Developing a Text-Based MMORPG to Motivate Students in CS1

Richard Barnes and **Maria Gini**
Department of Computer Science and Engineering
University of Minnesota
4-192 EE/CSci, 200 Union St. SE, Minneapolis, MN 55455
{barnes,gini}@cs.umn.edu

Abstract

We present the outline of a class project in which entry-level students in our CS1 course spent a full day developing a text-based massively multi-player online role-playing game (MMORPG) using Scheme. We describe briefly our CS1 course, the specifics of the game we asked students to implement, and the project organization. Comments from the students about their experience are also presented. Most students felt that the project was a beneficial learning experience. The project was organized as part of a larger multi-year effort to increase student learning and student participation. Class performance shows that more students have completed the course and have obtained higher grades than in the past, providing support to the educational value of this project and the other active learning opportunities we have used during the semester.

Project Rationale

Games and AI are often used to attract students to computer science (see, for instance, (Rao & Mitra 2008; Kumar *et al.* 2008). It is common wisdom that students gain confidence by hands-on manipulation, and by seeing concrete effects of their work. Hands-on programming experience increases self-confidence (Beyer *et al.* 2003) and improves students' self-perception about computer science and career goals.

The participants to the project we describe here were all entry-level students taking their first class in the computer science program. At this point, the curriculum tends to be somewhat discouraging because there is a discord between the projects the students envision themselves doing and what they are capable of. This project is meant to help bridge this gap.

The project was scheduled towards the end of the semester and was designed to include many of the constructs and examples students learned and worked on in their weekly labs. In this way, the project was meant to put to practical use what had previously been merely exercises.

Finally, the project was designed to provide fewer step-by-step directions than the labs to give the students the liberty to conceive and develop their own ideas using what they'd been taught rather than simply demonstrating their knowledge of the course material. Group cooperation was an essential part of the project, since students were divided into groups, each group dealing with a different aspect of the project. Cooperative learning is known to enhance learning (Johnson & Johnson 1988).

Our CS1 course

Our CS1 course is modeled after the MIT 6.001 entry-level computing course. Our students are expected to take this course in their freshman year, but some take it in the sophomore year. The course is offered with a single section per semester. The students are divided into smaller groups (30-34 students each) for a 2 hours/week lab, where they work on programming assignments in pairs with the help of the TAs. The course enrolls 100-150 students per term.

The course offers an introduction to the fundamental principles of programming and to different programming paradigms, with emphasis on the design of abstract data types and recursive algorithms. The course teaches students how to think as computer scientists, by teaching the process of decomposing problems into simpler problems, and of controlling program complexity by using abstractions that hide implementation details.

The course does not assume any prior programming knowledge. Students come to the class with very different backgrounds. Some have never programmed, some have significant programming experience, but they have rarely gone beyond the mechanical understanding of how to write a program to reach an understanding of how to organize their thinking process.

The language taught in the course, Scheme, is concise, has clean and consistent semantics, and it is perfect to teach the students how to think. An additional reason for the choice of Scheme it that it is a language unknown to most of the students taking the course. This provides a more even starting point for all the students in the course. However, the students are often reluctant to put a significant effort into understanding the material, since they do not see how knowledge of Scheme will increase their short term marketability.

In 2005 we were selected as one of the twelve teams at the University of Minnesota which participated in a campus-wide three-year long project supported by the Bush Foundation on "Promoting Student Learning in Large Classes." In the first year of the project we assessed the current situation and started discussing what types of interventions were appropriate for the course. We wanted to maintain the rigor of the course and its contents, but find ways of engaging the

students more in the learning process (Smith *et al.* 2005).

In the second year we experimented with a variety of active learning techniques, ranging from different small group activities in class, to doing a lab using the Sony robot dogs AIBO (Chilton & Gini 2007), to placing a strong emphasis on problem-based learning (Prince 2004) and other forms of active-learning.

In the last year, we continued experimenting with what worked well the previous year and added the project we are presenting here. We assessed overall student performance in the three semesters and noticed a clear improvement, as we will describe later.

MMORPG Synopsis

MMORPGs (massively multi-player online role-playing games) are probably best exemplified by *World of Warcraft* and also, to an extent, *Second Life*. Although they are best known today as being graphical, they date back to the 70s in their text-based form, known as MUDs, MUCKS, or, more generally, MU*s. They are akin in nature to the board game Dungeons and Dragons. The MMORPG offers a persistent, on-going world to which players connect and in which they develop a character through interactions (trading, chatting, fighting, etcetera) with other characters and NPCs (non-player characters), as well as with the environment of the world. The environment stems from the MMORPG's theme - generally medieval fantasy. The MMORPG, then, is a mirror of our own world idealized for the purpose of game-play.

