Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 1993

"PanaeaMud An Online, Object-oriented Multiple User Interactive Geologic Database Tool"

Erich Boring
Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1993-010

PangaeaMud
An Online, Object-oriented Multiple User Interactive Geologic Database Tool

Erich Boring

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

PangaeaMud

An Online, Object-oriented Multiple User Interactive Geologic Database Tool

by

Erich Boring
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #93-010 12/93

PangaeaMud

An Online, Object-oriented Multiple User Interactive Geologic Database Tool

Master's Thesis Project

by

Erich Boring

December 3, 1993

Abstract

This paper provides an overview of the design, development, and use of the PangaeaMud Database System. Section I gives an introduction to pertinent concepts and discusses previous work in the area. Section II is devoted to the non-technical aspects of the system. A brief user's view of the system is provided, along with discussion of the internal environment utilities. Section III illustrates the workings of the system from the programmer's viewpoint and contains information on the main entity relationships within the database and their implementation.

Contents

I. Introduction

A. Purpose

There exist problems of communication, information retrieval, information storage, and interdisciplinary data usage in many fields at the present time. Students and professionals alike are hampered by lack of reference skills, time, and effort in finding information they need for their work. For these reasons and others, modern professionals are in need of various tools to let them find data, test hypotheses, and communicate their findings at the rapid pace required today.

PangaeaMud is one such tool. PangaeaMud provides the user a friendly virtual environment in which anyone with Internet remote login capabilities may convene for real-time meetings, exchange discourse, and access stored geologic information in user friendly, natural language ways, rather than utilizing complex or arcane retrieval languages.

PangaeaMud was created by modifying a currently available free software package written in C and a variant of C++ known as LPC (Lars Penji C). Specific objects and classes were designed and coded for the object database, functions for the object interfaces were created, and the freeware was stripped of unused code to improve the performance and space requirements. Finally, the database was loaded from the Encyclopaedia of Mineralogy and other definitive texts [11], [13], [18], [26], and the system prototype was made available for use on the machine known as phoenix.

B. Previous Work

The Multi-User Dimension (MUD) is not a new concept, or even new in operation. Since the time when our ancestors first visited tribal magic workers who were thought to be able to travel in spirit to observe distant happenings or even send messages, man has had the idea; the MUD is merely the latest expression of this desire to communicate in a virtual world. In this latest incarnation, the MUD, there is finally a network of virtual meeting spaces for the exchange of ideas and information.

At least two similar projects are already in existence, Xerox's Jupiter Project, which virtually models Xerox's research laboratories in Palo Alto, California and Birmingham, England. In the virtual model, the two are tied together, allowing researchers from either facility to exchange information and mingle socially in real-time [9], [10].

The second work-oriented MUD is the MIRE Project at MIT. MIRE (Multi-user information retrieval environment) utilizes a base MOO software to enable astrophysicists to hold meetings and retrieve and display data available through Internet "gopher" facilities [16], [20].

Classically, papers upon MUD research have been divided into two major camps - ethnographic reports and technical reports. Hard science coders are trying to show the usefulness of the MUD within

a framework of the kind of software and hardware of which the MUD is composed, while the soft sciences seem drawn to the MUD as a new frontier in social behaviour phenomena [8],[25]. This paper will aim for a balance of the two, though a balance decidely favoring the hard science view, as almost all of the social aspects of mudding will be covered in this background section.

C. What is a MUD/MOO/MUSE?

At the lowest level of commonality, all of these are virtual environments residing on a server to which any machine with remote login capability may establish a connection over the Internet [30]. The user may then create a virtual icon to represent himself/herself in this environment. Multiple users may interact in these environments in real-time, rather in the manner of teleconferencing. Depending on the wishes of the administrative coders of the environment, the remote users may have a greater or lesser power to define this icon, to express their ideas, to exchange information, or to create additional software within the environment to actually permanently change the environment.

Although currently most of these environments have a text-based interface only, a few are expanding into the world of graphics on X-terminals, and a very few, such as the Jupiter Project at Xerox Palo Alto Research Center, are working to include realtime video and audio capabilities [10], [15].

The original MUD was written in 1979 by Richard Bartle and Roy Trubshaw of the University of Essex, England [3], and started rather an avalanche of varieties of the software, written in numerous languages. Of the numerous offspring, there may be said to have formed two distinct schools of MUD use, the adventure game MUDs, which generally retain the word MUD in their names (AberMUD, LPmud, DikuMud, KMud) and the social interaction MUDs, which tend to use a variety of acronyms (MUSE, MOO, TinyMud, SMUG, TinyMuck, MUSH, LambdaMOO) [30], [31].

Most of the higher level work on MUD research is being performed within the confines of this second group, where coding internally takes place in interpreted languages which support object orientation such as LambdaMoo, which was originated by Pavel Curtis of Xerox PARC [8], [9]. The second major division in MUD architecture is based on the language used to develop the MUDs. The adventure MUDs are usually coded both externally and internally in C and C++ variants such as LPC, while the social MUDs tend to switch internally to interpreted language coding. PangaeaMud is a hybrid, as it maintains the C coding structure as do the adventure MUDs, but is more similar in environment and function to the social MUDs.

D. Developing PangaeaMud

The idea for PangaeaMud came about when a decision was required for the author's Masters thesis project. The MUD environment was sufficiently complex that it tied together almost all of the coursework required for the degree - programming, database, information systems, data communication, design of data

structures, and even operating systems. As the Masters' candidates are encouraged to utilize their undergraduate fields in their projects, a geologic software package was suggested.

Informal surveys of local geologists suggested that such a virtual meeting place, with mail and data manipulation capabilities would be welcome. Thus the core concept of PangaeaMud crystallized. The beginning work was unguided, however, as the intention was to run the MUD from the Dell Server in the geology computer lab in Shideler Hall.

In preparation, I finished the setup of the Novell Netware 3.11 network for the lab, installing the cabling, setting up user accounts, and configuring the system. At this point, I went in search of the base MUD software, only to realize that my experience was only with those running under a UNIX environment, not MS-DOS.

