Everything I Needed to Know About Teaching I Learned in Kindergarten: Bringing Elementary Education Techniques to Undergraduate Computer Science Classes

Shannon Pollard
Elon University
Elon, North Carolina
spollard@elon.edu

Robert C. Duvall
Duke University
Durham, North Carolina
rcd@cs.duke.edu

ABSTRACT

By expanding the teaching styles used in computer science classrooms, we can expand the audience of students that enjoy and excel in technology. Rather than focusing on major curriculum changes or new programs specifically for non-traditional students, we propose that relatively simple expansions in teaching style can have significant results. In particular, we advocate incorporating teaching techniques reminiscent of kindergarten: games, toys, stories, and play. These techniques promote an active learning environment, level the playing field for non-technical students, provide motivation beyond grades, and make class time fun. In this paper, we want to acknowledge the many activities others have proposed by providing a coherent categorization of such activities and show how to use these techniques throughout the curriculum rather than as special experiences.

Categories and Subject Descriptors

K.3.2 [**Computing Milieux**]: Computers and Education—*Computer and Information Science Education, Computer Science Education*

General Terms

Human Factors

Keywords

Active learning, active teaching, classroom management, curriculum issues, pedagogy

1. INTRODUCTION

Several recent papers have suggested various non-standard teaching techniques, but there is no further evidence given that these are used consistently throughout the curriculum [3] [2] [18]. While most recent papers focus on the introductory programming course, many examples also exist for later courses in the curriculum [8] [4] [14]. In fact, many of the ideas presented in this paper are not our own. Where possible, we have cited others' work, but often ideas such as these are shared informally on listservs or BOF sessions. In particular, we would like to thank the many teachers that have participated in our annual "Toys Night" session while grading the Advanced Placement Computer Science exams. Our goal in writing this paper is to categorize these techniques, recount our experiences in using games and prizes, toys, action-based activities, and stories in our computer science classes, give lessons learned in incorporating these techniques, and encourage other professors to attempt to make activities a part of their daily routine.

In our previous work [17], activities and games were specifically relegated to lab sessions and not classroom sessions. Because of their success, we have worked to integrate these techniques more fully since that time. By considering a variety of motivational techniques, we were able to make learning interactive through games, physical manipulatives, role-playing activities, and storytelling a daily occurrence rather than a special event. The immediate result is a more fun classroom atmosphere. The long-term effects will hopefully include better overall student performance, student diversity, and perhaps increased enrollments.

2. GAMES AND PRIZES: MOTIVATION

Competitive games are commonly used in social science classrooms as well as elementary education [10]. Class reviews, homework, exam preparations, and even the exams themselves can be enhanced by the introduction of games and prizes. Students who are not motivated by grades can often be motivated by competition or team dynamics. The games can be simple and the prizes can be cheap; the fact that a game is being played is enough motivation for most students to study, participate, and even tutor one another.

All professors have experienced the uncomfortable silence of students staring at their desks, avoiding participation, when asked review questions for an upcoming exam. This was our experience also, until initiating "Dollar Prize Days", a whole class period devoted to reviewing by playing games. The same students who previously had blank stares for review questions were suddenly on the edge of their seats, trying to be first to "buzz" in or cheer on a teammate. The students understand that all game prizes come from the local everything-for-a-dollar store, yet they compete for first pick from the prize table to get the "best" of the prizes.

SIGCSE'06, March 1–5, 2006, Houston, Texas, USA.

For this exercise, the class is divided into five or six teams of at least three people. The game consists of several rounds of play: some are focused on the individual, to encourage everyone on the team to contribute; some are held only at the team level; and some are "all play" in which every team works on the same problem. For example, in the first round, each student is asked a short answer question. If the student can answer it without help, the team gets the points. If not, the student can consult with the team for half the points. If the team as a whole does not know, other teams get the chance to "steal" the points. Topics for a round of play include identifying vocabulary, predicting program output, finding syntax errors, and writing code. At the beginning of the game, the prizes are displayed with great fanfare, and at the end of the game, students get to pick a prize in the order their teams placed.

For these games, the students are strategically placed in groups ahead of time to ensure good competition and to encourage stronger students to work with their weaker teammates prior to the game day. Rather than every student working for personal success, the usual model for classes we have seen, the teams are working together for total success. This is an example of using cooperative incentives [25]; that is, the way students achieve their personal goals is by helping others achieve theirs. It has been shown that peer instruction is effective in improving overall success of students [6]. Team games provide the better students with motivation to provide this instruction.

