Games, Robots, and Robot Games: Complementary Contexts for Introductory Computing Education

Dianna Xu
Bryn Mawr College
101 North Merion Ave
Bryn Mawr, PA 19010
dxu@cs.brynmawr.edu

Douglas Blank
Bryn Mawr College
101 North Merion Ave
Bryn Mawr, PA 19010
dblank@cs.brynmawr.edu

Deepak Kumar
Bryn Mawr College
101 North Merion Ave
Bryn Mawr, PA 19010
dkumar@cs.brynmawr.edu

ABSTRACT

Using games to teach introductory computing courses provides another context with which to exploring the possible attraction, retention, and education of a new generation of computer science (CS) students. At Bryn Mawr College, we have been actively exploring these contexts and have identified four that have great promise for use in teaching introductory computing courses: visualization, multimedia, robotics, and, most recently, games. We are currently using and analysing robots and have some preliminary results. We believe that much of what we have learned in using robots in the classroom can be applied to the other contexts, especially gaming. In addition, many aspects of gaming can also be used in an introductory course using robots. This paper will explore robotics, gaming, their interactions, and provide suggestions on how best to proceed in making the most out of games in the classroom.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education - *computer science education, curriculum*

General Terms

Design, Experimentation

Keywords

Game development, robot, undergraduate education

1. INTRODUCTION

At Bryn Mawr College, we have been exploring a variety of alternate ways of approaching the introduction to computing course, CS1 (see [4]). Recently, an additional variation has been identified: teaching computing in "context" [17]. Teaching in context meshes very well with our other patterns of curricular design. By framing computer science in a particular context, a topic can be made concrete and can be shown to have direct relevancy to a student's understanding. We have identified four contexts to explore: visualization, multimedia, robotics, and gaming. In this paper, we will describe our initial findings in using robotics as a context, and how those findings could be applied to gaming in the introductory course.

The effect of games on student interest and motivation is apparent and the addition of a "games" course in a CS curriculum is gaining momentum. Game development can be seen incorporated into the CS curriculum in all levels and in various ways. A number of schools have chosen to use a "games first" approach in CS1 or CS2 classes [2], [10], [9], while others are creating entire programs or concentrations focusing on game development [1], [5], [7], [11]. Some started introducing games as an experimental special topics class since the early 90s [14], while for many others including Bryn Mawr College, it is still a new teaching methodology that is being explored.

Teaching computer science in a small, all-women liberal arts college has its unique set of challenges. Very few of our majors come to Bryn Mawr College with any intention of pursuing a degree in computer science. This is due to a variety of reasons, including prior negative exposure or complete lack of exposure to computer science. It has been our experience, however, that many of them do end up changing their minds and decide to major after having taken our introductory course. The biggest challenge turns out to be to get the students to want to take a first course in computer science in the first place, and preferably to get them to take it in their first year. We need ways to rework our introductory course so that it is not just about learning how to program, but that during the process, the students will also experience some exciting application areas of computer science. We believe that providing context to the education is motivationally important and just plain more fun.

We would also like our students to learn about the potential of how the computer may assist them in their studies, whether or not they continue to take courses in computer science. The computer science program at Bryn Mawr College differs from many in that instead of training only computer scientists, it aims to produce women who have a full grasp of modern technology, the role it plays, and its implications in society, regardless of their future career plans. In the access provided to individuals to previously inconceivable amounts of experience and information, we believe computer science has emerged beyond its traditional perception as a subset of Mathematics and Engineering to the very core of a liberal

GDCSE'08 Feb. 28-Mar. 3, 2008, On board the Celebrity Century cruise ship, departing from Miami, FL

Figure 1: The IPRE Robot Kit, Fall 2007. Clockwise, from top-left: lunchbox carrying case, gamepad, flashlight, Scribbler robot with IPRE Fluke (camera and serial Bluetooth adaptor), USB Bluetooth dongle (for computer if it doesn't have Bluetooth), and drawing pens.

arts education [4].