I reluctantly tabled plans for the MUD being based on the Gondwananet system in Shideler and requested space on the Applied Science RISC/6000, phoenix, using AIX 3.0. With the account in place, I searched the UseNet bulletin boards for source sites for MUD software. I used ftp to acquire several versions of the base software from various sites, but kept running into compiler problems when trying to create the MUD driver software, as single programs within the driver system would fail to compile.

Finally settling upon the MudOS driver software version 0.9.18 and the TMI-2 mudlib version 1.1.0, I was able to compile the driver source after only minimal code changes under bsdcc. Further work was needed to to change the runtime configuration files and directory structure of the mudlib to prevent the driver from crashing upon execution. A path error in the config file also worked to prevent anyone from logging in for a while, but was corrected. Currently, the driver is running smoothly, starting the mudlib, preloading MUDwide daemons such as weather patterns, and opening the mud for logins.

II. Inside the MUD, a User's Viewpoint

A. Sample Internal View

It is at this point that most of the ethnographically oriented papers all tend to feel the urge to abstract the exact same section from one of Pavel Curtis's papers upon MUDding and start writing as if they were children's book authors [8], [22]. This will be avoided in this text, and samples of possible, though not actual environments, dialogue, and commands will be examined. To paraphrase Curtis [10], the strength of the mud system lies largely in the intuitively obvious nature of most commands, as they appeal to the user's real-world experience.

Here begins a sample login session.

```
$ telnet 134.53.3.230 3000

VM TCP/IP Telnet V2R2
```

Connecting to 134.53.3.230, port 3000

Using Line Mode...

Notes on using Telnet when in Line Mode:
- To hide Password, Hit PF3 or PF15
- To enter Telnet Command, Hit PF4-12, or PF16-24

(User hits return)

Welcome to PangaeaMud (version 0.9.18)
Please use the name 'Guest' if you just want a look.

PangaeaMud is running the TMI-2 1.1 mudlib on MudOS 0.9.18

Current users: Archcoder, Takacs.

By what name do you wish to be known? (User types erich)

"Erich" is a new character.
Is this really the name you wish to use? (y(es) or n(o)): y

As you're logging on a new character name, we'll assume ...
(A non essential paragraph of text is displayed here)

Please enter a unique password for your character:
(User types password, but it does not appear on screen)
Please reenter your password to confirm: (User types it again)

Your gender can be male, female, neuter, or hermaphrodite.
Please enter your gender: (User enters male)

Please enter your email address (user@host): (user types in boring@phoenix.aps.muohio.edu)
Please enter your real name: (User types Erich Boring)

(At this point, whatever startup messages for the mud appear, and the user is placed at the default start environment.)

Welcome to PangaeaMud! PangaeaMud is a virtual environment dedicated to the conceptual modeling of geologic processes and mineral structures, the exchange of news and information of use to those in the geologic scientific and educational fields. We provide bulletin boards on various topics, secure mailing facilities, tools for altering the constructs, and the ability to create and alter your own constructs and models. Eventually, we hope to include gopher capabilities, so you will be able to pull information from anywhere in the net. To use the help facilities, merely type help at your prompt.

Welcome to PangaeaMud!
** No new News to read **

:Press ENTER to continue:

This is the foyer of Banzai Research Laboratory's Geoscience Division. To the west lies the Post Office, to the east is the main conference room, south lies the divisional archives, reference libraries and bulletin boards, and north leads out into the rest of the building. Obvious exits are: north, south, east, west.
> (User types help)

* Basic Commands *
~~~~~~~~~~~~~~~~~~~

Commands always begin with a verb unless they are aliases. There is a set of standard aliases for common commands, type "alias -global" to see it.

Some useful commands to get started are:

faq	this displays the list of Frequently Asked Questions
say <msg> or ' <msg>	your most basic communication command
go <direction>	the most basic movement - moves in a direction
get <object>	attempts to move an object to your inventory
drop <object>	attempts to move an object to your surroundings
look	used in many ways to examine your environment
alias, unalias	used to set, change, and remove aliases
help <command>	gives additional help on the command, if available
help <topic>	gives more general help
help start	gives a more lengthy explanation of this mud
help topics	gives a list of the help topics available
help commands	list all the available commands
help help	How to use the help command

(User decides to try a few commands.)

> say hmmmm.
Ok.

(Others in same room environ, if there are any, see:)
Erich says 'hmmmm.'
> who

There are 3 users connected.
EST time is Thu Dec 3 15:35:02 1993

Name	Idle
Archcoder the system Admin	21m
[Takacs the Senior coder]	
Erich	

> south

/main/archives/room1
This is the entry room to the archives, bulletin board areas, and other reference areas of Banzai Research Labs Geoscience Division. To the south lies a door through which you see a room filled with
~~~~~~~~~~~~~~~~~~~

books from floor to ceiling, while to the east, you see a hallway lined with bulletin boards. Obvious exits: north, south, east.

>

As is readily apparent, in this case the environment conforms to a building, and movement is accomplished by choosing a direction, various types of communication actions are possible, and objects model a realistic environment. Had the Erich icon continued, he might have read or posted a note on a bulletin board, chosen a reference book to look up data, possibly even created new rooms or objects to place therein to expand the world of the MUD. Had he come across either of the other users, or both of them, they may have engaged in conversation, passed data, or merely passed each other by.

B. Why the PangaeaMud Environment?

By using a parser built to accept a variety of common commands, a great deal of realistic modeling may be achieved, whether for creating the consensual virtual space for user interaction, or for the specific scientific work within the environmental framework. In addition, the internal coding ability allows the coder to create even more specialised parsers for use with the models they create, whether an object like a microscope, a room such as a mine shaft, or a metaclass like a mineral.

As has been mentioned, most research work is done on the type of MUDs that employ the internal interpreted language, but PangaeaMud retains the use of C externally and C++ internally, to aid in ease of coding. All major changes to the virtual environment are thus inserted into the system network in C, while virtually everything that will be used directly or inherited only locally is coded in the C++ variant LPC.

