Measuring the Effectiveness of ProgrammingLand

Curt Hill
Department of Mathematics
Valley City State University
Valley City, ND, USA
701 845-7103
Curt.Hill@vcsu.edu

Brian M. Slator
Department of Computer Science
North Dakota State University
Fargo, ND, USA
slator@cs.ndsu.edu

Vijayakumar Shanmugasundaram
Concordia College
Moorhead, MN, USA
218 299-3343
shanmuga@cord.edu

ABSTRACT

ProgrammingLand, as a resource for Computer Science instruction, continues to evolve and improve. ProgrammingLand delivers content, monitors student activities, contains a variety of educational exercises and may make assignments. Certain facilities of Learning Management Systems (LMS) are now incorporated into the system.

The existing and new features were used in an experiment with two sections of the same course. One of these used a standard textbook and LMS, while the other one used ProgrammingLand, without either a textbook or another LMS. The results of this experiment favor continued development of the system.

KEY WORDS

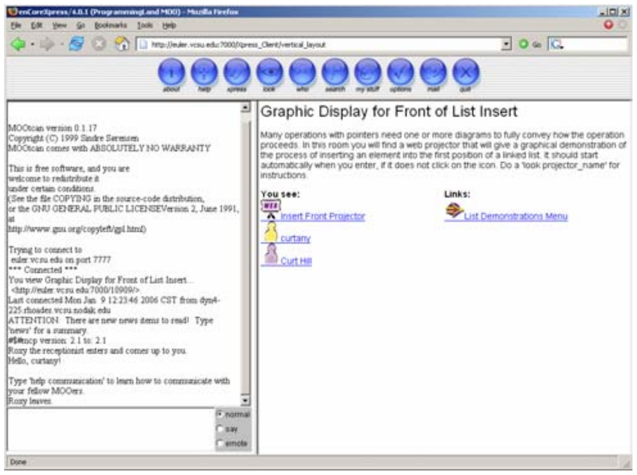
CS educational research, online instruction, blended education, distance education.

1. Introduction

ProgrammingLand is an online computer science educational resource [4]. It delivers educational content material, monitors student progress, facilitates educational exercises and delivers offline assignments. It is based upon a MOO [3], but uses a web interface. It has been in use and development for several years as a substitute for a conventional textbook. It was recently enhanced with several features to manage a course more effectively. ProgrammingLand is one of a series of projects by the World Wide Web Instructional Committee [8,9].

The paradigm of ProgrammingLand is that of a museum, where the students browse exhibits in a learner-centered fashion. An exhibit is a room in MOO terminology and the passages between these are MOO exits. A typical MOO client would have a person type the exit name to move to a destination. However, in the web client [6], moving is accomplished by clicking the destination exhibit's name as shown in one of the client's panes. Figure 1 shows the web client and a typical exhibit. The content of the exhibit is shown in the upper part of the right hand pane of the display of Figure 1. This exhibit has a short amount of text in the top part of the right hand pane. Also in this pane is the icon of an interactive object, and the icons of two people logged into the MOO and in this room. This exhibit has a single exit which is on the right hand side of that pane.

Superimposed on the room and exit structure of a MOO is a lesson structure peculiar to ProgrammingLand. Several rooms with related content may be entered through a lesson room. ProgrammingLand records several kinds of activities of a student which indicate the student's progress. The lesson uses this data and the requirements that the instructor has established to satisfy the lesson. These requirements may include visits to certain exhibits, prerequisites or subordinate lesson completion or the use of several types of interactive objects. The first visit of a student to a lesson room displays the requirements. At any later time, the student may display the requirements. This display also shows which of the requirements have been completed by this student. Upon the completion of a lesson, that event is recorded, along with the rest of the student's accomplishments.