Multiple studies have been conducted to understand the motivations of players of online games (see, for instance, (Yee 2006)). Most often MMORPG are used in the classroom as a tool to support learning and build online communities (see, for instance, (Steinkuehler 2004; Hughes & Scott 2005; Childress & Braswell 2006)), not as a way to learn programming. In (Wadley & Sobell 2007) a study is reported in which students were asked to build database tables and procedures for a MMORPG as way of learning about client-server architectures and relational databases. The course was a final year undergraduate course. The course in which we built the MMORPG we report in this paper was a freshman course.

Games are also often used in Artificial Intelligence courses, to build AI into games. Our course did not include any material on Artificial Intelligence, and the part of the game the students developed did not have any AI in it. We did this project as an experiment to engage students in more complex programming and to prepare then for future AI work. The fact we chose a game appealed to many students who were intrigued by the idea they could build an entire game.

What an MMORPG Looks Like

Connections to an MMORPG are made with a telnet client. Upon connecting, a player is prompted for a name and password after which they wake up in the games world. This may appear as follows:

> You are presently standing in the common room of a tavern. In one corner, a large fire blazes in a stone hearth. Scarred oak tables are scattered throughout the room. The atmosphere is hazy with tobacco smoke.
> Exits: {E}xit, {Up}stairs, {K}itchen
> Objects: A beer mug, a sword, and gold (5).
> With: Bob, the Barkeep; Frank, the vicious outlaw.

The game may be navigated by taking exits from one room to the next (some of which may not be visible). Objects may be acquired for profit (five gold pieces), health (beer), or other uses (swords). The inhabitants of the game may be human players (such as Frank), who lead interesting lives filled with adventure and heroic feats; or non-player characters such as Bob, who generally fill necessary, yet boring, roles. There is no definite way for a player to distinguish between a real human and an NPC.

Why use an MMORPG?

First, why use a game? Because games are meant to be fun and everyone has an intuitive idea of what qualities are necessary to produce an enjoyable gaming experience. One important property of games that contributes to their enjoyment is a social element. MMORPGs allow an arbitrarily large number of players to enjoy the game simultaneously, thus everyone who participates in building the game can play the game.

Since all our students are familiar with games and, if only in passing, with MMORPGs, the idea of the game is easily conceived. Since the game models the real-world, the contents of the game are also easily conceived. Thus, the only thing left for the students to do is to translate already familiar elements into code. An additional benefit of this project is that it is open-ended enough to include students of all skill-levels. As an example envision a taxi cab. It is simple to relay, with appropriate formatting, one player's speech to all the other passengers; it is more difficult to parse this speech for possible destination commands given to an NPC cabbie. More complex situations are conceivable since NPCs and other elements of the game may be subject to the Turing test.

Project Outline

Why a single day?

Planning to do the project in a single day was purposeful. We felt that doing so would provide the students a better view of the big picture, maximize their excitement, and promote better attendance by providing less opportunity for scheduling conflicts. Additionally, planning for only a single day meant that the project would not interfere to any great degree with the students' normal schoolwork.

Preparation

A wiki page (at https://wiki.umn.edu/view/CsGame1901/) for the project was set up and students were directed to it. The page included surveys to determine the theme of the project and what time we would start work on it (on a Saturday morning; the students were enthusiastic and

chose 8:00am). The wiki had a break-down of various sub-projects and descriptions of what sorts of activities those projects would involve. Some of our sub-projects were: server-coding, world building, combat systems, storyline, scripting-language coders, economics, database-coding, and policy. The wiki also included a list of design considerations. These questions were meant to cover every aspect of the game from beginning to end along every possible route of development. Although this was clearly impossible, having such a list of questions encouraged students to develop their own and, when the questions were applicable, they encouraged independent work freeing the teachers up to concentrate on other problems. The essential point of the preparation was to eliminate as much as possible the need to consider anything but development on project day.

Project day

We met at 8:00am as a large group in a lecture hall and discussed games in general, gradually focusing to what qualities our game needed to include. We then had the students divide themselves initially into teams to tackle critical sub-projects (e.g. server and database coding) with the plan that, as they completed these they would move to less important work (e.g. economics and policy). Abstraction was critical because many of the functions which various teams used were not coded till much later in the day. This also meant that the students had no way to test their code till the game's critical components had not only been coded, but debugged as well. The students were instructed to keep a running log on the wiki and to post to other teams' wiki pages for communication. We had scheduled twelve hours of coding, but, for reasons which will be discussed below, stayed for fourteen. The day included a scheduled pizza lunch and no scheduled supper. Lunch was a good break and the students used it as an opportunity to discuss their work with other teams.

Follow-up

In the end, there were eight students who remained (out of more than 40) till 10:00pm debugging and combining all the other teams' code. These were generally average to above-average students, but, universally, they wanted to be there. This was so much the case that we continued the project for another week and a half during which the students met without any guidance.