2. COMPUTING IN CONTEXT: ROBOTICS

Two of us (Blank and Kumar) are co-principal investigators of the Institute for the Personal Robot in Education (IPRE) (`http://www.roboteducation.org`), hosted, in part, at Bryn Mawr College. IPRE, a joint partnership with the Georgia Institute of Technology and Microsoft Research, is developing a personal robot, software, and curricula to help teach introductory computing courses [3]. The IPRE vision is that the text for an introductory course would come shrink-wrapped with a ready-to-run personal robot in the same price range as current CS1 texts (see Figure 1). In this instance, robotics is the context and provides the intrinsic motivation to both the instructor and the student to explore the science and engineering behind it.

We believe that much of the philosophy of the robotics-context can be applied to other contexts, especially gaming. To explore this idea, we first developed a set of general principles.

Let the needs of the curriculum drive the design of the materials The design of the context (such as the personal robot) should be motivated by the requirements of our new curriculum and should be an outcome of feedback obtained from students in our pilot course offerings. For example, we found that robots designed for education don't have much in common with robots designed for industry. This suggests that games for education might not have much in common with games for industry.

Let the tasks of the context drive the motivation to learn. The table of contents of the robotics-context text does not focus on programming constructs, such as "loops", "variables", etc. but rather topics like "Making Music", and "Dancing." The students are introduced to loops and variables as they need them to solve a particular robotics-based problem. In fact, many of the students "invent" recursion on their own, because they need it and find it obvious.

Use tools that are easy to use, scale with experience We want the students use of the tools (computer, programming language, IDE, etc.) to be such that they are not designed specifically (and only) for use in CS1. We want the entire programming environment to be "pedagogically scalable." This way, concepts acquired in the new CS1 easily carry over into more advanced computing situations without the need to change the programming environment. We want students to be immediately able to jump to the most interesting problems in a context. This puts the emphasis on the algorithm rather than on programming language or programming technique.

Create an accessible, engaging environment for new, diverse population of students It has been argued that the introductory computer science curriculum is broken and is in need of a major overhaul [3]. We are taking the issue of accessibility to a wider population as our primary goal. We have recognized the attractiveness and engagement potential of contexts like the personal robots and gaming. However, we are also mindful of the adverse affects of certain technological contexts can have on people from different gender and backgrounds. Our curricular materials are an attempt to address these issues.

Computer science is not programming While programming is central to our approach to CS1 we are also conscious of avoiding the misperception that programming is all there is to computer science. Students from the new CS1 should come away with a solid understanding of the scope of computing and the role of programming in it. The use of "game design" (see below) falls into this category.

Make computing a social activity There is an explicit attempt in our approach to make computing a social activity. By this we mean that we will strive, in our curriculum, to make every aspect of the learning process a collaborative activity. Students learn from each other and by working on their robots in their own environments (dorm hallways, dining halls, study spaces, labs) and interacting with others in meaningful ways.

Performances vs. competitions All robot exercises in the course include demonstrations. However, the demonstrations are to be depicted and evaluated as performances and not as competitions among peers. We have found that competitions tend to attract only a small population of student body and serve to deter many students away. However, a non-competitive, collaborative, and social environment encourages learning and motivates students to strive for higher goals.

Make computing a medium for creativity Creativity is central to robot design and we have included several creative aspects into the curriculum. Examples include exercises that demonstrate robot behaviors like dances, choreographed movements, music and song generation, movie making, game playing, robot application design, etc. Most exercises will be open ended (i.e. correctness of a program is not determined by a limited set of output) and encourage students to experiment, play, and be creative.

As is evident from above, the curriculum for a CS1 course that uses personal robots takes a fresh perspective. While it deviates from the traditional approach, the overall coverage of topics provides a comprehensive exposure of traditional CS1 concepts. In fact, in many ways, it goes beyond the traditional notion of a CS1 syllabus (see Figure 2). Yet, the key driving factor in the design of this curriculum is the robot context. The materials have been used in six pilot offerings

