Using Media as a Context to Motivate Women and Non-CS Majors in Computer Science

PI: Mark Guzdial

January 16, 2004

Project Summary

Computer science educators have a problem in motivating and engaging students, especially non-CS-majors and women. These students are withdrawing or failing introductory computing courses at an alarming rate, or simply avoiding computing entirely. Women make up less than 20% of the undergraduate CS majors in the U.S. today, and it's been falling.

Based on the research on what makes women successful in computer science, we have developed an approach based on teaching computation as a communications skills. We have had a successful trial offering of an undergraduate course in *Introduction to Media Computation* (in the programming language Python) aimed at non-CS majors. By conventional assessment, such as retention and WFD rates, the course was quite successful at its goals. 121 students enrolled—2/3 women. 89% of the class earned an A, B, or C in the course. 60% of the respondents on a final survey indicated that they would like to take a second course on the topic.

We propose to build upon this success in three directions:

- We are working to extend the approach to a second course and develop materials in Java to support CS major courses that might use media to motivate students. We hope that these two courses and our use of these in degrees will become a national model for constructing alternative paths into computer science.
- We plan to improve and extend our evaluation, to understand better any changes in attitudes towards computer science, what learning is occurring, and if there is a longitudinal impact on students attitudes toward computation.
- Finally, we are teaming with other schools to further develop the use of media as a context in computing. We plan to hold an annual meeting to share materials, methods, and data and to develop joint projects where we can actively explore developments in multiple school contexts at once.

Intellectual Merit: This proposal builds upon existing research to implement an alternative path through introductory computer science for women and non-CS majors. *Broader Impacts*: The potential broader impacts include developing better methods to address the lack of women, and in doing so, creating a broader-base of computing-literate citizens.

Using Media as a Context to Motivate Women and Non-CS Majors in Computer Science

Contents

Using Media as a Context to Motivate Women and Non-CS Majors in Computer Science

1 Goal: Assessing New Routes to Computer Science

Computer science departments are not currently successful at engaging a wide range of students who are taking introductory computer science. The evidence for this statement includes international studies of programming performance [27], declining retention rates [15], and failure rates sometimes as high as 50% [37]. This comes at a time when the need for Information Technology (IT) professionals is growing [12].

At Georgia Institute of Technology ("Georgia Tech"), all students are required to take an introductory course in computing, including programming skills. Our traditional CS1 course, the only one that met the requirement before the Spring 2003 semester, 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 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 *process*, 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 [32][33], to Andrea diSessa's arguments for "computational literacy" as a critical component of many fields [10]. 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 particularly not meeting the needs of many 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 [25]. 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 [35]. Additionally, women undergraduate CS students tend to be interested in real applications of computing, as opposed to simply computing for its own sake [25][1]. If we can address the reasons why women are avoiding computer science, we may be able to interest more males, too—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." [1]

There are new, innovative efforts to address these issues. Most well-known among these is the work at Carnegie-Mellon University (CMU) which was able to raise dramatically the percentage of women in their undergraduate CS program [25]. New courses are being developed, such as an Alice-based pre-CS1 course [9], whose goal is to meet some of the problems identified as preventing women and non-CS majors from succeeding at Computing. The Alice-based course is succeeding at having higher retention rates than other introductory computing courses [9]. Other innovative new courses include a focus on graphics [26], virtual reality [42], and Web programming (e.g., servlets) [21]. New approaches within existing classes, like pair-programming methods, are leading to improved retention in the class and within the CS major [28, 30].

Our approach builds on the lessons learned in others' work, and particularly focuses on (a) providing relevance, (b) creating a social context, and (c) encouraging creativity. We at Georgia Tech have developed a course, *Introduction to Media Computation*, to meet the needs of women and non-CS majors. Students learned how media are manipulated computationally for communications, often with open-ended assignments that allowed them to be creative, and then shared their creations in a collaborative webspace.

The course was offered as a pilot in Spring 2003 with 121 students, 2/3 women, with *none* of the students majoring in CS or Engineering. These students were from the College of Architecture, Ivan Allen College of the Liberal Arts, Dupree College of Management, and the School of Biology. The hypotheses of this effort were that this course would demonstrate: (1) Improved student retention, (2) better student attitudes toward computer science, and (3) better student learning of computer science.

