Media Computation as a Motivation and Structure for a Non-Majors CS1 Class: "Data-First" Computing

PI: Mark Guzdial

June 3, 2002

Project Summary

The problem being addressed by this proposal is the disinterest in computer science exhibited by large groups of students, especially non-CS-majors and women—a particular problem at institutions like Georgia Tech where an introductory computing course is required. The approach that we propose is a prototype course in *Introduction to Media Computation* aimed at non-CS-majors. Our argument is that these audiences are most interested in computing to manipulate data of interest to them. We propose a "data-first" approach where we introduce computing in terms of creation, manipulation, and transformation of data of interest to students. At our institution, non-CS-majors are often manipulating multimedia in our English and other communications classes. Literature on gender-issues in computing suggest that women seek an applications-oriented focus to computer science and assignments that lend themselves to creativity. The media computation approach addresses these needs. The base premises for the course are:(a) All media are moving to a digital format; (b) digital media are manipulated using software; (c) learning to control computation, including programming, then becomes a communications skill.

The products of this project will include: A course structure, lecture slides, and course notes in support of such a class; technology to support such a course, including a student development environment, media manipulation tools, and a cross-platform multimedia API appropriate for novice programmers; and evaluation results, focusing on retention, motivation, and learning.

Media Computation as a Motivation and Structure for a Non-Majors CS1 Class: "Data-First" Computing

Contents

Media Computation as a Motivation and Structure for a Non-Majors CS1 Class: "Data-First" Computing

1 Goal: A New Route to Computer Science

Computer science departments are not currently successful at reaching a wide range of students who are taking introductory computer science. The evidence for this statement includes international studies of programming performance [24], declining retention rates [12], and failure rates sometimes as high as 30% [33]. This comes at a time when the need for Information Technology (IT) professionals is growing [8].

At Georgia Institute of Technology ("Georgia Tech"), all students are required to take an introductory course in computing, including programming skills. The current course[1] is undoubtedly one of the most unpopular courses on campus, especially among those *not* in explicitly computing-related fields. While this is certainly a problem for the College of Computing at Georgia Tech (where the course has its academic home), it points towards a larger problem for the field. Alan Perlis in April 1961 made perhaps the first argument that programming should be part of a liberal education for *all* students. If Calculus is the study of *rates*, and that's important enough to be part of the liberal education, then so should computer science. Perlis argued that computer science is the study of *processes*, which is certainly relevant to even more fields than those concerned with rates. The argument has been echoed and strengthened over the intervening years—by Seymour Papert arguing for a programming as a way of learning about learning [27][28], to Andrea diSessa's arguments

[1]A second course for Engineering students only is at a prototype stage.

for “computational literacy” as a critical component of many fields [5]. As long as non-CS-majors have such a dislike for computing, the hope is diminished for computer science as an accepted part of a liberal education and for computing generally to meet its potential for intellectual impact across the range of disciplines, not just in computational science and engineering.

Introductory computer science classes are also not meeting the needs of women. The rates at which women take computer science classes are falling. While the factors causing women to avoid computer science (and IT careers in general) are multi-faceted, the curriculum does play a significant role [22]. A report in 2000 by the American Association of University Women [1] suggested that part of the problem, at least for women, is that computer science courses are, frankly, too boring. Specifically, they claim that computer science courses are “overly technical” with little room for “tinkering.” At a session on increasing enrollment of women in computer science at the latest ACM SIGCSE conference, speakers reported that women who pursued computer science degrees were surprised at how much “creativity” there was in later computer science courses—introductory courses did not highlight that aspect of CS [31]. Additionally, women undergraduate CS students tend to be interested in real applications for the computing, as opposed to simply computing for its own sake [22][1]. If we can address the reasons why women are avoiding computer science, we may be able to interest more males, too—all those currently expressing disinterest in computing. “Women in engineering programs are kind of like ‘canaries in the coal mine”’, said Stephen W. Director, Chair of the Engineering Dean’s council. “If women do well in a program, most likely everyone else will also do well in the same type of program." [2]

The focus of this proposal is **to use multimedia construction projects as the domain of assignments and lectures in an introductory computer science course, explicitly aimed at non-CS-majors, to engage students in computer science**, *Introduction to Media Computation.* The hypotheses of this effort are that this course will demonstrate: (1)Improved student retention, (2) better student attitudes toward computer science, and (3) better student learning of computer science.