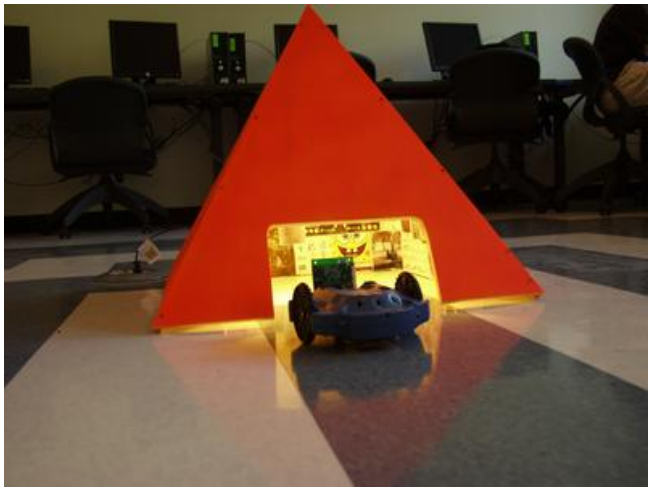

Figure 2: For one of the assignments, students program the Scribbler to make its way into the pyramid, photograph the walls, and then exit.

of CS1 courses: at Bryn Mawr College in spring and fall 2007 with 60 students (nearly all women), at Georgia Tech in spring, summer, and fall 2007 with over 200 students.

Extensive feedback (both qualitative and qualitative) was collected from all completed courses. The results obtained from students can be summarized these main points [15]:

1. When taught in context, students still learned CS concepts
2. Robots made learning experience more hands-on, tangible, and exciting
3. Most frustrating parts were dealing with robot hardware inconsistencies
4. Viewed CS as a type of logic and problem solving; requiring patience & thought
5. Discovered that CS and robots are applicable to the real world

These are informative and suggestive findings. We will now turn to a computer game design course and explore it in these points.

3. A GAME DESIGN COURSE

"Video Game Design and Programming" is a special topics course where a game of the students' own design is developed as a semester-long group project. It was initially offered at Bryn Mawr College in the spring semesters of 2005 and repeated in the spring of 2007. In many institutions' CS curriculum, a game course is usually offered at the junior or senior level to majors with significant programming experience and preferably a prerequisite course on computer graphics. At Bryn Mawr College, it was important to the department and the instructor that the course be accessible to a wider audience than a traditional CS course would typically entail. In a small liberal arts college with a rich tradition of excellence in humanities and social sciences, the possibility of rewarding interdisciplinary work in such a class is clear. On the other hand, it also allows for larger class sizes than the small number of CS majors would otherwise make up, thus improving group projects possibilities and class dynamics. Ultimately, it was also our hope that this course would showcase an exciting and fun application area of computer science, as well as facilitate communication between our majors and non-majors. Thus, although "Video Game Design and Programming" was listed as a 200-level CS course, prior programming experience was not required and we made efforts to advertise to a wide variety of students. The final enrollment makeup still had CS majors as the majority, but also included a number of students who have never before programmed. Some were self-professed "avid" gamers but others were simply curious about a "fun" course.

3.1 Focusing on Game Design

One approach that is different in the game course at Bryn Mawr College, is the focus on game design rather than game development. The decision was partly based on making the course a more creative and social activity. Students in the class were asked to perform a series of exercises in class and out in order to familiarize with the concepts such as game play, rules, feedbacks, rewards and punishments and game balancing. It was also a great way to get the brain storming started so that they may firm up on the design of their own game. Some examples are listed here.

1. Game co-tutorial. The students were asked to teach a friend or a classmate to play a game. They were then to report on the tutorial experience addressing issues such as how it takes for the trainee to get good enough to have fun, what aspect of the game did the trainee find easy to learn or confusing, and whether that was as expected.
2. Modifying poker. The students were allowed one rule change (modify, add or delete) of the popular poker game Texas Hold'em. The resulting game must still be a playable variation. This allows for a deeper understanding of the roles of game rules.
3. Rapid games. The students were asked to design fast repeatable games that can be played in under a minute, such as the classic rock-paper-scissors.
4. Theme-based games. The students drew themes such as "vampire", "subways", "life at the sea" from a proverbial hat and designed games accordingly.
5. Game hacking. The students were asked to "hack" an existing open-source or freeware game such as Quake and Crossfire. Only small cosmetic changes were required, such as when an enemy was shot in Quake, he turned into a chicken. Creativity was greatly encouraged.
6. The Boardgame test. When the students present their own game proposals, they were asked to write down their game rules and play it as a board game, to test the completeness and effectiveness of the rules.