Students who would not normally study to do well personally will instead study so as not to let down their team. In these games, it is usually obvious who did and did not study, and those who are not prepared are seen in the social atmosphere of the class as not being good team players. These students unfailingly are prepared the next time game day occurs. The team dynamic introduces positive peer pressure and social acceptance as motivators for success.

There are many resources for games to play, although the games do not need to be complex or take an entire class period to be effective. The books in the innovative "Head First" series [24] [9] include many non-standard exercises that can be used in Computer Science classes at all levels of the curriculum. These include puzzles in which the students fill in missing code, rearrange lines of code to get a desired output, match vocabulary words with descriptions, or find and fix syntax errors. Adding game-like elements to these puzzles like teams, points, prizes, or racing turns these simple exercises into a game and makes the class come alive.

Many traditional assignments and problems can also include a game-like element. We have found that students are more motivated on their projects when there is a competition involved. Assignments that are themselves games lend themselves naturally to competition. For example, our CS 1 class has a project for learning 2D arrays in which they create a strategy for playing Connect Four. At the end of the project, we hold a Connect Four competition of the computer players, where the winner gets a prize. Other common games that have been used in the classroom include Othello, Boggle, Minesweeper, and the dice game Pig. From this reasoning, it may not be surprising to find that several of SIGCSE's *Nifty Assignments* are games [16].

On the other hand, non-gaming projects can still include an element of competition. For example, using a basic graphics package in Java students can be encouraged to explore loops and even recursion to create artistic pictures or animations. Creating artwork has also been used in the theory class through an assignment on grammars that uses L-Systems to generate pictures by mapping grammar elements to Logo Turtle actions. The programs that produce the best pictures, as decided by the course staff and/or the students, are chosen and their output displayed around the classroom or the department to give students wider public recognition for outstanding work. We have found that students develop a much deeper understanding for the topic when motivated by creativity and competition rather than a simple grade.

Even for standard programming assignments, prizes can be given for submissions with the best design, most features, or most creative user interface. The prizes offered for winning range from small toys (as used in the game days described above) to smaller prizes like stickers or candy [13], to intangibles like extra credit and recognition. In all cases, our goal is to publicly reward those who do well rather than rely on the threat of grades for motivation. Positive reinforcement is simply more effective than negative reinforcement [25].

3. TOYS: MAKING THE ABSTRACT CONCRETE

While much current education research is focusing on using technology in the classroom [1] [21], when the subject *is* technology, it is often better to introduce non-technical props to make abstract notions concrete. For a student who is intimidated about learning technical skills, using physical manipulatives to explain concepts eases apprehension and gives confidence. The innovation of the original kindergarten, invented by Froebel in 1837, was the use of physical objects to describe concepts. He believed the concrete should be introduced before the abstract [20].

For example, a problem often seen in Discrete Math or Theory class is to prove that any 2^n by 2^n checkerboard with one square removed can be completely covered by L-shaped tiles. We have used this inductory proof exercise for the past two years, and the students have never completed a solution on their own during class time. This year, however, we gave each group of students a checkerboard, a "square removed" marker, and several L-shapes cut from paper. With this setup, every group except one was able to complete the proof without help. Making the problem components concrete enabled the students to more successfully analyze the problem.

It is easy to find physical objects that represent computer data structures, code constructs, and memory cells. Astrachan [3] gives ideas for using frisbees to illustrate parameter passing, beaded blocks for linked lists, and icky-poo for pointers. Poon [18] shows how to use stacking boxes and teletubbies dolls to illustrate arrays. Additionally, several educators, including us, use magnetic letters on the board to demonstrate various String functions, flash cards to search and sort, dice to discuss randomness, playdough with cookie cutters to represent memory allocation, rows of bottle caps glued to cardboard to help students verify array algorithms, and even M&M's to show light transfer in the basic radiosity algorithm for computer graphics. As these examples show, the use of toys in the class does not have to include complex scenarios; the simple presence of the physical metaphor puts the abstract concept into a concrete context.

The result of bringing physical manipulatives to class on a regular basis, rather than a few special days, is that students tend to look forward to coming to class to see what is in our "bag of tricks." Learning programming concepts looks more like play, and those that would normally be turned off to studying computer code are lured into understanding.