Rather than forcing each user to learn an entirely new language in which to code, the power and versatility of C is provided, along with an ease of portability, as the coder may work on his/her own machine and then upload files to the confines of the mud, instead of using the editor provided. C may not be a universal language, but it is decidedly more common than such artificial MUD languages as the interpreted language U, which must be run through extra processing by the system.

As PangaeaMud will be created as a registration-required MUD, users will be limited to those actively involved in the Geologic community, screening out the casual user who might log on from curiousity or just happened to find the Internet address. This issue will be dealt with in more depth in the section on Security.

C. But Why MUD?

Empirical evidence suggests that benefits are possible by interacting with others via computer as opposed to in person [7]. Nuances of expression and body language are stripped away,

requiring participants to be more precise in their language in order to communicate effectively, yet other factors such as biases based on prejudice are also able to be dismissed. Users feel freer to express ideas they might not in personal contact, as if the computer provides a shield from contempt [28].

To an extent, this is because a level of anonymity is provided by the system [23]. The user, once registered, may choose almost any name or description for the personal icon, and only the system administrator must know who that user actually is. No one need know what gender, what race, or what beliefs he or she holds. The user alone controls the level of personal data he/she allows others to know, setting the stage for impartial interaction in which only the important information about a problem will be considered, not who might have posed it, although most users are sufficiently self-assured to actually state their gender [27].

D. Embedded MUD Features

MUD software usually comes with several useful features already coded: e.g. real-time communication between users, mail facilities, and security features. A line editor is provided for coding, and file transfer and remote login abilities are sometimes provided. The base PangaeaMud software provides all but file transfer, allowing the administration to decide the file transfer protocol choice.

1. Communication

Communication may be achieved within the MUD in a number of ways. Besides a variety of commands that mirror verbal communication, a "channel" system allows users to receive messages about a given topic much as if they were sitting with a bank of radio receivers. Both individuals and environments may be set to "muffled" so that private conversations or conferences may be conducted free of distractions.

Several caveats must be provided at this point. First, a number of the communication commands can simulate anonymous messages or even communiques from users other than those of the sender, and thus their usage is logged so as to prevent spurious messages. Also, gwiz and interwiz, two of the channels, are links to a network of other MUDs and are to be sparingly used as they consume an inordinate amount of CPU time and Internet resources. A list of real-time communication commands and examples of their use may be found in Appendix B [33].

A mailer is provided, so that secure communications may be sent from user to user, whether or not the recipient is currently active. The mailer uses a special subset of the ed line editor. Once lines are entered, they may not be changed or deleted, though, so a more expedient way to send notes is to enter the full ed mode, create a note file, then copy it to the /open directory of the recipient.

The mailer does allow copies to be kept of communiques after they have been read though, as well as enabling copies to be sent to multiple recipients. Muffled rooms are provided for explicit

use of the mailer, so that shouted messages elsewhere will not interfere with the composition of communiques.

The other main mode of delayed communication is via bulletin boards. Objects that function as realworld bulletin boards are provided. Users may place and remove notes on them. Notes are created in similar fashion to mail messages in an ed subset. Typically bulletin boards are open to all users. By placing bulletin boards in virtual areas accessible only to certain icons, their usage may be restricted.

2. MUD Security Features

MUD security is fairly unobtrusive from the user's viewpoint, but nonetheless vital. First, passwords are required for connection to the MUD, and are encrypted so that even the MUD administrator does not know the individual user passwords. Next, as the user enters the MUD, he enters into a hierarchy of permissions which allow him to view filenames in directories, read, write, or execute files, and clone copies of objects or copy the code directly. Either the adminstrator or the programmer responsible for a given domain, such as driver maintenance or mudlib coding (creating rooms, objects, and commands), will decide what level of permissions are suitable for each user, depending on his or her skills and interests.

III. The Coder's Viewpoint

Once the user decides that the current state of PangaeaMud does not fully satisfy his needs, he may either ask one of the other coders to implement a virtual modeling tool for him or code it himself. This section is geared towards the user who wishes to realize the full utility of the system by coding specific tools to manipulate, retrieve or store data. A brief description of C and C++, object oriented programming, and object design is followed by detailed security information and coding specific to the PangaeaMud project.

A. The Utility of C and C++

The C family of languages is well suited to the design of object oriented work, and thus such offshoots as object databases and artificial intelligence work [19]. PangaeaMud driver code is written in C, and contains the source code to utilize the C++ variant LPC that internal coding is performed in, giving the full power of C as well as the added features of LPC to the users who decide to create their own modeling tools or AI systems within the MUD.

The way in which C can be used to represent and store data under an object oriented framework provides the basis upon which LPC, with its built in message passing, parsing, and file handling functions, allows users to code highly powerful object interfaces with relative ease.

B. Object Orientation

A thumbnail sketch of the tenets and concepts of object orientation and their use by the MUD is in order at this point. Object orientation is still a bit of a buzzword, but a number of key features have come to be accepted as prerequisites for a system to be considered object oriented.

The object itself is at the root of the paradigm. An object is a variable or set of related variables and associated operations that correspond to either a real object or an abstraction, and may be made up of sets of other objects [12]. The object is a way to encapsulate data, presenting the user with an interface that shows only the information relevant to his/her needs, much in the same the olfactory system works. Each interface site has a certain logical shape, as it were, and only those objects with congruent interfaces can trigger data interchange.

Objects come in several forms. At the highest level is the metaclass object. (In PangaeaMud most of these objects reside in the /std directory.) The metaclass may be viewed as an abstract data types or as a base upon which to build other objects. Thus, the mineral metaclass object, which corresponds to no actual mineral but which has attributes common to all minerals, is found here. Likewise the object metaclass object is in the /std directory.

A class object, for example, might be the object code for gold. It takes on common features of minerals, such as the ability to leave a streak, but also has its own specific attributes, such as being a malleable metal.

At the low end of the spectrum is the individual instance of an object, which has its own set of variable attributes that may change without effecting changes on the classes or metaclasses it is derived from. Thus, a copy of the gold class object might be brought into existance via the "clone" command, and then could be deformed when the 'fractured' flag is set. As long as the instance is in existence, it will remain 'fractured', yet neither the class nor metaclass objects will be changed. This behaviour allows coders to update their code interactively without need for recompiling the entire system.