enCoreXpress/4.0.1 (ProgrammingLand MOO) - Mozilla Firefox
File
Edit
View
Go
Bookmarks
Tools
Help
http://euler.vcsu.edu:7000/Xpress_Client/vertical_layout
Go
about
help
xpress
look
who
search
my stuff
options
mail
quit
normal
say
emote
Done

```
MOOtcan version 0.1.17
Copyright (C) 1999 Sindre Sørensen
MOOtcan comes with ABSOLUTELY NO WARRANTY

This is free software, and you are
welcome to redistribute it
under certain conditions.
(See the file COPYING in the source-code distribution,
or the GNU GENERAL PUBLIC LICENSEVersion 2, June 1991,
at
http://www.gnu.org/copyleft/gpl.html)

Trying to connect to
 euler.vcsu.edu on port 7777
*** Connected ***
You view Graphic Display for Front of List Insert...
 <http://euler.vcsu.edu:7000/10909/>
Last connected Mon Jan  9 12:23:46 2006 CST from dyn4-
225.rhoades.vcsu.nodak.edu
ATTENTION:  There are new news items to read!  Type
'news' for a summary.
#$#mcp version: 2.1 to: 2.1
Roxy the receptionist enters and comes up to you.
Hello, curtany!

Type 'help communication' to learn how to communicate with
your fellow MOOers.
Roxy leaves.
```

Graphic Display for Front of List Insert

Many operations with pointers need one or more diagrams to fully convey how the operation proceeds. In this room you will find a web projector that will give a graphical demonstration of the process of inserting an element into the first position of a linked list. It should start automatically when you enter, if it does not click on the icon. Do a 'look projector_name' for instructions.

You see:
Insert Front Projector
curtany
Curt Hill

Links:
List Demonstrations Menu

Figure 1. A typical exhibit.

The instructor creates a course object that contains references to a variety of other objects that the course needs. The course refers to the students in the course, the required lessons of the course, and a gradebook. The course does not need to refer to all the lesson rooms that are needed; typically only the higher level ones. Any lesson may require the completion of a subordinate or pre-requisite lesson and the course only needs to refer to the first of these, not those required by the refererenced lesson. Since the student is a member of the course, this makes available a status command indicating the required lessons. and which ones have been completed by this student.

The typical, but not required, use of ProgrammingLand has each of the lessons listed in the course having an assignment. When a student completes the lesson, an agent visits the student and delivers the assignment. This assignment is often a programming assignment to be done outside of ProgrammingLand, but it could be nearly anything. Assignments have also included completion of worksheets, interaction with a simulator, and writing exercises. The agent delivers the assignment; at any later time, the student may use a command to see it again.

The agent that delivers an assignment for a particular lesson also may contain more than one assignment. These assignments need to be equivalent by requiring approximately the same amount of work and exercising the same skills. The assignments are given out in a round-robin fashion as students complete the lesson. This has shown itself to be an effective means of minimizing student cheating on assignments, although making equivalent but different assignments is somewhat challenging as they become more difficult.

The lesson and assignment mechanism motivates students to do the actual reading required by the instructor. When used in this way, a student cannot receive the assignment until he or she has done a certain amount of work in ProgrammingLand. It is much more problematic to determine if a student has actually performed an assigned reading in the textbook.

One of the issues with either ProgrammingLand or an LMS is the time it takes a student to learn how to use the system. This was exacerbated in ProgrammingLand by its lack of several facilities common in an LMS. In that case a student needs to be able to master ProgrammingLand to utilize its unique capabilities *and* an LMS to obtain more mundane services such as accessing current grades and taking tests and surveys. This increased the non-course related learning problem for students. Thus it was determined that these features could and should be incorporated into ProgrammingLand, so that the students would only have one system to learn. The new, innovative, and unique element of the system is that it combines the functionality of an LMS with the functionality of a multi-user immersive virtual environment (IVE), where students are doing their work in a vast, exploratory, and engaging virtual reality.

In the summer of 2005 two completely new subsystems were added to ProgrammingLand: a gradebook and a testing facility. These provide the needed LMS facilities and are described next.

2. New Facilities

ProgrammingLand is much different from a regular LMS such as Blackboard [2], WebCT [2] or Angel [1]. In the latter, the system emulates a web site. There is typically a login followed by a series of active pages. Whether these are standard web pages or a variety of pages dynamically generated for this visit is irrelevant to the student. Generally the only interface needed is the web point and click approach. There was no intent to make ProgrammingLand like these.