1.1 Overview of the Plan and its Rationale

Undergraduate computer science courses tend to emphasize the processing of any kind of data at all. In a sense, it's *data agnostic*—all data should be treated exactly the same. In this way, focus can be shifted to abstracted data representations laid on top of the data.

However, most non-CS-majors come to computers because *they want some processing of data of interest.* The focus for them is on their data. By paying attention to the data that the students care about, we may be able to increase their motivation for learning programming. In the terms of the ACM/IEEE Computing Curriculum 2001[3], this approach is "data-first"—we start from the data that students care about, then introduce computing as a way of creating, manipulating, and transforming the data that they care about.

The choice of multimedia as the data of choice in this project allows us to address several

[2]Testimony by Stephen W. Director, Chair, Engineering Deans Council, American Society of Engineering Education, to the Commission on the Advancement of Women and Minorities in Science, Engineering, and Technology Development, Washington, DC. July 20, 1999

[3]`http://www.acm.org/sigcse/cc2001`

of the issues identified as critical for increasing the number of women in computing. Several of the academic units at Georgia Tech emphasize that multimedia literacy is a critical component of educated professionals in the future [36]. Multimedia is clearly a real applications domain—especially true at Georgia Tech, with Turner Broadcasting, CNN, and the Cartoon Network based within a few miles of campus. Multimedia is clearly not computing for its own sake. Further, the multimedia approach allows us to give students assignments that are open-ended and creative. For example, if the students need to implement a particular sound algorithm or a particular Photoshop-like image manipulation, we don't need to care what sound or image is used. The emphasis on multimedia allows the class to be more concrete, more relevant, and more creative than traditional CS1 approaches.

The premises and core concepts of the proposed course are:

- All media are being published today in a digital format.
- Digital formats are amenable to manipulation, creation, analysis, and transformation by computer. Text can be interpreted, numbers can be transformed into graphs, video images can be merged, and sounds can be created. We call these activities *media computation.*
- Software is the tool for manipulating digital media. Knowing how to program thus becomes a communications skill. If someone wants to say something that her tools do not support, knowing how to program affords the creation of the desired statement.
- Core computer science concepts can be introduced through media computation. For example, programs can get large and cumbersome. Abstraction is our tool for managing

program complexity and allowing programs to become even larger yet more flexible.

- However, computing has limitations. There are some programs that cannot complete in our lifetime, and knowing that these limitations exist is important for technological professionals.

As is discussed further below, the course is already being developed at Georgia Tech in a collaborative design process with students and faculty from outside of computer science[4]. Both student and faculty feedback has been very positive:

I'm very enthusiastic about your proposal, and know (others in my department) would be too. We very much need *this kind of class. I think the structure of the proposed class is really inspired and is exactly the right approach for our students.* **(English professor)**

*The proposed course is definitely a motivator, since the current requirement does little but get in the way of the courses for the major. Integrating the required computer education credit with an area that sufficiently yields to the material of the major/field would be enormously beneficial. I am pleased that GA Tech is responding to students concerns and allowing those of us not majoring in CS to take a more creative based course that will be beneficial instead of a chore.***(Architecture student)**

I think this is a good idea and I wished I'd had the opportunity to participate in this instead of (the current CS class). **(History student)**.

The one problem that I am worried about if this class were to be added to the curriculum is the amount of spots open during registration. Everyone I know dreads taking the CS courses that are available now, and they'd jump at the chance to take this so it would fill up very quickly. **(Chemistry student)**

[4]`http://coweb.cc.gatech.edu/mediaComp-plan`

2 Project Plan

The prototype course is already in development at Georgia Tech. In this section, I describe the current plans, the technology that we are developing to support the course, and our development process. The work proposed to be funded by this grant is to focus on the evaluation of the prototype and the dissemination of the prototype materials.