As object oriented systems allow for inheritance, where class objects take common attributes from higher level class objects, it can be seen that the object-oriented structure is typically hierarchical, with class objects inheriting features from metaclass objects, and individual instances inheriting attributes from class objects.

C. Object Design

Once the ideas of the object oriented paradigm are assimilated, object design becomes fairly simple within the MUD framework. Coders have to decide which of several ways to present and access data depending on the access they wish to allow others. The driver software, which contains LPC function libraries, data communication software, and file handling tools is usually left unchanged. The base mudlib software however, is modifiable even

when the environment is running, and is modular for extensibility.

Typically, the coder will choose to work either with command functions such as those found in the /cmds directory and its subdirectories, or to design object metaclasses, classes, and instances. If a command is created or modified, it becomes available to a subset of the users and provides one more interface site between the set of users and subsets of other object types as specified [29]. If matching interface sites do not already exist or are not added to other objects, responses return null or false values.

If the coder decides to work with classes of objects, he/she will generally need to inherit the OBJECT superclass, then design his or her own metaclass and work downward. The main metaclass objects are standard features of any MUD standard code library, and object design entails users first deciding which, if any, major metaclass most nearly suits the need, then altering a copy of the selected metaclass to create two additional files, a standard metaclass file and the individual instance of the metaclass.

Additional instances may be created including or inheriting data from the metaclasses as needed. The instances are then copied into the Active Object database as needed. Two examples may be found later in the section on PangaeaMud specific code, the design of minerals and the streak plate, a simple testing tool to extract data from the mineral instances.

D. Security and the Coder

Taking the step into coding, the user needs to be more fully cognizant of the various aspects of MUD security. Two files control the various levels of access within the MUD. From the mudlib directory (tmi2lib_1.1) they are /adm/etc/groups and /adm/etc/access. In the groups file are sets which have various levels of permissions, into which users may be sponsored. The access file then specifies which directories each group has read and/or write permissions to.

Another aspect of security is that a number of commands are level dependent, and users will only have the directories in which these commands reside added to their command path upon promotion to the appropriate level. Virtually all of the powerful user commands will open the access files to check for a user's permissions before executing [33].

The hierarchy of coders starts with the admins, those who have root access to the system and are assumed to have login capabilities to the root account. Next come those archcoders who have control over and responsibility for specific domains within the game. Only these two top levels of coders may promote lower level users to coding status within the game. Several lower levels of coders exist, each having slightly fewer abilities than the last. At all levels of coding, a specific domain should be assigned, so that the chain of command is clear, though nothing stops any coder from coding for any domain. As a final note, all domains fall under the /d directory and each coder is assigned a coding workspace under the /u directory in the subdirectory that matches the initial of their chosen name.

E. PangaeaMud Specifications

Code changes specific to PangaeaMud are enumerated in Appendix A. This section will go over the types of changes made to the system, point out difficulties with the database architecture, and examine several pieces of code in detail with an eye to design and function.

1. Cosmetic Changes

The TMI-2 (The MUD Institute II is a MUD where LPC and coding design are taught and online technical support for the MudOS driver and TMI-2 mudlib is provided.) mudlib provides a limited number of virtual environments, several handy data extraction tools, some autodaemons for handling various complex tasks such as logins and weather control, and data directories and files. In the course of working up PangaeaMud, one set of changes was merely altering the appearance of the basic rooms provided and attaching the specific rooms created for 'Banzai Research Laboratories', the starting area created for the users of the geologic research facilities. These files are generally to be found in the /d/Fooland directory, as more importance was attached to creating the basic geologic modelling tools than to renaming directories or moving files.

2. Functional Changes

Slightly more serious changes were needed in several places to aid indirectly in changing the MUD from a basic adventure game into a more useful environment for social interaction and research.

The first type of these changes was that needed to set up the software to run on the RISC/6000 platform. These were limited to minor changes to /src/regexp.c to allow correct compilation and changes to /src/Makefile and /src/config.h to properly configure the driver software to the system, and the two runtime configure files, /etc/config.PangaeaMud and /bin/PangaeaMud.info.

Further functional changes were made for a variety of reasons. Site accesses were altered, messages displayed to incoming users changed, and weather and time modelling information altered.

One specific problem with the database structure of this system, as well as object oriented databases in general, is the difficulty of establishing the set of all instances of an attribute of a given class. The MUD software operates in a way such that one can only query instances of a given object class.

Thus, in order to provide an answer to the question 'Which minerals are in the Isometric crystal system?' for instance, one can only give an answer if an active copy of each and every mineral exists in the Active Object database, and have a coded object that will run through all of the instances of the minerals one after the other, checking the "system" attribute. (This actually is workable, since the set of all minerals only runs to around 3500 instances. This code would work poorly on larger databases.)

3. Geologic Modelling Changes

A number of files and directories were created or altered for the specific purpose of making a base environment for the geologic users of PangaeaMud. In general, these files provide rooms, commands, example mineral objects and a selection of tools to extract data from the mineral objects or alter data on specific instances of the minerals.

The previous example of the user within the system will be continued here to illustrate the creation of an object and show how the system allows one to compile individual objects and update code without affecting the rest of the system. The example following assumes the user has been promoted to coding level and is logged on already, working in his user directory /u/e/erich.

```
> ls
   1 workroom.c
> cd /obj/tools
> ls
   1 bandages.c      3 memopad.c      5 shadow_spy.c   4 webster.c
   1 cod.c          15 scroll.c       1 terminal.c
> cp bandages.c /u/e/erich
Cp: /obj/tools/bandages.c copied to /u/e/erich/bandages.c
> cd /u/e/erich
> ls
   1 bandages.c     1 workroom.c
> mv bandages.c streakplate.c
Mv: /u/e/erich/bandages.c moved to /u/e/erich/streakplate.c
> ls
   1 streakplate.c  1 workroom.c
> ed streakplate.c
Editing: /u/e/erich/streakplate.c
:1,5p
// bandage.c
// This is a bandage which you can use on a player for healing.
// Mobydick@TMI-2, 10-27-92

#include <mudlib.h>
:
```