The course was quite successful at the goal of improving student retention. 121 students enrolled – 2/3 women. 89% of the class earned an A, B, or C in the course, i.e., an 11% Withdrawal-Failure-or-D (WFD) rate. We are less certain of the impact on attitudes and learning. 60% of the respondents on a final survey indicated that they would like to take a second course on the topic–but at the same time, only 9% claimed that they would ever take another course in computer science! They did learn enough programming to successfully write original programs. Students wrote eight programs (six collaboratively and two individually) creating or manipulating pictures, sounds, HTML pages, and movies, with some of their programs reaching 100 lines of code. When we attempted to use isomorphic exam problems (as in [17]) to compare students between the classes, we ran into difficulties with course sequencing and language differences. In our second offering of the course (Fall 2003), we had 305 students, again 2/3 women, with a 13% WFD rate. Only six students dropped the course.

We do not believe that media computation is the *only* context in which students not traditionally attracted to computer science might succeed. There are certainly others. For

[1] Testimony by Stephen W. Director, Chair, Engineering Dean's 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

example, we have described elsewhere another potential CS1 course organized around information management in which students might build Web harvesting programs and data visualizations [20]. The general approach we are exploring is to use an application domain of information technology that is rich in CS concepts and relevant to the students, then explore introducing computing concepts in terms of that domain. However, we do believe that media computation is a *viable* alternative path, and through the proposed work, we plan to apply this approach more broadly and study it more carefully.

In this proposal, we propose to expand the media computation approach into a viable alternative path to CS and CS learning; to study more carefully what's going on in this approach; and to create a context for extending this approach beyond Georgia Tech into a national model. In particular, we propose:

- We plan to develop materials to create a complete path into CS (major or minor) from a media computation perspective. We are working to extend the approach to a second CS course (on data structures and design methods, per CS curriculum guidelines [2]. We are already developing materials in Java (the programming language that now predominates introductory computing courses) to support more traditional CS major courses that might use media to motivate students. We plan to use our two courses and materials at Georgia Tech to construct a minor option for non-CS majors and also to construct a path into a CS major through media computation. We hope that these two courses and our use of these in degrees will become a national model for constructing alternative paths into computer science.

- We plan to improve and extend our evaluation, to understand better any changes in student attitudes towards computer science, what learning is occurring, and if there is a longitudinal impact on students attitudes toward computation.

- Finally, we are teaming with other schools (a two year college, a four year college, and a private research university) to further develop the use of media as a context in computing. We plan to hold an annual meeting to share materials, methods, and data and to develop joint projects where we can actively explore developments in multiple school contexts at once.

2 Results from Proof-of-Concept Trial

Introduction to Media Computation (CS1315 at Georgia Tech) is an introduction to computation and programming contextualized around creation and manipulation of media (funded by proof-of-concept NSF CCLI grant #0231176). The explicit goal is to interest groups typically disinterested in such courses (especially women and non-CS majors). The course was offered as a pilot in Spring 2003.

The course was a success on several measures.

- 89% of the students who enrolled earned an A, B, or C. Our "best-practice" point of comparison in the literature had a 66.4% success rate for non-majors in a specialized CS1 [30].

- The course may have met our goal of getting female students to find CS interesting, relevant, and creative where they hadn't previously, but as we'll explain, our data has some contradictions in it yet.
- We attempted direct comparisons of student performance on similar exam problems given in our traditional CS1, a prototype Engineering-focussed CS1 course, and the Media Computation class. We found that such comparisons are fraught with difficulties that rendered the results inconclusive. Nonetheless, student performance in a problem described in the literature on CS education was relatively comparable between the Media Computation and traditional CS1 students.

Our assessment of the course, and the comparison with the other courses, went beyond measures of success to explore how and why the class was successful. For example, collaboration may have played as large a role in the success of CS1315 as the context. When asked the open question on the final survey, "What absolutely must *not* change about the course?", nearly 20% of the Media Computation student respondents named the collaboration tool CoWeb [14] [18] and over 20% mentioned collaboration in general. This is significant because we know that the social context of computer science is an important inhibitor to success in CS for many students [1] [25].

2.1 Implementation of the Course

The choice of multimedia as the course context in this project allows us to address several of the issues identified as critical for increasing the number of women in computing, such as the lack of relevance in traditional computer science classes [1, 25], the lack of a creative component in introductory computing classes [35], and the tediousness of traditional problems in CS classes [1]. Several of the academic units at Georgia Tech emphasize that multimedia literacy is a critical skill for educated professionals in the future [40]. Students recognize that multimedia is a significant applications domain, with real career paths. 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. Overall, 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 Media Computation course start with all media are being published today in a digital format. Digital formats are amenable to manipulation, creation, analysis, and transformation by computer. We call these activities *media computation*. Software is the tool for manipulating digital media. Knowing how to program thus becomes a communications skill. 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.