Analysis

Leadership

Students developed their team leadership spontaneously. This process generally happened quickly and effective leaders emerged. Small, coding-intensive teams were generally led by the most skilled programmer of the team. This is likely because this programmer was able to distill from the many available options those that were important and develop, independently, solutions. Larger teams developed administrative leaders who facilitated communication within their sub-groups. The lack of communication between separate teams was a problem and will be discussed shortly.

Once leaders had emerged, the project could be directed through them. The leaders had a more intimate knowledge of their team's work than the teachers could and the teachers, with an intimate knowledge of the "big picture," were able to suggest appropriate modifications.

A Single Day

The students' comments were divided as to whether a single day was the best choice. The reasons cited for this generally indicate that time lost organizing teams, developing leaders, defining problems, and forming new teams to fill development gaps could be gained back by running the project over two days. We contest this: although the follow-up work was publicly announced and the entire class was invited to join in on it, only eight students took part. Ameliorating other pitfalls (mentioned below) will probably not eliminate the aforementioned problems, but it may make a second day unnecessary.

Pitfalls

There were five major pitfalls in the organization of the project: (1) lack of comfort with abstraction, (2) lack of a skeleton for the code, (3) team size, (4) inter-team communication, and (5) a failure to familiarize students with text-based MMORPGs.

1. Although we had used abstraction throughout the semester, many of the students still felt uncomfortable writing code using self-defined functions the contents of which they would not code themselves. This was especially so when no one had yet chosen to do the coding.
2. The project began without a single line of code having been written beforehand. Our server and database coders dealt well with this; the server coders learned TCP/IP commands and produced a bug-free product in six hours, but, on the whole, having skeleton code would have been valuable. Since it easy to predict the essential components of the project, we think that a dictionary of these functions should be provided so that those students uncomfortable with abstraction have something to work with and so that these crucial elements are coded in a timely fashion. The degree to which this skeleton is fleshed out is variable. Not having a fully-functioning server meant that some of the most capable students were absorbed by that problem. They learned from the experience, but the game, as a whole, did not progress as far as it might have had they been free to work on other problems. This question is fundamental to the project as a whole: should programming the game involve coding its engine or merely developing in-game systems? Both lead to learning, but one presents a greater chance of "completion" and its attendant satisfaction.
3. Team size was another problem as most of our teams tended to be too large. Those students who worked in small teams on a problem tended to report more positive experiences, as did those students who led teams - regardless of the size. A balance must be struck between forcing a problem on the students and allowing them to all tackle the problem which seems the most enjoyable.

4. The students' comments uniformly cited lack of communication as the worst problem. Many of the teams attacked their problems and were excited about what they were individually doing, but forgot to communicate their work to others. The divisions between teams contributed to this: although it made sense from a coding perspective to separate the combat systems teams from the historical team, this didn't make sense from a plot perspective. Each team can be thought of as being part of several other teams and it is important to appreciate this and point it out. We had anticipated that providing the students with a wiki would promote organized communication, but updating and reading the wiki proved both time-consuming and boring. People leaving the project also presented a problem, often setting teams back. An early warning to students about the planned project dates is essential to avoid this.

5. Although we were correct in anticipating that using an MMORPG would aid students in conceptualizing the project, we did not anticipate the necessity of the students' playing text-based MMORPGs. Although we had suggested that they do this and provided links for them (to *Redwall MUCK* and *Merentha*), few students took advantage of this. We realized, belatedly, that the text-based environment must be experienced before students can fully appreciate its unique properties and visualize what makes such an environment convincing to a player.

The nature of the labs the students had done earlier in the semester did not adequately prepare the students for the different style of thought and action that would be necessary for this project. Many of the above problems could perhaps have been avoided by stating more explicitly this distinction: that there would be few, if any, guidelines; that communication would be essential and have to be continuous; that they must take the initiative and code boldly.

Student response

We have collected written comments in free form from the students. Comments are very detailed, and include personal opinions about their individual experience as well as detailed suggestions on how to improve the project for future students.

The students uniformly reported enjoying themselves and learning from the experience. Many of the comments on learning concerned learning the importance of teamwork, the difficulty of establishing communication among teams, and a better understanding on how to work on large-scale projects. Many students mentioned a greater appreciation for the difficulty of the development process. Many also mentioned that it was the first time they used a wiki.

Several students were pleased that Scheme, which had hitherto seemed useless, suddenly had a real-world application.

Many of the comments mentioned the project's relation to the "real world." The students felt that their experience was indicative of how things really worked and were grateful for the opportunity to experience that.

Overall student performance

It is hard to assess the impact of this specific project on the overall performance of the students in the class, since multiple other active learning opportunities have been included in the class.