2.1 Existing facilities

ProgrammingLand uses a different interface paradigm, since it is based on both MOO and web technology. Navigation mainly uses the point and click format, since translating an exit into a hot link is relatively easy. The client also converts several common MOO commands into buttons at the top of the form. One of these gives access to the internal email system of a MOO. Another allows editing of various features of the client or student object. Others display helpful information, such as the people currently logged into the MOO. However, the left two panes of the form shown in Figure 1 are used for typing in MOO commands and viewing the results. All of the capabilities mentioned previously, such as mail, are available via command. Moreover, a variety of educational objects require this type of interface.

A code machine is an object that attempts to demonstrate a short piece of code. It is an object with three main methods. One will show the code with line numbers, the second will explain the purpose and form of each line and the third will trace through the code simulating an execution. All three of these are commonly used by students and often an explanation or trace of a code machine may be part of the requirements of a lesson.

The display of an exhibit will show any objects in the room. These objects may include other students (everything is an object in a MOO) or any of the interactive objects. If the object has one main method that a student should use, then clicking on the object's icon will execute the method. The slide show is one of these types of objects; clicking on it will show a series of web pages in succession. This behavior may also be started by typing a command. In contrast to this is the code machine, which has no good default method to execute via a click.

Modification of the behavior of any MOO, including ProgrammingLand is relatively easy, compared to an LMS. In these, it is easy to change some things and impossible for others. It is easy to modify or add new content, while changing the behavior of the grade book would require the vendor to make expensive modifications. In contrast to this, creating new objects and their methods is done with a simple MOO command. The methods, be they private or accessible to all, are programmed with a C-like script. The authors and their students have created usable objects and their methods.

2.2 Gradebook

The gradebook is a new object that performs the expected functions. It allows an instructor to add or change student scores, allows the system to enter scores that can be generated by the system, and gives a student access to his or her score only.

A gradebook object is created and then attached to the course object, although a course does not require a gradebook object. Students added to or removed from the course are automatically processed for the gradebook.

An instructor needs to create scoring opportunities for the gradebook. Each such opportunity must have a name and a maximum number of points. The points are integer valued. Once the scoring opportunity has been created the score may be entered manually by the instructor or automatically entered by some component of the system. At any later time, the instructor may change the score, even those automatically generated.

Currently there are two situations where a score may be generated by other system objects, although it is not hard to add others. The first is when a student takes a test and the second when the student completes a lesson that has a roving goalie agent. The multiple choice part is fully graded by the system and the results entered into the gradebook. Other parts of a test would be graded by an instructor or assistant and entered manually. The test system is discussed later.

A roving goalie agent is summoned when a student completes select lessons. This agent will usually deliver an assignment to the student that needs to be completed outside of ProgrammingLand. This is typically a programming assignment, but could be a worksheet, writing exercise or nearly any other type of assignment. In addition to this assignment, the agent may give a score to the gradebook. This score may have one value if the lesson is completed on or before a preset date, and lesser values depending on how late the lesson is completed. In some cases, the agent's only purpose is to congratulate the student and add points to the gradebook.

The instructor also requires displays of the current values; these are provided via command. In addition, the values may be exported in a form suitable for easy import into a spreadsheet. Generally, the gradebook should be sufficient for the normal administration of a course, but moving the data to a spreadsheet is handy for analyses, such as is presented in a later section of this paper.

The student also needs to access the gradebook. This is done with a command that shows each scoring opportunity and the score received for it. In addition it computes the student's current letter grade, based on one of several algorithms chosen by the instructor. The student is generally unaware of the object structure of ProgrammingLand and thus has no real access to the course or gradebook objects. This, among other things, prevents students from violating the privacy of other students or illegally affecting the functioning of the system. In general, MOOs classify users according to their privileges; students are the least capable users.