2.1 Developing a Prototype

The prototype course is scheduled to be taught in the Spring 2003 semester to no more than 100 students[5]. The emphasis will be on non-CS-majors and non-Engineering majors. The plan is to develop the prototype course during the Summer of 2002, and then to gain approval for the course in Fall 2002 including meeting the Institute computing requirement. The plan over the course of the summer is to develop: A set of course slides and course plan (e.g., lab activity descriptions, student assignment descriptions); a set of course notes; and a set of technologies to be used in the course. During the Fall, the specific labs and assignments will be developed for the initial offering.

2.1.1 Curriculum Development

The semester outline of the course appears in Table 1. The course is currently planned to be three credit hour with a weekly lab activity with optional recitations[6].

[5]If successful, the course will be offered starting in Fall 2003 with multiple sections of 200 or more

[6]The curriculum development can be viewed at `http://coweb.cc.gatech.edu/mediaComp-plan/Curriculum`

Week 1 Introduction: What is Computer Science and Media Computation Variables and functions
Week 2 Sound as an array of samples Loops for manipulating samples
Week 3 How sound works and how it can be manipulated Increasing/decreasing volume, trimming sounds, creating reverb
Week 4 Developing a mental model of the program: Debugging Images as a two-dimensional array of pixels
Week 5 Manipulating images by changing Alpha and RGB values Filtering images using conditionals (for thresholding functions)
Week 6 Manipulating a portion of an image: Masks and varying the loop endpoint Drawing on an image: Graphics on the image
Week 7 Developing a mental model of the program: Tracing conditionals and loops Manipulating the files that the media live in
Week 8 Writing scripts that move and process files Video: A series of images/frames in files
Week 9 Applying image techniques to video frames "Why is this taking so long?!?": An introduction to algorithm complexity
Week 10 Text as a media type: Manipulating text with programs Making other programs do the work: Programs that process data for other programs
Week 11 Graphing data: Media conversion from text to graphics Graphing data with an external program: Using Excel and GnuPlot
Week 12 "Can't we do this any easier?": Functional decomposition to reduce program complexity "Can't we do this any easier?": Recursion to traverse data
Week 13 "Can't we do this any easier?": Objects as a technique to manage complexity
Week 14 Applying these techniques to media manipulation Thinking about languages and representations for process: What computer scientists do
Week 15 Introduction to Java Java for Media Manipulation

Table 1: Outline of the prototype course

The current plan is to use the programming language *Jython* for the course. Jython[7] is a variation of the Python[8] programming language which has been designed to be easily used by novices and non-technical users. Jython is a variation of Python written in Java. Jython can use Java classes, and Jython can be used for virtually anything for which Java can be used. We considered other languages (e.g., Java and Scheme), but selected Jython based on survey responses from students and teachers. Jython's design is based on the research findings on studying novice programmers (e.g., [34], [25], [26]).

2.1.2 Technology Development

Three technology development efforts are on-going to support this class[9].

There is no Jython development environment well-suited for students. We (the PI and a group of undergraduate student researchers and developers) are building one now. The PI has previous research experience developing novice programming environments [9][15]. Our plan is to develop a simple system and then develop scaffolding over time to respond to student needs [37].

While Java has the base multimedia support needed for this course, the API for manipulating media is complex. We have initial versions now of *Picture* and *Sound* classes that draw together the complex Java API and provide a simple interface for student programmers in Jython.

Finally, we are creating a cross-platform set of *MediaTools* that allow students to high-

[7] `http://www.jython.org`

[8] `http://www.python.org`

[9] The technology development can be viewed at `http://coweb.cc.gatech.edu/mediaComp-plan/Technology`

level media manipulations, "debug" their media, and prepare media for programming. These functions include: Recording a sound; viewing a sound; and converting an MPEG movie to a series of JPEG frames. A cross-platform prototype (full-functionality) of the MediaTools currently exists, and is being iteratively improved this summer.

We plan to use our existing collaboration technologies in the media computation course to support student sharing of their media artifacts in the introduction to media computation course. The *constructionist* theory of learning suggests that the creation of public artifacts creates a strong potential for learning [29]. We have had good success in the past using our collaboration tools to support sharing of multimedia artifacts to encourage critique and motivate learning [20][4][39].

2.1.3 Involving Faculty and Students

Much of the PI's research is in the field of computer-supported collaborative learning (e.g., [19] [17] [18]). To support the development of this course, the PI is using the tool developed by his group, the CoWeb, to create a collaborative setting for the development of the course, `http://coweb.cc.gatech.edu/mediaComp-plan`.