As the point of this example is not to show all the mechanics of the online line editor, the actual editing of the new file will be skipped. The code for the streakplate follows, with comments inserted, after which the example continues.

```
// streakplate.c
// A ceramic streak plate.
// written 11-12-93, Erich

// # includes in <> are in /include, otherwise they are specified
// by full pathname, such as "/u/e/erich/strk.h"
```

```
#include <mudlib.h>
// OBJECT is specified in /include/mudlib.h
inherit OBJECT;
// this function sets the various properties of the object.
void create() {
// descriptions of the object.
     set("short", "a ceramic streak plate");
     set("long", "This is a small square of unglazed porcelain."+
          "  You suspect you could test\nthe color of a mineral's"+
          " streak by typing streak <mineral>.\n");
// a list of names the object can be identified with.
     set("id", ({ "plate","streakplate","porcelain" }) );
// holdover from bandage.c, part of the economy included in the
// mudlib.
     set("value", ({ 5, "silver" }) );
// Users can only carry so much.
     set("mass", 5);
     set("bulk", 5);
     }
// this allows for specific commands to be applied to the object.
void init() {
     add_action("streak", "streak");
     }
// the meat of the code.
int streak (string name) {
// declarations of objects, strings, integers, floats, etc.
     object target;
     int strk;
// Various ways for the function to fail.
     if(!name) {
          notify_fail ("Test the streak color of what?\n");
          return 0; }
     target = present(name,this_player());
     if(!target) {
          notify_fail ("I don't see that here.\n");
          return 0; }
     if(living(target)) {
          notify_fail ("Kind of hard to do a streak test on"+
          " living things.\n");
          return 0; }
// If the object is an unknown mineral, use this code.
     if(name == "mineral") {
          return 1; }
// set strk variable from the "streak" attribute of the mineral
// being tested.
     strk = target->query("streak");
     if(!strk) {
          notify_fail ("You can't test that!\n");
          return 0; }
// What to do if you actually have a mineral being tested.
// write is the message displated to the user of the plate.
     write("You scrape the mineral across the porcelain "+
          "square.\n");
// say is what others in the virtual room see.
```

```
     say(this_player()->query("cap_name")+"tests a piece of "+
          name+" on a streak plate.\n");
// some minerals' streaks aren't listed in the encyclopedia.
     if(strk == "x" || strk == "unknown") {
          write("Hmmm... You still aren't sure.\n");
          return 1; }
     write("The "+name+" leaves a "+strk+" streak on the"+
          " plate.\n");
     return 1; }
```

Back to the example from inside the system.

```
> clone streakplate
Cloning: /u/e/erich/streakplate.c to /std/user#275.
> i

A ceramic streak plate

> l at plate
This is a small square of unglazed porcelain.  You suspect you
could test the color of a mineral's streak by typing streak
<mineral>.
> clone /obj/minerals/a/abhurite
Cloning: /obj/minerals/a/abhurite.c to /std/user#275.
> i

A ceramic streak plate
A crystal of abhurite

> streak abhurite
You scrape the mineral across the porcelain square.
The abhurite leaves a white streak on the plate.
>
```

The streak plate is one of several fairly simple tools used to discover information about a mineral. In reality, the streak color is the color of the powdered form of the mineral. On the mud, the streak color is a property set in the mineral object from data taken from several texts of mineralogic information [11], [13], [18], [26].

The level of information available about each mineral is dependent upon several factors. The first is whether or not the user is of coder level. At that level and above, the command minstat [mineral] is available. This command, found in /cmds/wiz/_minstat.c, allows virtually all of the information present about the mineral to be viewed. The command code includes some basic metaknowledge, and so only displays attributes pertinent to the instance of the mineral. As an example, minstat abhurite would produce the following.

```
+-------------------------------------------------------------+
| Name           : abhurite                                   |
| Formula        : Sn3O(OH)2Cl2                               |
| Crystal System: Trigonal                                    |
| Class          : b3m, 3m, or 32                             |
| Space Group    : Rb3m, R3m, or R32                          |
| Z              : 21                                         |
|-------------------------------------------------------------+
| LATTICE CONSTANTS                                           |
|-------------------------------------------------------------+
| a              : 10.0175                                    |
| c              : 44.014                                     |
|-------------------------------------------------------------+
| 3 STRONGEST DIFFRACTION LINES                               |
|-------------------------------------------------------------+
| 2.5313 (100), 2.8915 (70), 4.139 (50)                       |
|-------------------------------------------------------------+
| Cleavage       : none                                       |
| Fracture       : hackly                                     |
| Hardness       : 2                                          |
| Calc. Density : 4.34                                        |
| Meas. Density : 4.29                                        |
|-------------------------------------------------------------+
| OPTICS         : uniaxial                                   |
| Sign           : +                                          |
| Epsilon        : ~2.11                                      |
| Omega          : 2.06                                       |
| Twinning       : on {0001}                                  |
+-------------------------------------------------------------+
```

The minstat command knows that Trigonal class minerals only have a and c lattice constants and does not display space for b, alpha, beta, or gamma constants. Similar knowledge is inherent in the optics section, where uniaxial minerals have one set of attributes and biaxial minerals another set of properties.

Coder level users may also use the unixlike commands to go to the directory the particular mineral is stored in and scan the code for the base mineral object directly, picking up any data stored in comments in the mineral code.

The next way users may find information on the minerals is through the use of a similar command, index [mineral], which is essentially the same, but available to all users, if they are in a room which has a special flag set. With this flag set, the room acts as if a file of 3"x5" cards with information on the minerals is present. For self or outside testing, the flag can be changed so that the card file is unavailable.

The final level of gathering information on specific minerals is by utilizing the coded modelling tools on instances of the minerals. Tools already coded as examples include the streak plate, a rock hammer, a lamp, a Mohs' hardness scale, and a polarizing microscope.

The streak plate provides streak information, the hammer gives data on parting, fracture, and cleavage, the lamp gives relative opacity, the scale gives hardness, and the polarizing microscope

gives optical information such as axiality, indices of refraction and di- or pleochroism.