4. PLAY: KINESTHETIC LEARNING

According to the constructivist theory, "cognitive constructions develop through action" [12]. Students can be physically engaged in the learning process by acting out algorithms or being used themselves as physical props. Students can represent objects which have to work together to get a job done [2]. They can act as data points that get sorted, as nodes in a graph or network, as a loop while they perform exercise routines, as function records in a series of recursive calls, or as objects in a 3D scene that is being ray traced. Any activity that asks the students to get out of their seats and participate will add to the energy of the classroom and incorporate a new learning style[4]. This kind of active learning has been seen to increase motivation, give an opportunity for immediate feedback, and involve the class in higher-order analysis [5]. In addition, the student creates personal connections to the scientific concepts at hand [19].

These activities provide a way for the students to experiment with problem solving or other "deep" topics without having to know program syntax. As such, they are excellent techniques to use to introduce a topic, unit, or even an entire discipline. Often the activity results in material for discussion: the students may find an innovative solution to a problem, encounter communication stumbling-blocks, or come up with multiple solutions to choose from [19]. The problem solving process becomes the central topic and the syntax of the program is rightfully considered secondarily.

Kinesthetic learning again encourages the students to work together as a team to accomplish a task. Students are motivated to help one another because they care about the group as a whole [25]. The students that emerge as leaders during the activity are not always those students with the strongest technical background. Those who excel in communications and leadership will be highlighted by these activities and will gain confidence in the corresponding technical skills that the activity models. In particular, women respond well to these group activities [7]. The class as a whole will also retain the material better [2].

Students can also be motivated to study a topic outside of class by asking them to create a video, song, or commercial about the topic. For extra credit, we currently ask students to make up new lyrics for an existing song or a script for a commercial about any Computer Science topic they deem worthy. On their own, several students took the extra step to actually act out their lyrics or skits and record it, so now creating a music video is a standard part of the assignment. Several other educators have made more serious versions of these activities a regular part of their course, rather than an extra credit. For example, one assignment asks students to show how specific Computer Science topics appear in everyday places like the grocery store.

5. STORIES: EXTENDED METAPHORS AS MNEMONIC DEVICES

"There is no idea worth explaining that cannot be explained by a good story." This anonymous quote captures the idea that storytelling can be used in all classrooms to explain new concepts. We can expand our idea of explanations from lectures to stories, where the concept is embedded in a context that is more easily recalled later, more interesting to the listener, and even more fun to deliver.

Papadimitriou [15] states, "incredibly, many people do not find computer science interesting. To make storytelling an integral part of computer science education would go a long way toward correcting this." There is evidence that women in particular are interested by storytelling [15]. However, the most compelling reason to use stories is that it helps the students remember concepts.

In our CS 1 course, we relate programming to playing a game, and knowing the syntax of the programming language as knowing the rules of the game. In these terms, the stories help our students to remember the rules at game time. For example, one story we use discusses variable scope by saying that variables are born in a country and are not allowed to immigrate to other countries. While programming in the lab, a student made an error in variable scope. One lab partner read the error message and said, "Oh! Variables born in Methodpotamia must stay in Methodpotamia."

In addition to being useful mnemonic devices, the stories also help the students understand abstract concepts and generally make a fun start to class. Eighteen CS 1 students were surveyed about the use of stories in class. All but one student marked either "agree" or "strongly agree" with the statement that the stories were a fun part of class. Many questions were given on the survey asking about whether specific stories used in the class were helpful with understanding or remembering certain concepts. While a couple of students marked "neutral" as their feelings about the helpfulness of the stories in these ways, only one student *disagreed* on any story's helpfulness in understanding or remembering a concept. One student remarked, "The stories were a fun way to help further my understanding (or clear up some confusions) of certain topics."

Several educators have told us that they use stories by Dr. Seuss [22] [23] as well as the story of "Martin and the Dragon" by Dave Touretzky [26] to enhance their teaching of recursion topics in particular. Yeoman [27] says, "Storytelling, magic, and the enhancement of the human imagination have always seemed to me to be a very big part of what teaching should be about." Using stories in our classes will stimulate the students' creative minds as well as teach technical skills.

6. POSSIBLE PITFALLS

There are several words of warning that should accompany our enthusiastic advice to use these techniques in the classroom. Many of these warnings may seem like common sense, but we feel they are worth repeating since every new class activity should be examined from the students' point of view to be sure that the elements used are appropriate and inoffensive, contribute to an overall positive class atmosphere, and do not cause any student to be intimidated.