The planning CoWeb is particularly useful for involving faculty and students in the development of the course. Undergraduate students working with the PI on technology are using the space for coordination, sharing code, and sharing plans. Undergraduate non-CS-majors are answering surveys in the CoWeb to provide student feedback on the course plans. An advisory panel of faculty (from outside of CS) are leaving their comments on the course development in the CoWeb.

2.1.4 Experience of PI and Results from Prior NSF Funding

The PI's most relevant recent NSF funding was REC-9814770 with co-PI's Matthew Realff and Pete Ludovice from the School of Chemical Engineering and Tom Morley from the School of Mathematics. The goal of this project was to explore the use of collaborative technologies to create learning opportunities that cross curricular boundaries. The project resulted in the development of the CoWeb and its surprisingly high adoption rate among faculty, both at Georgia Tech and beyond [17]. However, we were unsuccessful at developing much use among the Engineering students we were targeting, in part due to cultural and infrastructure issues [16]. We did, however, demonstrate significant learning in an English class [32] and develop some new theoretical positions on the sources of learning in collaborative environments [13].

As the focus shifted from student participation to faculty outreach, an extension was granted to focus on helping faculty to integrate collaboration and even some of these media computation ideas into their CS classes. A three day workshop was held May 3-6, 2002, at Georgia Tech, with some 50 participants. The materials from that workshop are available at `http://coweb.cc.gatech.edu/mm-csed`[10]

2.2 Evaluation Development

This proposal seeks funding for the evaluation and dissemination of the project. The detailed evaluation plan will be developed during the Fall 2002. The analysis of the data will take place during Summer 2003.

[10]A follow-on half-day workshop is being proposed for the ACM SIGCSE 2003 conference.

The evaluation effort has three hypotheses: (1) the prototype course will improve student retention, especially among non-CS-majors and women; (2) the prototype course will improve student motivation toward computer science; and (3) the prototype course will lead to good student learning, perhaps better than in a comparable course.

Our evaluation effort will be conducted under the review of the Georgia Tech Human Subjects Review Board, to whom this proposal and all our instruments will be submitted as they are developed. Students will be informed about the research project, the instruments, how they will be impacted, and potential risks.

For all our evaluations, we will use as a comparison class our existing CS1 course for all majors. The prototype offering of the course will be limited to 100 students. Approximately 1200 students take the existing CS1 course each semester, so there is sure to be a comparable sample of students for our analyses.

The PI will be leading the evaluation with advice from the Director of Georgia Tech's Center for Enhancement of Teaching and Learning (CETL), Dr. Donna Llewellyn. Dr. Llewellyn and the PI worked together on studies of the current course, including a survey addressing issues of motivation, during the Spring 2002 semester[11]. While the PI clearly cannot be as objective as an external evaluator, it is not unusual and sometimes even preferable for the PI to be the lead evaluator for a *formative* evaluation such as this [7][6]. Further, the PI is known in his research community for an unusual objectivity in his evaluations. He defines metrics for measuring effectiveness and publishes the results of applying those metrics to his own systems, even when the results are negative [10][16].

[11]A paper is planned on these studies for ACM SIGCSE 2003

2.2.1 Evaluation of Retention

Retention rate is defined operationally in this study in two ways.

- The *completion retention rate* will be defined as the ratio of the number of students completing the course to the number of students enrolled in the course;
- The *course success retention rate*will be defined as the ratio of the number of students completing the course with a C or better to the number of students who enrolled in the course. (Several academic units at Georgia Tech require students to re-take courses if they received a D.)

We will measure these retention rates overall, and with women and non-CS-majors separately.

The literature on both rates is rather negative. Completion rates of less than 30% is not uncommon in CS1 [33]. Course success retention rates of less than 50% have been reported at several schools [12]. Our existing class does remarkably well in comparison with the literature, with a completion retention rate of 80-90% and a course success retention rate better than 70%, so it will offer a high standard to meet.