Due to the flexible nature of LPC, the metaclass typically contains any and all possible types of information that any of its subclasses will inherit, with the actual instances of each attribute initialized to a null value of some sort. Each class object then instantiates the attribute data slots, and each instance of the object in the active database queries the class object as to potential attributes, and maintains a set of the current state of the instantiated attributes of the unique copy of the object along with an identifier, the UID. For an example of a class object of the metaclass mineral, abhurite is presented below. Notes on the code that are not actually in the code are preceeded by a %.

```
// /obj/minerals/a/abhurite.c
// Written: 11-12-93, Erich

% Here we inherit the metaclass.
inherit "/std/mineral";

create()  {
     set("name",  "abhurite");
     set("long",  "A tiny twinned crystal. It is colorless and "+
          "transparent, with\nopalescent luster.\n");
     set("frac",          "It has fractured in hackly fashion.\n");
     set("short",         "a colorless twinned crystal");
% messages may address the object by any name in the list.
     set("id",            ({ "crystal", "abhurite" }) );
     set("habit", "Platy, hexagonal 1.5mm twinned crystals");
     set("mode",  "As blisterlike growths on tin ingots from a "+
          "shipwreck in the Red Sea.");
% The set command does not need to specify the type of the
% variable instance of the attribute, routines that call the
% attribute will specify type.  Thus we can set floating point
% integers here either inside or outside of " "'s.
     set("diff1",         "2.5313");
     set("diff2",         "2.8915");
% if so desired, this integer could also be set as "70"
     set("strdiff2",      70);
     set("diff3",         "4.139");
     set("strdiff3",      50);
     set("cleavage",      "none");
     set("fracture",      "hackly");
     set("hardness",      "2");
     set("maincolor",     "colorless");
     set("colors",        "none");
     set("streak",        "white");
     set("luster",        "opalescent");
     set("density",       "4.29");
     set("dens_calc",     "4.34");
     set("formula",       "Sn3O(OH)2Cl2");
```

```
      set("opacity",      "transparent");
      set("axial",        "uniaxial");
      set("opticsign",    "+");
      set("epsilon",      "~2.11");
      set("omega",        "2.06");
      set("spacegroup",   "Rb3m, R3m, or R32");
      set("system",       "Trigonal");
      set("class",        "b3m, 3m, or 32");
      set("twinning",     "on {0001}");
      set("forms",        "{0001} and {01b15}");
% the metaclass has many more set commands, notably "lat_b",
% "lat_alpha", and so on, but this class does not need them,
% so we ignore them.
      set("lat_a",        "10.0175");
      set("lat_c",        "44.014");
      set("unitcell",     21);
}
```

Any of the "set" variables may be accessed, including those not specified in the class instance, in which case the database will check the metaclass and provide whatever it finds there, usually a null value. An exception for the mineral metaclass is the "strdiff1" setting, which is always 100 and is set to 100 in the actual metaclass object. Additional settings may be added interactively to the instances of the object, but unless a coding object that can modify files is used, these settings will not be added to the class object code.

IV. Epilogue

PangaeaMud has come a long way since the idea first came to me, yet much remains to be done. Hopefully, the extensible nature of the LPC environment will attract users who will add their own expertise, making the system ever more powerful. Much work remains to be done to branch out the system to fully create the vision I have of PangaeaMud as a workspace and meetingplace for all kinds of geologists, not just those of a mineralogic bent.

I have learned much in the time since I began this project, though most of what I learned was about myself or about the bare bones of portability of code and hardware troubles, rather than about modelling algorithms, as I had expected. What took the most time was not the actual coding, but finding the small idiosyncrasies of the various file transfer protocols, compilers, and operating systems with which i worked.

I sat idle for two months trying to get the base software to compile, and finally only got it to work after dipping into Unix man pages, changing a program within the driver code, and trying 3 separate compilers. Getting from compiled code to enabling telnet access, even from the host machine on the root account, took another two months and the help of a gentleman in Scotland. LPC coding, on the other hand, was so straight forward I could literally code as fast as I could type.

V. Acknowledgements

I would like to thank the following people for having made this project possible through their support and encouragement.

Dr. James Kiper, for allowing me to choose this topic, keeping me moving when I got sluggish, and reading and rereading this paper as I repaired it.

Dr. Valerie Cross for giving me the original incentive to choose this topic while working on my term paper for her advanced database course and for correcting my misuse of the Queen's English.

Dr. Marcia Bjornerud, for putting up with my continual lateness throughout this work and other projects I worked on for her.

Robert Pickering and Kent Covert, for providing technical support in dealing with system problems.

The staff of The Mud Institute II, for providing technical support for the driver and mudlib software.

Mark Takacs, for providing extensive research material from his thesis and his contacts at the Human Interface Technologies Labs at Washington State University.

My parents, Lawrence and Joan Boring, for far too much to mention in a single sentence.

Dan Myers, Scott McCall, and the rest of the gang down at Banzai Research Laboratories, for calling me up all the time when I'm playing games and telling me to work on my thesis instead.

And especially Jonette Alexander, for giving me a reason to want to finish in timely fashion.

VI. References

[1] Johan Andersson (d8andjo@dtek.chalmers.se). The design & implementation of lpmud: Implications for vr technology. Technical Report R-91-4, Human Interface Technology Lab - Seattle, Wa, 1991.

[2] Johan Andersson (d8andjo@dtek.chalmers.se). Mive - multi interfaced virtual environments. Technical Report R-92-7, Human Interface Technology Lab - Seattle, Wa, 1991.

[3] Dr. Richard Bartle. Interactive multi-user computer games. Technical report, British Telcom plc, December 1990. Electronic text available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/mudreport.ps.Z.

[4] Benford et al. From rooms to cyberspace: Models of interaction in large virtual computer spaces. Interacting with Computers, 1993. (a Butterworth-Heinmann journal).