In Spring 2007 we repeated some of the activities we did in Spring 2006, i.e short quizzes in preparation for the exams, written homeworks, pop quizzes, a lab using the Sony dogs AIBO to engage the students in a collaboration/competition activity (Chilton & Gini 2007), and used a student management team (Nuhfer 2003). We added multiple ways for students to get bonus points by solving additional problems. Many students took advantage of the opportunity to improve their grade, and, as a result, learned more. We have also added this game project that student could do for some limited extra credit. The number of students who participated (40-45) was significantly higher than what we expected.

An indication of the importance of active learning comes from the grades of the students in the course over the three semesters we worked on assessing and improving student learning.

The grades, as indicated in the table, are higher than in the past. Many more students attended the lectures all the way to the end of the semester, which might have contributed to their improved performance. All the students who completed the class in 2007, except 1, passed with a grade of C or better and only 3 students failed the class, compared to the 26 who failed in 2005.

	Spring 05	Spring 06	Spring 07
A	45	56	67
B	40	25	36
C	23	21	16
D	4	3	0
F	26	11	3
Total	138	116	122

Table 1: Grades of the students in the course over three semesters. The grades are grouped by including in the same line +/- grades.

Conclusions

We have presented a project where beginner students were engaged for a full day of programming to build a MMORPG using Scheme. Students were given limited directions and no software was prebuilt. Students had to use the knowledge they acquired during the course and lots of teamwork and coordination to pull this off.

We feel that the project has merit and could serve as a useful tool in the course, in particular if a skeleton of the game software is built, if the students are made comfortable with abstraction earlier, and if team sizes and goals are more explicitly delegated. The project will be repeated in an entry-level C++ course in Spring 2008 with the above modifications. We are considering using this project as a regular

project in the CS1 course, and extending it by adding some AI in the undergraduate AI course.

Acknowledgements

We would like to thank the Bush Foundation for their support of the program to promote student learning in large classes that made this project possible. We would also like to thank Paul Baepler and J. D. Walker for their continuous support and suggestions on how to improve our CS1 course, and John Chilton for all his help in this project.

Work supported in part by the National Science Foundation under grant DUE-0511304.

References

Beyer, S.; Rynes, K.; Perrault, J.; Hay, K.; and Haller, S. 2003. Gender differences in computer science students. In *Proc. of the 36th SIGCSE Technical Symposium on Computer Science Education*, 49–53.

Childress, M., and Braswell, R. 2006. Using massively multiplayer online role-playing games for online learning. *Distance Education* 27(2).

Chilton, J., and Gini, M. 2007. Using the AIBOs in a CS1 course. In *AAAI Spring Symposium – Robots and Robot Venues: resources for AI education*, 24–28. AAAI Press, Technical Report SS-07-09.

Hughes, G., and Scott, C. 2005. No pain, no game: Use of an online game to explore issues of online identity and the implications for collaborative e-learning. *E-Learning* 2(4).

Johnson, R. T., and Johnson, D. W. 1988. Cooperative learning: Two heads learn better than one. *Transforming Education*.

Kumar, D.; Blank, D.; Balch, T.; O'Hara, K.; Guzdial, M.; and Tansley, S. 2008. Engaging computing students with AI and robotics. In *AAAI Spring Symposium – Using AI to motivate greater participation in Computer Science*. AAAI Press, Technical Report SS-08-08.

Nuhfer, E. B. 2003. Manual for student management teams. Idaho State University.

Prince, M. 2004. Does active learning work? a review of the research. *Journal of Engineering Education* 223–231.

Rao, T., and Mitra, S. 2008. Synergizing AI and OOSE: Enhancing interest in computer science through game-playing and puzzle-solving. In *AAAI Spring Symposium – Using AI to motivate greater participation in Computer Science*. AAAI Press, Technical Report SS-08-08.

Smith, K.; Sheppard, S.; Johnson, D.; and Johnson, R. 2005. Pedagogies of engagement: classroom-based practices. *Journal of Engineering Education* 87–101.

Steinkuehler, C. A. 2004. Learning in massively multiplayer online games. In *Proc. 6th Int'l Conf. of the learning sciences: Embracing diversity in the learning sciences*, 521–528. Lawrence Erlbaum Associates.

Wadley, G., and Sobell, J. 2007. Using a simple mmorpg to teach multi-user, client-server database development. In *2nd Annual Microsoft Academic Days Conf. on Game Development*.

Yee, N. 2006. The psychology of MMORPGs: Emotional investment, motivations, relationship formation, and problematic usage. In Schroeder, R., and Axelsson, A., eds., *Avatars at Work and Play: Collaboration and Interaction in Shared Virtual Environments*. Springer-Verlag. 187–207.