2.2.2 Evaluation of Motivation

The stated goal for the prototype course is to address the disinterest in computer science, especially among non-CS-majors and women. We are concerned with three kinds of motivational factors:

- Do students find the current course engaging? An engaging course is more likely to lead to learning [21] [30] [3].

- Do students find computer science engaging? For example, would students agree with the statements "Programming a computer is an interesting activity" or "Thinking about data representations is intriguing"? We hypothesize that students who find computer science engaging are probably more likely to use computing in their professional lives, though we have no empirical evidence yet to support that claim.

- Would students consider taking future computer science courses? While the goal of the prototype course is to simply address issues of disinterest, positive response to this question would suggest that the media computation ("data-first") approach has potential for attracting potential CS majors[12].

The plan is to develop a survey instrument similar to the one that was used in Spring 2002 in the comparison class. While the new instrument will be used in both the prototype and comparison classes in Spring 2003, the common questions with Spring 2002 will allow us to compare across time and even to see how reliable the motivation factors are in the comparison class.

2.2.3 Learning

Our plan for evaluating learning has two components. The first is to have some directly comparable final exam questions between the comparison class and the prototype class, to measure learning of CS concepts. These may be identical questions or *isomorphic*[13] questions,

[12]We would like to follow-up to see how many students completing the prototype course *do* go on to take more computing courses, or even changing majors to CS, but that's beyond the proposed one year course.

[13]Not identical, but very similar, e.g., where only constants are changed

which we have had good success with in the past [14] [11] and has been used by others as well [2]. Our hypothesis will be confirmed if there is *no* statistically significant difference between the comparison and prototype classes, when comparing against students in the comparison class who are not CS-majors.

The second strand is to ask students in the prototype class to attempt a program (possibly a variation, perhaps isomorphic, in order to fit within the media computation context) whose results with CS1 students has been published, to measure learning of programming skills. There are several published benchmarks for performance in CS1, including Soloway's Rainfall Problem [35] and McCracken's Calculator Problem [24]. The results from this second strand will be quite intriguing, since there are several non-major CS1 courses described in the published literature (e.g., [38], [23]), but there are virtually no published papers demonstrating programming skills in non-major CS1 courses.

3 Conclusion: Deliverables and Dissemination

At the end of the funding period, several resources will be available publicly to others who would like to use this method at their own institutions, such as the lecture slides and technologies, as well as the course planning website, where the rationale for the course decisions is made explicit, and the assessment instruments and evaluation results. All of these materials will be made available at the development website and at the course CoWeb when it is constructed. These will be open websites.

To disseminate these materials and findings, we propose three strategies:

- We plan to advertise the materials to the mailing list set up for the workshop (mentioned earlier) of CS faculty already expressing interest in multimedia and collaboration in their courses[14]. We will advertise the materials with the findings to the ACM SIGCSE members mailing list.

- We will submit a proposal to ACM SIGCSE 2003 for a faculty poster on the materials developed for this course. Since ACM SIGCSE 2003 will be in February, we will not have any evaluation results available yet.
- We will also submit a paper proposal to ACM-sponsored ITICSE 2003 (Innovation and Technology in Computer Science Education) which will be held in Summer 2003. There we will be able to present evaluation results.
- We will also propose an IEEE FIE 2003 paper for November 2003, where evaluation results can be disseminated to the more engineering-oriented CS education community.

[14] `mailto:mm-csed@cc.gatech.edu`

4 Budget Justification

To fund this effort, this grant includes a budget request for a one year project.

- One summer month of the PI's time to direct evaluation and write up results for dissemination,
- One graduate student research assistant to help design, implement, and analyze the evaluation effort,
- Our costs for these personnel include fringe, computing charges (for support within the College of Computing), and graduate student tuition,
- Travel support the PI and graduate student to attend ITICSE and FIE ($4500),
- Materials and supplies, both to support the development and evaluation effort, but also to explore the media computation directions that faculty suggest to us ($4000). As we have developed the course thus far, faculty from many departments have suggested to us "I want my students to understand that concept in terms of application X which we use often in my field!" We are trying to collect instances of those applications X (or texts about them, in some cases) to understand the issues and how to integrate them into our course.

5 Facilities

The College of Computing maintains a variety of computer systems in support of academic and research activities. These include more than 50 Sun, Silicon Graphics, and Intel systems used as file and compute servers. In addition, there are more than 1,000 workstation class machines from Sun, Silicon Graphics, Intel, and Apple especially for student use.