[5] Benford and Fahlen. A spatial model of interaction in large virtual environments. In Proceedings of 3rd European Conference on CSCW. milan, September 1993. (at press)

[6] Carl Brown (cbrown@netcom.com). Micromuse history. Electronic Text available via anonymous ftp from 18.43.0.102 :muse/info/Muse.History, October 1992.

[7] Amy Bruckman (asb@media lab.media.mit.edu). Identity workshop: Emergent social and psychological phenomena in text-based virtual reality. Technical report, MIT Media Laboratory, April 1992. Electronic Text available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/identity-workshop.ps.Z.

[8] Eva-Lise Carlstrom. Better living through language: The communicative implication of a text-only virtual environment, or, welcome to lambdmoo! Technical report, Grinnell College, May 1992. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/communicative.txt.Z.

[9] Pavel Curtis. Mudding: Social phenomena in text-based virtual realities. Technical report, Xerox PARC, 1992. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/DIAC92.ps.Z.

[10] Pavel Curtis and David A. Nichols. Muds grow up: Social virtual reality in the real world. Technical report, Xerox PARC, January 1993. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/MudsGrowUp.ps.Z.

[11] Edward Salisbury Dana. A Textbook of Mineralogy. 4th edition. William Ford, ed. John Wiley and Sons, Inc., November 1948.

[12] C. J. Date. An Introduction to Database Systems, 5th ed, Volume I. Addison Wesley Publishing, Reading, Mass. 1991.

[13] W. A. Deer, R. A. Howie, and J. Zussman. An Introduction to the Rock Forming Minerals. Commonwealth Printing, Ltd, Hong Kong, 1983.

[14] Randal F. Farmer and Chip Morningstar. The lessons of lucasfilm's habitat. Michael Benedikt, editor of Cyberspace: First Steps. MIT Press, Cambridge, MA, 1992.

[15] Arthur M. Glenberg and William E. Langston. Comprehension of illustrated text: Pictures help to build mental models. Journal of Memory and Language, 31:129-151, 1992.

[16] Erik A. Kay. Mire: A multi-user information retrieval environment. Technical report, Massachusetts Institute of Technology, May 1992. Electronic full-text available via anonymous ftp from lysator.liu.se :/pub/lpmud/misc/thesis.ps.Z.

[17] Kevin Kelly and Howard Rheingold. The dragon ate my homework. Wired, 1(3):68-73, July 1993.

[18] Cornelis Klein and Cornelius Hurlbut, Jr. Manual of Mineralogy. 20th ed. John Wiley & Sons, New York, 1985.

[19] George Luger and William Stubblefield. Artificial intelligence - Structures and strategies for complex problem solving, 2nd ed. Benjamin Cummings Publishing Co., 1993. p209

[20] Masinter and Ostom. Collaborative information retrieval: Gopher from moo. In Proceedings of INET, 1993.

[21] Tony Mason John R. Levine and Doug Brown. lex & yacc. O'Reilly & Associates, Inc., 2nd edition, 1992.

[22] Scott Moir. Rants & raves - muds. Wired, 1(4):17, September 1993.

[23] Gerald M. Phillips (GMP@PSUVM.bitnet). Implicit philosophy. PSYCOLOQUY, 3(30), Tue June 2 1992. Electronic full-text available via anonymous ftp from princeton.edu :/pub/harnad/psyc.92.3.30.space.2.phillips.

[24] C. C. Presson and B. R. Roepnack. Multiple mental models. PSYCOLOQUY, 3(65), Wed December 16 1992. Electronic full-text available via anonymous ftp from princeton.edu :/pub/harnad/psyc.92.3.65.space.12.presson.

[25] Elizabeth M. Reid (emr@munan.ee.mu.oz.au). Electropolis:

Communication and community on internet relay chat. Technical report, University of Melbourne, 1991. Honor's thesis for Dept of History. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/electropolis.ps.Z.

[26] Roberts, Campbell, & Rapp, The Encyclopaedia of Mineralogy, 2nd. Edition

[27] Michael S. Rosenberg (msr@casbah.acns.nwu.edu). Virtual reality: Reflections of life, dreams, and technology - an ethnography of a computer society. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/ethnography.txt.Z, March 1992.

[28] Jill Serpentelli. Conversational structure and personality correlates of electronic communication. Technical report, Haverford College, 1992. Electronic document available via anonymous ftp from beta.xerox.com :/pub/MOO/papers/conv-structure.txt.Z.

[29] Ben Shneiderman. Designing the User Interface: Strategies for Effective Human- Computer Interaction. Addison-Wesley Publishing Company, 2nd edition, 1992.

[30] Jennifer Smith. Frequently asked questions 1/3: Muds and mudding. Electronic text distributed via USENET group rec.games.mud.announce, July 1993.

[31] Jennifer Smith. Frequently asked questions 2/3: Mud clients and servers. Electronic text distributed via USENET group rec.games.mud.announce, July 1993.

[32] Mark Takacs (tak@hitl.washington.edu). Muds as groupware. Class paper, December 1991.

[33] TMI-2 Internal documentation for driver, mudlib, 1993.

[34] UNIX development team. Unix man pages, 1991.

VII. Appendices

A. Code Segments Changed

This is a list of code segments/programs I had to write or change from the base code. * indicates minor changes, ** indicates major changes, and no marker indicates I wrote at least 90% of the segment. The list is ordered by directory. All listings may be presumed to be preceeded by /usr/users/boring/MudOS_0.9.18/, the basic driver directory, and all the changes below the mudlib change line are preceeded by /usr/users/boring/MudOS_0.9.18/tmi2lib_1.1/.