2.3 Tests and Surveys

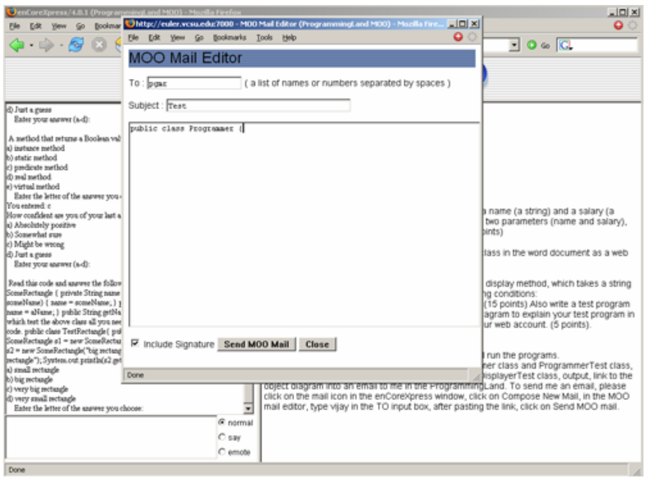
The system has had a quiz facility for some time. This was an option for students who may have knowledge of the subject matter prior to seeing it in ProgrammingLand. A student who leaves a lesson without completing the requirements was given the option to take a quiz to prove mastery of the material. If the student passed the quiz, he or she would receive credit as if the entire requirements had been satisfied. After the student left the lesson with unfulfilled requirements, an agent would visit and ask if the student wanted to take the quiz. An affirmative answer would transport the student to a quiz room that was inaccessible in any other way. The quiz would then be administered in this room. A quiz could only be taken if certain conditions were met. The student could not have taken this quiz more than once, nor could he or she be missing any requirements of the lesson other than merely visiting exhibits.

Each quiz has five questions with five multiple choice answers. The particular quiz given to any student was randomly generated, based upon exhibits that this student had not visited. Each room should have quiz questions with one or more right answers and four or more wrong answers. These questions were not visible to any student visiting the room. The quiz was generated by reducing the pool of questions to five and choosing one right answer and four wrong and then randomly arranging them. The student would be presented the question and the five potential answers. He or she would respond with the letter of the answer and be told whether it was correct or not. Answering four correctly passed the quiz and satisfied the lesson.

This form of quiz was only an option to satisfy the requirements of a lesson. Although information is recorded about the quiz event, it is recorded on the student object, not in the gradebook. Thus has no effect on the student's grade. However, it did demonstrate proof of concept when tests and surveys before tests and surveys were considered.

The students' perspective of a test is rather simple. It is an object, usually placed in the room that they start in whenever they login to the system. The icon of that object may be clicked on to start the test, when a test is available. Tests are available only at certain times; starting a test outside these times gives a message stating the unavailability of a test at that time. Although there is just one object and corresponding icon, all the tests of the semester are taken by interacting with that object. The test is given in both the right and left-hand panes of the client window. A multiple choice test is given in the left-hand pane and the instructor graded portion is given in the right-hand pane, using the mail feature to send the instructor the answers. If the student leaves the room that has the test or takes more time than is allowed, the test is terminated. A particular test may only be taken once; in the case of system failure, the instructor may reset the test.

Figure 2 shows a test in the process of being taken, with two windows and several panes. There are two overlapping windows. The smaller window in the center is the MOO Mail editor; in this a student is filling in the answer for a question. Behind that window is the main client window. On the left-hand side are the multiple choice questions, which are answered in the lower left-hand pane. In the right-hand pane are the questions to be answered in the mail window. Of course, either window may be brought to the foreground.


enCoreXpress/4.0.1 (ProgrammingLand MOO) - Mozilla Firefox
http://euler.vcsu.edu:7000 - MOO Mail Editor (ProgrammingLand MOO) - Mozilla Fire...
File Edit View Go Bookmarks Tools Help
MOO Mail Editor
Include Signature
Send MOO Mail
Close
Done
normal
say
emote

To : pgmr (a list of names or numbers separated by spaces)

Subject : Test

```
public class Programmer {
```

d) Just a guess
Enter your answer (a-d):

A method that returns a Boolean val
a) instance method
b) static method
c) predicate method
d) real method
e) virtual method
Enter the letter of the answer you
You entered: c
How confident are you of your last a
a) Absolutely positive
b) Somewhat sure
c) Might be wrong
d) Just a guess
Enter your answer (a-d):

Read this code and answer the follow
SomeRectangle { private String name
someName) { name = someName; } p
name = aName; } public String getNa
which test the above class all you nee
code. public class TestRectangle{ pu
SomeRectangle s1 = new SomeRectan
s2 = new SomeRectangle("big rectang
rectangle"); System.out.println(s2.get
a) small rectangle
b) big rectangle
c) very big rectangle
d) very small rectangle
Enter the letter of the answer you choose:

name (a string) and a salary (a
two parameters (name and salary),
oints)

lass in the word document as a web

display method, which takes a string
ng conditions:
(15 points) Also write a test program
agram to explain your test program in
ur web account. (5 points).

run the programs.
er class and ProgrammerTest class,
isplayerTest class, output, link to the
object diagram into an email to me in the ProgrammingLand. To send me an email, please click on the mail icon in the enCoreXpress window, click on Compose New Mail, in the MOO mail editor, type vijay in the TO input box, after pasting the link, click on Send MOO mail.