We used the programming language *Jython* for the course. Jython[2] is a variation of the Python[3] 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, which allowed ease of access to Java's Media API. We considered other languages (e.g., Java and Scheme), but selected Jython based both on survey responses from students and teachers [16] and because Jython's design is grounded in the research findings on facilitating learning and performance by novice programmers (e.g., [38], [29], [31]).We developed *JES* (Jython Environment for Students), modelling it after the popular DrScheme environment [11].

The general structure of the course was 11 weeks of media-relevant material (e.g., sound, pictures, movies, text manipulation, network, and database), with four weeks of computer science that answers questions that arose during the course of the 11 weeks. Student homework assignments started with the implementation of simple Photoshop-style filters (average score 92% with a standard deviation of 22.1). Some of the assignments invited creativity, such as the third homework which required creation of a collage, but allowed students to select whatever pictures they wanted in the collage (average score 86.3% standard deviation 26.0). These collage programs became large–the average number of lines of code (of students who gave us consent to look at their homework results) was 64 with a standard deviation of 32.4. The largest was 166 lines of code. The final homework assignment involved programmed Web access and creation of animations (average score 84.6% standard deviation 29.2). Final exam questions spanned a wide range of computing concepts, including issues of networking, databases, and computing theory.

We used our existing collaboration technology, CoWeb [19, 14][4], to support student question asking and sharing of media artifacts. The *constructionist* theory of learning suggests that the creation of public artifacts creates a strong potential for learning [34]. We have had good success in the past using our collaboration tools to support sharing of multimedia artifacts to encourage critique and motivate learning [23][6][43]. The CoWeb became the central place for question-asking and discussion, which is a critical role in CS1, since comfort in asking questions has been highly correlated with attrition in CS1 [41].

2.2 Evaluation Methods and Results

Our evaluation effort had 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 was conducted under the review of the Georgia Tech Human Subjects Review Board, to whom our proposal and all our instruments were submitted for review as they were developed. Students were informed about the research project, the instruments, how they might be impacted, and potential risks. Assessment was organized such that the

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

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

[4]Development and assessment funded in part by NSF grants from REPP and ROLE programs

PI and the assessment team[5] designed the assessment, but the assessment team gathered and analyzed all data apart from the PI, since he was also the teacher of the course.

We gathered and analyzed several sources of data.

- We collected demographic data, which we used to address the retention question. We retained copies of course work from consenting students.
- We took surveys during the first week of the term, midway through the term, and in the last week of the term. We used the same surveys in two other classes: A section of our traditional CS1 course ($n = 127$), and a section of the prototype Engineering-oriented CS1 ($n = 75$). From these surveys we learned important information such as the significantly lower prior background in computer science and programming among the Media Computation students than the other students.
- We attempted to develop and distribute polymorphic problems (same problem, but changed in language or other minor details for the course context, a method we used successfully in [17]) that were offered in all three CS1's on exams–for example, on computing an average test grade and determining a corresponding letter grade, and on the infamous Rainfall Problem [39].
- We interviewed a sample of female students in the *Media Computation* class on their attitudes toward computer science and the course. We conducted seven interviews in total–three early in the semester, and three at the end of the term with the same students, along with another student at the end.

2.2.1 Evaluation of Retention

Retention rate is defined operationally in this study in two ways. The *completion retention rate* is 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* is 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.)

The literature on both rates for introductory CS courses is rather negative. Completion rates of less than 30% are not uncommon in CS1 [37]. In an email survey on the ACM SIGCSE members list, course success retention rates of less than 50% were reported at several schools [15]. The "best-practice" comparison that we have for non-majors, in a pair-programming CS1 class, had a 66.4% course success retention rate [30] (in that study, compared to a non-majors success rate of 55.9% in a traditional CS1). Our traditional CS class typically 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 offers a high standard to meet.

[5]Graduate research assistants Andrea Forte and Rachel Fithian, and undergraduate research assistant Lauren Rich

The Media Computation course completion retention rate for Spring 2003 was 98%. Our course success retention rate was 89%. During that semester, the comparison section of our traditional CS1 had a 57.1% course success rate.

2.2.2 Evaluation of Motivation

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

- Do students find the current course engaging? An engaging course is more likely to lead to learning [24] [36] [5].
- Do students find computer science engaging?
- 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 approach has potential for attracting new CS majors.

When asked what they like about the class in the midterm survey, the students affirmed that we're succeeding at creating a course that students recognize for its relevance, particularly for non-CS majors:

> "Very applicable to everyday life."
>
> "I dreaded CS, but ALL of the topics thus far have been applicable to my future career (and personal) plans."
>
> "The professor answers questions in class and online and is concerned about our success in the class. He also seems to understand that most of us are not engineers and most likely won't do straight programming in the future—just the way of thinking is important."
>
> "I think that we're doing things that I could actually use as an architecture major–I like dealing with pictures and sounds.