driver changes

(src is the precompiled driver code)
src/Makefile *
src/config.h *
src/regexp.c *
(These two contain runtime configuration data.)
etc/config.PangaeaMud **
bin/PangaeaMud.info **

mudlib changes

(adm/etc is used by various mudwide daemons.)
adm/etc/access.allow *
adm/etc/approved_sites *
adm/etc/banishes.o **
adm/etc/dsrtweat.data
adm/etc/groups *
adm/etc/weather.data *
adm/etc/wintweat.data
adm/etc/yeardates.data *
adm/news/faq **
adm/news/help
adm/news/news
adm/news/nplayer_intro
adm/news/nplayer_news
adm/news/welcome
cmds/wiz/_minstat.c
cmds/std/_index.c
(d/Banzai/ is the rooms directory for most of my rooms.)
d/Banzai/archbull.c
d/Banzai/archives.c
d/Banzai/archlibs.c
d/Banzai/comproom1.c
d/Banzai/confroom.c *
d/Banzai/foyer.c
d/Banzai/labfoyer.c
d/Banzai/lab1.c
d/Banzai/mainboard.c *
d/Banzai/path.c
d/Banzai/postoffc.c *

```
d/Banzai/upstairs.c
d/Fooland/ * (Minor changes to all files in the directory.)
d/grid/rooms/ * (Minor changes to all 11 files in dir.)
d/Oxford/ * (Minor changes to all files.)
data/adm/daemons/emoted/Root.o *
doc/wizhelp/new_wiz *
include/body.h *
include/config.h *
include/weather_d.h *
obj/kendall.c
(obj/gtools is the directory for geologic data extraction
     tools.)
obj/gtools/diction.c
obj/gtools/hammer.c
obj/gtools/hardtest.c
obj/gtools/lamp.c
obj/gtools/microscope.c  (unfinished)
obj/gtools/streakplate.c
(obj/minerals and subdirectories are all the minerals
     directories, only the first is explicitly listed.)
obj/minerals/a/abelsonite.c
obj/minerals/a/abernathyite.c
obj/minerals/a/abhurite.c
obj/minerals/a/acanthite.c
obj/minerals/a/acetamide.c
obj/minerals/a/achavalite.c
obj/minerals/a/actinolite.c
obj/minerals/a/adamite.c
obj/minerals/a/adelite.c
obj/minerals/a/admontite.c
obj/minerals/a/aegirine.c
obj/minerals/a/aenigmatite.c
obj/minerals/a/aerugite.c
obj/minerals/a/aeschynite.c
obj/minerals/a/aeschynite-nd.c
obj/minerals/a/aeschynite-y.c
obj/minerals/a/afghanite.c
obj/minerals/a/afwillite.c
obj/minerals/a/agardite-la.c
obj/minerals/a/agardite-y.c
obj/minerals/a/agrellite.c
obj/minerals/a/agrinierite.c
obj/minerals/a/aguilarite.c
obj/minerals/a/ahlfeldite.c
obj/minerals/a/aikinite.c
obj/minerals/a/ajoite.c
obj/minerals/a/akaganeite.c
obj/minerals/a/akatoreite.c
obj/minerals/a/akdalaite.c
obj/minerals/a/akermanite.c
obj/minerals/a/akrochordite.c
obj/minerals/a/aksaite.c
obj/minerals/a/aktashite.c
obj/minerals/a/alabandite.c
```

```
obj/minerals/a/alamosite.c
obj/minerals/a/aldermanite.c
obj/minerals/a/aluminum.c
obj/minerals/a/antimony.c
obj/minerals/a/arsenic.c
obj/minerals/base.c
obj/minerals/min/  (all) (More minerals)
(std is where metaclasses SHOULD go, though not all have.)
std/food.c
std/mineral.c
(text is a directory for all plaintext documents, there is no
     actual "code" in here.)
text/avail
text/classes
text/mohs
text/scope1
text/defn/ (all) (Individual definitions for specific geologic
     terms.)
text/ext/indx.dic
text/ext/titlepg.dic
u/e/erich/workroom.c **
```

B. Real-time Communication Commands

echo <msg> - Everyone in the same virtual location as the sender receives the exact text of the message and nothing more.

> echo A hammer appears.
(Everyone in room sees:)
A hammer appears.

echoall <msg> - Everyone on the mud receives exactly the text of the message and nothing more.

> echoall A landslide occurs.
(Everyone logged in sees:)
A landslide occurs.

echoto <name> <msg> - The person named receives the exact text of the message and nothing more. The recipient may be anywhere in the mud.

> echoto takacs Your takeout order is here.
(Takacs sees:)
Your takeout order is here.

emote <msg> - Everyone in the room receives the sender's name with the message of the text appended.

(erich types:)
> emote studies the data.
(Everyone in room sees:)
Erich studies the data.

emoteto <name> <msg> - The person named receives the sender's name with the text of the message appended. The recipient may be anywhere on the mud.

(erich types:)
> emoteto takacs would like to speak with you.
(Takacs sees:)
Erich would like to speak with you.

say <msg> - The equivalent of real world speech, all of those in the same virtual room will receive the sender's name and the word says with the text of the message appended.

(erich types:)
> say is anyone here a mineralogist?
(Everyone in room sees:)
Erich says 'is anyone here a mineralogist?'

shout <msg> - A broadcast message, everyone within the mud will receive the sender's name and the word shouts with the message

appended except those who are specifically muffled to avoid shouts.

(erich types:)
> shout Can someone give me a hand here?
(Everyone not muffled sees:)
(C) Erich shouts 'Can someone give me a hand here?'

tell <name> <msg> - Rather like telepathy of popular fiction, the message is received only by the person of the name used, no matter where sender and recipient icons are located.

(erich types:)
> tell takacs Hey there!
(Takacs sees:)
Erich tells you: Hey there!

whisper <name> <msg> - Similar to tell, but the recipient must be in the same virtual location as the sender.

(erich types:)
> whisper takacs excuse me, but...
(Takacs sees (If he is in same room):)
Erich whispers to you 'excuse me, but...'

Communication may also be achieved by means of channels to which specific users are attuned, depending on their interests. Users may sign up for any number of these channels, which act as semidirectional shouts. All other users tuned to the specific channel will receive the message spoken over the channel.

(erich types:)
eiz anyone out there?
(Anyone attuned to the eiz channel sees:)
Erich eizzes: anyone out there?

(erich types:)
gwiz hello?
(This message is received by anyone tuned to this channel on any mud attached to the UDP mud network. To transmit the same message to anyone on the TCP network, one would type interwiz hello?)
Erich@pangaeamud gwizzes: hello?
(for TCP substitute interwizzes for gwizzes)