Figure 2. A screen during a test.

Not surprisingly, the test system is much more complicated from the system's view. There are four interacting object types that make up the test in order to maximize the ease of use and flexibility of the tests. These include the testbank, which is as the name suggests: a bank of questions on one broad or narrow topic. These questions are multiple choice only. Those questions that require more complicated answers than multiple choice are stored in a testform object. The testfront object is what the student actually sees and interacts with. Between the former two and latter one is the testgen object. This one selects questions from one or more testbanks, uses one testform and displays the resulting test using the testfront. Each of these will be discussed next.

The testbank object is intended to be persistent from term to term, containing questions that are reusable. The questions are in the same form as that of the quiz, with one or more right answers and four or more wrong answers. The system will randomly choose questions, and right and wrong answers, as well as the order of the answers. This disallows answers like “None of these” or “Both a and c,” but reduces the effectiveness of student collaboration when the test is not given in a class period.

The testform object is typically generated new for each test, although the object itself may be used over and over again. It contains questions in HTML format, so links to items outside of the MOO are acceptable. The answers to these questions are placed in email from within ProgrammingLand. This was the easiest solution to saving the data generated by the students. In the future it will be re-evaluated to determine if there is a better mechanism for this.

Instructors use a simple process to generate a test. The object has a prepare method which guides them through the process. Each testbank that is to be used is found and then the number of questions that are required from that testbank. The testform is also chosen, although a test does not require the presence of a testform. The testfront does not need to be mentioned, since it finds the course object; the testgen is also attached to this object. Supplying the grading opportunity, name, and the minutes a student may take on the test complete the process.

The testgen object has two additional methods that limit the window of opportunity to take the test. The test is normally closed; thus the instructor must open the test to make it available and close it when no further tests may be taken.

Surveys are somewhat simpler than tests. Like a test, a student may take a survey by clicking on a survey icon, but only take a survey once. Unlike a test, questions are never randomized and the student always answers all of them. The results of any survey question is a single line, often a multiple choice letter.

There are two objects in the survey system: a surveyfront object and a surveybank object. The surveyfront is the object which is available to the student, similar to a testfront. The surveybank has the capabilities of both the testbank and testgen. Survey answers are stored in the gradebook like the test, but in a different area. The student will not see his or her own survey results when the grade is accessed and but the instructor can view or export the survey.

3. The Experiment

The authors continue to conduct experiments to determine the effectiveness of ProgrammingLand as an educational tool. The most recent of these was the comparison of two sections of the same course. The two sections were taught by one of the authors in the same semester, using the same overall approach, the same tests and same assignments. One of these sections used an LMS and a conventional textbook as the supplementary content source. The other used ProgrammingLand as both the LMS and supplementary content source. The programming language in both courses was Java. The goal of the experiment was to minimize the differences between the two sections, other than the use of LMS and supplementary content source.

Despite the attempt to design our experiment in a way to maximize the differences between the features we wished to investigate, there were some problems. The first of these involved student populations. The two sections were both rather small for this type of comparison and not as similar in makeup as we had hoped. One section met at 8:30 AM and contained 11 students. It used the LMS from Angel Learning[1] and a popular textbook [7]. The second section met at 11:50 AM and contained 18 students. However, the character of the two sections was not as similar as could be desired.

This class has two types of students. The class usually contains computer science majors and minors, but a fair number of a non-technical majors as well. This course also satisfies a mathematics and science general education requirement at this institution, which is the main source of the non-technical majors. The bulk of these other majors were in the 11:50 section, which somewhat distorted the results.

The initial comparison found that the class average of the early section using a common LMS was substantially better at 81% than that of the ProgrammingLand section at 72%. However, when we categorized averages by the declared major, a different picture emerged. These results are summarized in Table 1.

Table 1. Course results.