Besides design, time was also spent on other areas of computer science with applications in games, such as animation and kinematics, game AI, networked and multiplayer games, etc.

Figure 3: Student game - Otis: splash screen

Figure 4: Student game - Otis: screen shot

3.2 Game Development

Most modern 3D games require multi-million dollars of investment, teams of well trained professionals across many disciplines, as well as years of hard work under often very harsh deadlines. Postmortems for games published in game developers' websites and conference proceedings offer great glimpses into the industry. The now open-source game engines such as Quake and Doom provide further insights into game development as software engineering projects. The game industry and its history from early assembly games to modern game engines and data driven designs from a software engineering perspective was of great interest to the students [6], [12]. On the other hand, what can be realistically achieved in an academic semester must be significantly scaled down from industry standards. The students were told to propose and design the game as a whole, but implement at most only one level or one quest, as a concept demonstration. It was nevertheless a constant battle of time to pull all the pieces together for most of the teams. Many found that in the end they accomplished much less than they originally thought they could, but most were able to come to a functional minimum so that it was possible to see how a polished and complete version could be achieved, given more time (see Figures 3 and 4).

3.3 Lessons Learned

The course was well received and obtained very favorable student reviews. The dynamics of having both majors and non-majors worked to great advantages of all groups. The students, particularly the CS majors, were grateful for the addition of skill sets that were sorely in need and were quite unanimous that those contributions were equally important as programming.

There are startling differences in the games genres that male and female students find interesting. None of the all-female groups gave any thought to any type of shooting games. The most popular choices seemed to be socially motivated Role-playing games (RPG) or mystery/puzzle centered adventure games. The male students on the other hand must be reminded over and over again that First-person Shooters (FPS) were not the only type of games.

The greatest frustration students experienced during the course were the inconsistencies as well as incompatibilities of the many different software packages involved in content creation. Typically a group used two or three separate toolkits for modeling, then additional ones for texturing and animation. The importation into the game engine often did not work smoothly, resulting in texture loss or other artifacts. A student stated: "I felt like a lot of time was wasted putting 'things' into the game. I would have preferred spending more time perfecting the flow and potentially improving the 'fun' of the game."

A survey done on the female students after the class revealed that while they liked having control over the creation of their own games, all found the game development process "rather tedious" and none would consider game development as a future career option. In general the women spend much less time on gaming than their male classmates and for those that do play, none plays FPS. These differences between the genders have been observed and argued over before [13], to the extent that some believed that including games in the curriculum actually served to discourage rather than encourage female enrollment [16]. We believe that instructor awareness and attention to the differences will be essential in preventing such difficulties.

4. A GAMING CONTEXT

It is our hope that we can learn from the success of incorporating context-relevant robotics into our introductory curriculum and work computer games into a similar frame work. After the explorations of the two initial offerings of the course, we are proposing the following recommendations.

A game engine designed specifically for educational purposes We believe that in order to develop an introductory course with a gaming context that incorporates the goals and focuses outlined in Section 2, a new pedagogically-oriented game engine designed with the intention of being used in CS1 will be of great importance. Such a game engine should ideally be object-oriented, come with a 3D primitive library of good size and variety, as well as modeling, texturing and animation tools that are designed to work with such an engine. Such a software package will free the students from spending an unwarranted amount of time on content creation.

An in-context course needs an in-context textbook Despite the recent popularity for game development courses, it remains difficult to find appropriate textbooks. Game design and game development are typically covered in entirely different books. There's usually a need to incorporate a further book on the particular game engine of choice, and often even more reading materials on other tools used for content creation and game art.

Books that cover game development are often too industry-oriented. They are laden with technical concerns and tricks, but lack a more general pedagogical point of view. Our students were particularly interested in the history and evolution of the gaming industry from a software engineering perspective and reading materials on this aspect were few and far in between. In addition, other areas of computer science relevant to game development such as computer graphics, artificial intelligence, networks and multimedia are often never properly introduced in context, but appear in the form of abrupt algorithm outlines to solve a problem or meet a goal.