The Graphics, Visualization, and Usability (GVU) Center houses a variety of graphics and multimedia equipment, including high-performance systems from Silicon Graphics, Sun, Intel, and Apple.

PI Guzdial is the Director of the Collaborative Software Lab, affiliated with GVU. The Collaborative Software Lab has a bank of ten servers supporting our experimental software for studying computer-supported collaborative learning. In addition, we have three Linux workstations, two NT workstations, and two Apple workstations used for development. The focus of the Collaborative Software Lab is on facilitating multimedia collaboration, so multimedia facilities available include a high-end Alesis keyboard, projection facilities, a Canon digital video camera, and a Sony Mavica digital camera.

References

[1] AAUW. *Tech-Savvy: Educating Girls in the New Computer Age.* American Association of University Women Education Foundation, New York, 2000.

[2] Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences 4*, 2 (1995), 167–208.

[3] Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., and Palincsar, A. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist 26*, 3 & 4 (1991), 369–398.

[4] Craig, D., ul Haq, S., Khan, S., Zimring, C., Kehoe, C., Rick, J., and Guzdial, M. Using an unstructured collaboration tool to support peer interaction in large college classes. In *International Conference of the Learning Sciences 2000.* Ann Arbor, MI, 2000, pp. 178–184.

[5] diSessa, A. *Changing Minds.* MIT Press, Cambridge, MA, 2001.

[6] Egan, D., Remde, J., Gomez, L., Landauer, T., Eberhardt, J., and Lochbaum, C. Formative design-evaluation of superbook. *ACM Transactions on Office Information Systems 7* (1990), 30–57. Uses critical incident analysis, log file analysis.

[7] Flagg, B. N. *Formative Evaluation for Educational Technologies.* Communication. Erlbaum and Associates, Hillsdale, NJ, 1990.

[8] Freeman, P., and Aspray, W. *The Supply of Information Technology Workers in the United States.* Computing Research Association, New York, 1999.

[9] Guzdial, M. Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments 4*, 1 (1995), 1–44.

[10] Guzdial, M. Information ecology of collaborations in educational settings: Influence of tool. In *Proceedings of Computer-Supported Collaborative Learning'97*, R. Hall, N. Miyake, and N. Enyedy, Eds. LEA, Toronto, Ontario, Canada, 1997, pp. 83–90.

[11] Guzdial, M. Using squeak for teaching user interface software. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education.* ACM Press, New York, 2001, pp. 219–223.

[12] Guzdial, M. Summary: Retention rates in cs vs. institution. Message posted on acm sigcse moderated members list, Georgia Tech, April 23 2002.

[13] Guzdial, M., and Carroll, K. Exploring the lack of dialogue in computer-supported collaborative learning. In *Proceedings of the 2002 Computer-Supported Collaborative Learning Conference*, G. Stahl, Ed. University of Colorado at Boulder, Boulder, CO, 2002, pp. 418–424.

[14] Guzdial, M., and Kehoe, C. Apprenticeship-based learning environments: A principled approach to providing software-realized scaffolding through hypermedia. *Journal of Interactive Learning Research 9*, 3/4 (1998), 289–336.

[15] Guzdial, M., Konneman, M., Walton, C., Hohmann, L., and Soloway, E. Layering scaffolding and cad on an integrated workbench: An effective design approach for project-based learning support. *Interactive Learning Environments 6*, 1/2 (1998), 143–179.

[16] Guzdial, M., Ludovice, P., Realff, M., Morley, T., Carroll, K., and Ladak, A. The challenge of collaborative learning in engineering and math. In *Proceedings of IEEE/ASEE Frontiers in Education (FIE) 2001 Conference*. IEEE, Reno, NV, 2001.

[17] Guzdial, M., Rick, J., and Kehoe, C. Beyond adoption to invention: Teacher-created collaborative activities in higher education. *Journal of the Learning Sciences 10*, 3 (2001), 265–279.

[18] Guzdial, M., Rick, J., and Kerimbaev, B. Recognizing and supporting roles in cscw. In *Proceedings of CSCW'2000*. ACM Press, New York, 2000, pp. 261–268.

