not be the same as the one running *Phoenix*. This gives great flexibility in the use of hardware and software and is in line with the easy extensibility that has made the Web so popular.

3.3 Presentation

In our implementation, the region of the *Phoenix* window that normally displays a viewed page is broken into two parts. The upper part displays output from the MOO server. The lower part holds one line of text and is where the user types commands to the MOO server directly. The upper section can scroll once it becomes full.

A single page (with the two sections just described) is used for each MOO server. Clicking on a URL to a MOO already opened returns the user to the existing page for that MOO and issues the `commands` in the URL.



```
Type:

  * purpose
      to read a statement on BioMOO's purpose,
  * connect userid password
      to connect, for example: connect Elmer stu888ph,
  * connect Guest
      to connect as a guest,
  * create
      for information on how to get your own userid,
  * who
      just to see who's logged in right now,
  * @quit
      to disconnect, either now or later.

For human help please email to Gustavo@bioinformatics.weizmann.ac.il.

connect guest

*** Connected ***

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

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) A big sign here reads:
Type tutorial to learn the basics of MOO. You see a Guest Book and a list of registrants with
bad email address here. Raincloud (idle 2h) and Pat Lorch are here.

Notice:

    To provide your name, type: @rename
    Otherwise, you stay in anonymity.
    To clear your name, type @rename and enter a blank line.

>tutorial
```

Figure 1. Web-Enhanced BioMOO Example

Information may be sent to the MOO server by clicking on a hypertext link in the upper window, by typing, or by cutting and pasting into the lower window. In particular, users can send HTML (containing formatted text, images, and hypertext links) to the MOO server. For example, with the MOO server's `say` command, a user can easily send images to the other classroom occupants. This would be accomplished via the HTML `<IMG>` markup (which in addition requires the URL describing where to find the image to be displayed). In *Phoenix*, this markup and corresponding URL is automatically and invisibly supplied when the user copies an image from any Web page and pastes it into the lower window.

[Fig. 1] shows one way that the a MOO could take advantage of the capabilities provided by *Phoenix*. The figure shows the *Phoenix* client connecting to a Web-enhanced BioMOO at the Weizmann Institute.

Through the use of the Web's Hypertext Markup Language and *Phoenix*'s powerful hypermedia and hypertext presentation formats, the BioMOO easily makes use of images, Web links, and many font styles and sizes. Rather than a bland *vt100-style* output, the BioMOO now offers its welcoming banner as an image. After the banner, the BioMOO displays user options in a nicely formatted list structure wherein user-inputed text is distinguished from the accompanying BioMOO text by the former's presentation in a different font.

By varying fonts, the BioMOO easily lends intuition, letting a user know to type the text connect literally but then to substitute in an actual userid and password where the keywords userid and password appear. Additionally, the BioMOO provides an e-mail address in the easily-read, fixed-width courier font so users will have no difficulty distinguishing similar-looking alphanumeric characters.

All text from the BioMOO is interlaced with the text typed in the upper window. The bottom window is used to compose messages to send to the BioMOO server. This separate window allows users to enter commands uninterrupted by the continuing flow of the events that occur in the upper window. When a user hits the return key, the text is appended to the upper window and is sent to the BioMOO.

4 Conclusion

We believe that incorporating the advantages of Web technologies with those of MOOs hold great potential to improve network-based learning environments. Striving for simplicity, extensibility, and ease of use, we have integrated the Web and MOO applications in our Web browser/editor, *Phoenix*. In addition to further enhancing our *Phoenix* client, we are currently designing MOO/Web servers that take advantage of these newfound capabilities.

Further information about *Phoenix*, the Web, MOOs, and other technologies discussed in this article can be found using the Web, accessing the sources listed below.

References

[Berners-Lee, 1995] Berners-Lee, T. (1995). *The World Wide Web.* CERN, [`http://info.cern.ch/hypertext/WWW/TheProject.html`].

[Borenstein & Freed, 1992] Borenstein, N. & Freed, N. (1992). *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies.* [`http://info.cern.ch/hypertext/WWW/Protocols/rfc1341/0_Abstract.html`].

[Braverman, 1994] Braverman, A. M. (1994). *Technical Information and Specifications.* NCSA, [`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/d2-tech.html`].

[Connolly, 1995] Connolly, D. W. (1995). *HTML Specification Review Materials.* HaL Computer Systems, Inc., [`http://www.hal.com/users/connolly/html-spec/`].

[Glusman, 1995] Glusman, G. (1995). *BioMOO, the biologists' virtual meeting place.* Weizmann Institute, [`http://bioinformatics.weizmann.ac.il:70/1s/biomoo`].

[Murray-Rust, 1994] Murray-Rust, P. (1994). *MOOs and MUDs, especially BioMOO.* [`http://seqnet.dl.ac.uk:8000/vsns-pps/technology/biomoo.html`].

[NCSA, 1994] NCSA (1994). *A Beginner's Guide to URLs.* [`http://www.ncsa.uiuc.edu/demoweb/url-primer.html`].

[NCSA, 1995] NCSA (1995). *A Beginner's Guide to HTML.* [`http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html`].

[Newberg, 1995] Newberg, L. A. (1995). *Announcing Phoenix: A Genuinely WYSIWYG HTML Editor.* The University of Chicago, Biological Sciences Division, Office of Academic Computing, [`http://http.bsd.uchicago.edu/~l-newberg/phoenix.html`].

[Reilly, 1994] Reilly, C. (1994). *MOO-WWW.* Trinity College Dublin, [`http://www.maths.tcd.ie/pub/mud/moo-www/directory.html`].