	Angel percent (count)	PLand percent (count)
CS Major	80% (7)	86% (4)
CS Minor	84% (2)	87% (1)
Other	82% (2)	68% (13)

Even Table 1 does not give a completely fair picture. Two of the non-technical majors abandoned the course without completing it; both of these were in the ProgrammingLand section. Removing these two gave a somewhat more balanced view, summarized in Table 2.

Anyone who is familiar with the realities of testing and experimentation in an educational setting will recognize this situation. Students are highly variable in terms of their life circumstances, their motivation towards their studies, and their interest in completing their assignments. In addition, data sets are typically small, with populations most often being below 30 subjects. This is why educational data is typically "messy" with many outliers in the data sets, and relatively high statistical variance. Experienced researchers learn how to operate within the constraints of the field of study. Inexperienced individuals quail at these realities.

Table 2. Course results without absentees.

	Angel percent (count)	PLand percent (count)
CS Major	80% (7)	86% (4)
CS Minor	84% (2)	87% (1)
Other	82% (2)	80% (11)

There were no exit surveys, so there is no way to determine if the students who did not complete were influenced by issues with ProgrammingLand. The informal response of the ProgrammingLand students to the instructor at the beginning of the class was not completely positive, but very favorable by the end of the semester.

4. Conclusions And Future Work

The ProgrammingLand section did not have a textbook, so it was the main source of content material to supplement the lectures, yet the section does not seem to have been disadvantaged. Furthermore, this section did not take full advantage of the lesson structure of ProgrammingLand. The students did not receive assignments or other types of scoring opportunities from the system. This deprived ProgrammingLand of one of its principle advantages but opened the possibilities for new research.

The next phase of this experiment is to have the completion of a lesson award points to the student in lieu of some other type of an assignment. In most respects, this keeps the section similar to the two studied previously, but gives the students better motivation to use the system. This phase of the experiment was proceeding at the time of the writing of this paper.

This study implies no criticism of the textbook used. This is a fine book [7], honed by the competition of the marketplace. Textbooks that survive to a fourth edition are generally widely regarded. It has good descriptions and far more prose than the sum of the ProgrammingLand Java wing. However, just as a novel and the movie made from it each have different strengths, the use of a textbook and ProgrammingLand each bring different virtues to a course.

ProgrammingLand is covered by the Gnu GPL. The project web site is available [5]. Inquiries should be directed to the first author.

5. Acknowledgments

We extend our thanks the students who have used, tested and contributed to this project. ProgrammingLand has been supported by ND-EPSCoR through the FLARE program under EPS-9874802 and is currently supported by National Science Foundation Grant EIA-0313154. The authors gratefully thank and acknowledge both of these organizations.

6. References

[1] Angel Learning, http://www.angellearning.com/products/lms/default.html. Date accessed 9 January 2006.

[2] Blackboard, http://www.blackboard.com/us/index.aspx. Date accessed 9 January 2006.

[3] Curtis, Pavel, Mudding: Social Phenomena in Text-Based Virtual Realities. *Proceedings of the Conference on Directions and Implications of Advanced Computing* (sponsored by Computer Professionals for Social Responsibility)

[4] Hill, Curt, Brian M. Slator, Vijayakumar Shanmugasundaram, and Lisa M. Daniels An Online Computer Science Instructional Resource. In *Proceedings of the International Conference on Web Based Education (WBE 2006)* (Puerto Vallarta, Mexico, January 23-25, 2006). ACM Press, New York, NY, 2000, 526-531.

[5] Hill, Curt, http://euler.vcsu.edu/pland.htm. Date accessed 4 January 2006.

[6] Holmevik, Jan Rune, and Cynthia Haynes. encore, Open Source MOO project. http://lingua.utdallas.edu/encore/index.html Date accessed 9 January 2006.

[7] Horstmann, Cay. *Java Concepts, 4th ed.* John Wiley, New York, NY, 2006.

[8] Saini-Eidukat, B., Schwert, D.P., Slator, B., Daniels, L., and Terpstra, J., 2005, Research on authentic assessment using a virtual world for learning geology. Geological Society of America North-Central Section, 39th Annual Meeting, 19-20 May 2005, Minneapolis, MN.

[9] WWWIC (2005). World Wide Web Instructional Committee. http://wwwic.ndsu.edu/. Date accessed 9 January 2006.