We are often unaware of the social statements we may be making by using a specific example, game, or story. Stories that are presented should not have socially incorrect statements or show a bias in race, gender, or creed. When working in groups, be careful not to isolate any minority student [7]. Physical activities should not ask a student to

reveal personal information that he may not be comfortable sharing or to do a physical "stunt" that may be inappropriate. It is important to be aware of the affect that using physical activity in the classroom may have for any student with physical challenges[4].

Of course, you should not sacrifice the material you are trying to teach for the sake of fun; it is vital that the metaphors you build by associating an abstract concept with a concrete prop, activity, or story are accurate and stand up to close scrutiny by the students. For example, we once used the metaphor of a chef in a kitchen following recipes as a metaphor how a computer worked. Initially, we were satisfied because, on the surface, there were several obvious positive features about this metaphor: the chef followed a set of instructions as does the CPU, the kitchen counters acted as the computer's RAM, and the pantry and refrigerator acted as its main memory. However, as the students probed the metaphor further, we found problems trying to explain things like input and output. In particular, a computer can easily produce many identical copies of a program's output, either on the screen or on paper, while a chef can only produce a single meal. In the end, we had to create a "magical" dumb waiter that copied the meal as it transported it from the kitchen to the outside world.

Using games in the class creates a competitive atmosphere, which can have some drawbacks. Often the students' personal strengths and weaknesses are made public in the games, which in addition to motivating the student, may make him feel uncomfortable. It should be the case that the same students do not always win every game. Elements of chance added to the game can give all students a fair shot at winning. When rewarding the "best" submissions for projects, technological skill should be balanced with creativity, effort, and innovation so that students with different strengths have the opportunity to win. In general, if enough opportunities are given, everyone in the class can be a winner at some time during the semester.

Finally, it is not the case that all techniques are appropriate for every class. Some activities work best in small groups, some only work if there are a large number of participants [1]. Some ideas may be appropriate for one group and not another, depending on the background of the students or how well they know one another. Toys used as metaphors work only when all the students can get close enough to the action to see, and they work even better when there are enough for the students to use themselves. The dynamics of each class need to be taken into account when deciding how to use these techniques.

7. A NEW CLASSROOM ATMOSPHERE

Many educators have reported that when they try non-standard techniques such as those we have presented, it just does not work: the students make little effort to participate; they do not understand the point of the exercise; or the teachers themselves lose control of the activity. We believe these problems are inherent in only occasionally using techniques that directly challenge the traditional classroom experience because it is not clear that all students will respond positively to a change in classroom dynamics. In regard to letting students engage in problem-solving, Felder and Brent [7] state, "The problem is that while the promised benefits are real, they are neither immediate nor automatic. The students, whose teachers have been telling them everything they needed to know from the first grade on, don't necessarily appreciate having this support suddenly withdrawn." While some students will be reluctant to participate, providing motivation such as prizes and an atmosphere without the pressure of grades encourages all to get involved. Others have found similar success motivating students to leave their comfort zones to participate [11]. This implies that using these kindergarten techniques consistently throughout the curriculum helps make students more accepting of them which, consequently, makes the activities themselves more successful.

Our success has led us to experiment with using toys in other contexts than strictly teaching. Again, simple elements reminiscent of kindergarten can add to the overall fun atmosphere. For example, we have been surprised at how much students like stickers or smileys on their papers [13]. When students are working in the lab and have a question, they place a small stuffed animal on their computer monitor. In addition to letting the students continue working while waiting for a professor or teaching assistant, the toys are another fun element incorporated in the class.

In addition to these practical benefits, we have discovered many pedagogical benefits as well because these techniques match very well with several pedagogical patterns [13] that attempt to describe the best practices of teaching. For example, role-playing a topic before treating it formally in a lecture allows students to see the topic multiple times in multiple formats, explore it at different levels, and experience the material rather than passively receiving it. Using a game show style format to review for an exam gives students confidence because they have a venue that forgives failure. Even when students are writing songs or making videos at the local grocery store, they are experiencing the discipline from a different, more real-world, perspective than can be achieved in the classroom.

While each of these techniques has educational merits, when used in conjunction, they create an atmosphere of fun, interactive learning. From all the ideas other teachers have shared with us, we have found that adding games, toys, play, and stories to a class can be done fairly easily using a little creativity. Our goal is to combine the techniques discussed so that there is an element of kindergarten in each class. We are pleased to report that we have done this without having to sacrifice topics in the class. We are not spending more time on each topic, we are simply making better use of the allotted time. Not only have our classes been more fun for everyone, but we believe they have wider appeal. The more teaching and learning styles that we incorporate into our classes, the more students will enjoy and succeed in our classes.