[19] Guzdial, M., and Turns, J. Effective discussion through a computer-mediated anchored forum. *Journal of the Learning Sciences 9*, 4 (2000), 437–470.

[20] Kehoe, C. M. *Supporting Critical Design Dialog*. Unpublished ph.d. dissertation, Georgia Institute of Technology, 2001.

[21] Malone, T., and Lepper, M. Making learning fun: A taxonomy of intrinsic motivations for learning. In *Aptitude, Learning, and Instruction.*, R. Snow and M. Farr, Eds., vol. 3 of *Conative and Affective Process Analyses*. LEA, Hillsdale, NJ, 1987, pp. 223–253.

[22] Margolis, J., and Fisher, A. *Unlocking the Clubhouse: Women in Computing*. MIT Press, Cambridge, MA, 2002.

[23] Marks, J., Freeman, W., and Leitner, H. Teaching applied computing without programming: A case-based introductory course for general education. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 80–84.

[24] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *ACM SIGCSE Bulletin 33*, 4 (2001), 125–140.

[25] Miller, L. A. Natural language programming: Styles, strategies, and contrasts. *IBM Systems Journal 20*, 2 (1981), 184–215. Languages require iteration where aggregate operations are much easier for novices.

[26] Pane, J. F., Ratanamahatana, C., and Myers, B. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies 54* (2001), 237–264.

[27] Papert, S. Teaching children to be mathematicians versus teaching about mathematics. Ai memo no. 249 and logo memo no. 4, MIT, 1971.

[28] Papert, S. *Mindstorms: Children, computers, and powerful ideas.* Basic Books, New York, NY, 1980.

[29] Papert, S. Situating constructionism. In *Constructionism*, I. Harel and S. Papert, Eds. Ablex Publishing Company, Norwood, NJ, 1991, pp. 1–11.

[30] Paris, S. G., and Turner, J. C. Situated motivation. In *Student Motivation, Cognition, and Learning: Essays in Honor of Wilbert J. McKeachie*, P. Pintrich, D. Brown, and C. Weinstein, Eds. Erlbaum, Hillsdale, NJ, 1994, pp. 213–237.

[31] Pfleeger, S. L., Teller, P., Castaneda, S. E., Wilson, M., and Lindley, R. Increasing the enrollment of women in computer science. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 386–387.

[32] Rick, J., Guzdial, M., Carroll, K., Holloway-Attaway, L., and Walker, B. Collaborative learning at low cost: Coweb use in english composition. In *Proceedings of the Computer Supported Collaborative Learning 2002.* Lawrence Erlbaum Associates, Mahwah, NJ, 2002, pp. 435–442.

[33] Roumani, H. Design guidelines for the lab component of objects-first cs1. In *The Proceedings of the Thirty-third SIGCSE Technical Symposium on Computer Science Education, 2002*, D. Knox, Ed. ACM, New York, 2002, pp. 222–226. WFD (Withdrawl-Failure-D) rates in CS1 in excess of 30

[34] Soloway, E., Bonar, J., and Ehrlich, K. Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM 26*, 11 (1983), 853–860.

[35] Soloway, E., Ehrlich, K., Bonar, J., and Greenspan, J. What do novices know about programming? In *Directions in Human-Computer Interaction*, A. Badre and B. Schneiderman, Eds. Ablex Publishing, Norwood, NJ, 1982, pp. 87–122.

[36] Soloway, E., Guzdial, M., and Hay, K. E. Reading and writing in the 21st century. *EDUCOM Review 28*, 1 (1993), 26–28.

[37] Soloway, E., Guzdial, M., and Hay, K. E. Learner-centered design: The challenge for hci in the 21st century. *Interactions 1*, 2 (1994), 36–48.

[38] Zimmerman, G. W., and Eber, D. E. When worlds collide! an interdisciplinary course in virtual-reality art. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM Press, New York, 2001, pp. 75–79.

[39] Zimring, C., Khan, S., Craig, D., Haq, S.-u., and Guzdial, M. Cool studio: Using simple tools to expand the discursive space of the design studio. In *Design Thinking Research Symposium*. MIT, Cambridge, MA, 1999.