Textbooks have been developed for other non-conventional but in-context approaches for teaching introductory courses to great success [8]. We believe that a similar effort is necessary for a in-context gaming course.

Be mindful of the gender differences As mentioned in subsection 3.3, women tend to have a very different set of focuses in computer games. They are typically more socially oriented. It is often the case that for many of our female students, it is not enough to create something (software, games or otherwise) just because it's fun. They tend to show more interest and engagement if it has real life impact or human content. When asked why she was not interested in FPS games, one of our students said plainly: "Because it's boring. The whole point is to kill anything that moves."

In addition, there are societal and cultural (often negative) preconceptions of the type of people who play and develop computer games. There is a real danger that introducing games into the curriculum will serve more as a deterrence rather than encouragement. The difficulty is easily surmountable if the instructor is mindful of such issues and address them accordingly.

Games and robots have much in common in providing context for CS-1 To that end, we experimented in our most recent offering of CS110 (Fall 2007), the IPRE CS-1 course using the approach of personal robots. Robots and robot-related assignments were given for most of semester. Towards the end, students were given a project of creating a video game in the last few weeks in groups of 2-4. The project was made more interesting through the gamepad support in Myro (a Python based library developed at IPRE). The final student games showed amazing creativity and evident student enthusiasm (`http://www.cs.brynmawr.edu/games`). Student survey showed that they really enjoyed both the robots and the games aspects of the course. We also have phenomenal and unprecedented pre-registration numbers for CS-2 next year.

Acknowledgments We would like to thank our colleagues at the IPRE, especially Microsoft Research.

5. REFERENCES

[1] L. Argent, B. Depper, R. Fajardo, S. Ghertson, S. Leutenegger, L. M., and J. Rutenbeck. Building a game development program. *IEEE Computer*, 39(2):52–61, 2006.

[2] J. Bayliss and S. Strout. Games as a "flavor" of cs1. In *SIGCSE 2006 Proceedings*, pages 500–504, March 2006.

[3] D. Blank. Robots make computer science personal. *Commun. ACM*, 49(12):25–27, 2006.

[4] D. Blank and D. Kumar. Patterns of curriculum design. pages 77–86. Kluwer Academic, 2003.

[5] R. Coleman, M. Krembs, A. Labouseur, and J. Weir. Game design & programming concentration within the computer science curriculum. In *SIGCSE 2006 Proceedings*, pages 545–550, March 2006.

[6] D. Dalmau. *Core Techniques and Algorithms in Game Programming*. New Riders Publishing, 2004.

[7] T. Fullerton. Play-centric games education. *IEEE Computer*, 39(2):36–42, 2006.

[8] M. Guzdial. *Introduction to computing and programming in Python: a multimedia approach*. Pearson/Prentice Hall, Upper Saddle River, NJ, 2005.

[9] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. In *SIGCSE 2007 Proceedings*, pages 115–118, March 2007.

[10] M. Lewis and B. Massingill. Graphical game development in CS2: A flexible infrastructure for a semeseter long project. In *SIGCSE 2006 Proceedings*, pages 505–509, March 2006.

[11] J. Murray, I. Bogost, M. Mataes, and M. Nitsche. Game design education: Integrating computation and culture. *IEEE Computer*, 39(2):43–51, 2006.

[12] K. Oxland. *Gameplay and Design*. Addison Wesley, 2004.

[13] P. Palma. Viewpoint: Why women avoid computer science. *Communications of the ACM*, 44(6):27–30, 2001.

[14] I. Parberry, M. Kazemzadeh, and T. Roden. The art and science of game programming. In *SIGCSE 2006 Proceedings*, pages 510–514, March 2006.

[15] J. Summet, K. O'Hara, T. Balch, S. Tansley, D. Blank, and D. Kumar. Designing personal robots for education: Hardware, software and curriculum in practice (under review). 2007.

[16] H. Walker. Do computer games have a role in the computing classroom? *ACM SIGCSE Bulletin*, 35(4):18–20, 2003.

[17] S. Yarosh and M. Guzdial. Narrating data structures: the role of context in CS2. In *ICER '07: Proceedings of the third international workshop on Computing education research*, pages 87–98, 2007.