8. REFERENCES

[1] R. J. Anderson, R. Anderson, T. VanDeGrift, S. A. Wolfman, and K. Yasuhara. Promoting interaction in large classes with computer-mediated feedback. In *Proceedings of CSCL'03: the International Conference on Computer Support for Collaborative Learning*, pages 119–123, Bergen, Norway, June 2003. Kluwer.

[2] S. K. Andrianoff and D. B. Levine. Role playing in an object-oriented world. In *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, pages 121–125, New York, NY, USA, 2002. ACM Press.

[3] O. Astrachan. Concrete teaching: hooks and props as instructional technology. In *ITiCSE '98: Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education*, pages 21–24, New York, NY, USA, 1998. ACM Press.

[4] A. Begel, D. D. Garcia, and S. A. Wolfman. Kinesthetic learning in the classroom. In *Proceedings of the Technical Symposium on Computer Science Education*, 2004.

[5] C. C. Bonwell and J. A. Eison. Active learning: Creating excitement in the classroom. Technical report, ERIC Clearinghouse on Higher Education, 1991.

[6] J. D. Chase and E. G. Okie. Combining cooperative learning and peer instruction in introductory computer science. In *SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education*, pages 372–376, New York, NY, USA, 2000. ACM Press.

[7] R. M. Felder and R. Brent. Navigating the bumpy road to student-centered instruction. *College Teaching*, 44(2):43–47, 1996.

[8] A. E. Fleury. Acting out algorithms: how and why it works. *The Journal of Computing in Small Colleges*, 13(2):83–90, 1997.

[9] E. Freeman, E. Freeman, B. Bates, and K. Sierra. *Head First Design Patterns*. O'Reilly, 2004.

[10] J. M. D. Hill, C. K. Ray, J. R. S. Blair, and J. Curtis A. Carver. Puzzles and games: addressing different learning styles in teaching operating systems concepts. *SIGCSE Bull.*, 35(1):182–186, 2003.

[11] D. Jacobson, J. A. Davis, and B. Licklider. See one, do one, teach one - two faculty members' path through student-centered learning. In *Proceedings of the IEEE Frontiers in Education*, Nov 1998.

[12] T. Lainema. Implications of constructivism for computer-based learning. In *Proceedings of the European Conference on Information Systems (ECIS)*, 2003.

[13] D. Manolescu, M. Voelter, and J. Noble, editors. *Pattern Languages of Program Design*, volume 5, chapter Active Learning. Addison Wesley Professional, 2006.

[14] J. J. Mcconnell. Active and group learning and their use in graphics education. *Computers and Graphics*, 20(1):177–180, 1996.

[15] C. H. Papadimitriou. Mythematics: storytelling in the teaching of computer science and mathematics. *SIGCSE Bull.*, 35(3):1–1, 2003.

[16] N. Parlante. Nifty assignments. http://nifty.stanford.edu/, 1999–2006.

[17] S. Pollard and J. Forbes. Hands-on labs without computers. In *SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 296–300, New York, NY, USA, 2003. ACM Press.

[18] J. Poon. Java meets teletubbies: an interaction between program codes and physical props. In *ACSE '00: Proceedings of the Australasian conference on Computing education*, pages 195–202, New York, NY, USA, 2000. ACM Press.

[19] M. Resnick. Diving into complexity: Developing probabilistic decentralized thinking through role-playing activities. *Journal of the Learning Sciences*, 7(2), 1997.

[20] M. Resnick. Technologies for lifelong kindergarten. *Educational Technology Research and Development*, 46(4), 1998.

[21] M. Roblyer and R. Schwier. *Integrating Educational Technology into Teaching*. Prentice Hall, 2003.

[22] D. Seuss. *The Cat in the Hat Comes Back*. Random House Books, 1958.

[23] D. Seuss. *The Sneetches and Other Stories*. Random House Books, 1961.

[24] K. Sierra and B. Bates. *Head First Java*. O'Reilly, 2005.

[25] R. E. Slavin. Research on cooperative learning and achievement: What we know, what we need to know. *Contemporary Educational Psychology*, 21(1):43–69, 1996.

[26] D. S. Touretzky. *Common Lisp: A Gentle Introduction to Symbolic Computation*, chapter Martin and the Dragon. Benjamin-Cummings, 1989.

[27] E. Yeoman. Gadgetry, magic, storytelling, and teaching: The computer conference as writerly text. *The Morning Watch: Educational and Social Analysis*, 29(3-4), 2002.