Students who have positive attitudes about CS and who enjoy computing are likely to continue in computer science. While it is too early to tell what CS courses Media Computation students will take in the future, we asked on the final survey whether or not they would be interested in taking a more advanced media computation course. 60% of the female respondents answered that they would take a second course (overall course average of 63%).

Surprisingly, when asked *on the same survey* whether or not they plan to take more CS courses, only 6% of those same female respondents responded affirmatively (9.3% overall). Why would 60% be willing to take media computation, while only 6% planned to take more CS courses? One possible explanation is that there is currently no advanced media computation course, and, given the current selection of CS courses, Media Computation students don't see a compelling reason to take more. Another might be interpretation of the question: Students may interpret "CS courses" as "traditional CS courses" as opposed to specialized courses like Media Computation.

Students indicated that the course changed their attitudes about programming in general: they now view it as something that they could imagine themselves doing in the future. On the final survey, 30% of the students indicated that they believed that they would program again in the future.

> **Interviewer: Have you ever written code outside of assignments?**
>
> Student B: Sometimes I would write other stuff on my way to an assignment but not just like I sat down and wrote something. I don't have time to play with it. Like, I'm not gonna delete JES off my computer, and I may play with it when I get some free time later on, but not yet.

2.2.3 Evaluation of Learning

We developed several problems that we attempted to put on all three courses' exams and quizzes. Unfortunately, a number of issues such as different rates of development and sequencing of concepts in each course, different times and numbers of exams and quizzes, and problems equating concepts in different languages prevented us from distributing the problems as uniformly as we would have liked. In addition, those questions that did make it onto multiple exams were modified by the individual course instructors and teaching assistants.

In general, we found that the different programming languages used, the differences in directions given on exams, and the kinds of examples provided on the exams created unique conditions for each course and rendered the results fundamentally incomparable. For example, the most common problem among the Engineering students' was mishandling of `ELSE` clauses. Media Computation students never used an `ELSE` clause. It's difficult to compare these results in terms of learning.

A more reliable indicator of the Media Computation students' programming achievement might be found in students' attempt to solve a classically difficult problem, such as *The Rainfall Problem.* The rainfall problem is: "Write a program that repeatedly reads in positive integers, until it reads the integer 99999. After seeing 99999, it should print out the average." At Yale in the mid-80's, Elliot Soloway gave this problem to several groups of students [39]. Of first semester CS1 students, only 14% of the students got it correct (completely correct other than syntactic errors). Only 36% of CS2 students got it correct, and only 69% of students in a junior-senior level systems programming course. This was one of the first empirical demonstrations that learning about programming was less than what we might have expected. (This finding was replicated in [27].)

Our students were not directly comparable to the Yale students. For example, we hadn't covered the same things (e.g., at the time of the exam, they had only seen `WHILE` once in class), and we had to change the wording of the problem accordingly. Nonetheless, 14 people out of 113 (12%) who took the test "got it"—by Soloway's standards that means they had a correct solution (aside from syntactic errors). With partial credit, the average on the problem was 45%.

The traditional CS1 also included a variation of the rainfall problem on *two* of their tests, but we were unable to obtain the raw problems to code ourselves and compare. On the first test (which stipulated the use of tail recursion), the result as graded by teaching assistants was an average score of 53%. On the second test (which was open to any kind of iteration), the result was an average score of 60%.

Overall, the rainfall problem continues to be a hard problem for CS1 students. These results don't show that the Media Computation students are learning programming remarkably well, but they do show that these students aren't out of the ballpark either. The bottom line is that it's still an open question how much the Media Computation students learned about programming or CS1-relevant concepts, and how they compare to students in other classes.

2.3 Dissemination and Self-Sustained Distribution

During the time of this project, we have been very active in disseminating the results and materials, helping others to adopt the materials, and taking steps toward self-sustained distribution.

- At SIGCSE2003, Guzdial hosted a tutorial on *Media Construction Projects in Computer Science Courses* which covered the basics of doing media computation in Python, Java, and Squeak. The 11 participants all stated on the final evaluation that they would recommend this workshop to others.

- A paper on the planning and development of the Media Computation course was presented at ITiCSE 2003 [16].

- Prentice-Hall has signed a contract with Guzdial to produce a book *Introduction to Media Computation*. A pre-release version of the book is available as of January 2004, with an expected release of the completed book in the 2004-2005 academic year.

3 Project Plan: Development, Assessment, and Expansion

The *Introduction to Media Computation* class continues to be offered at Georgia Tech. During the Fall 2003 semester, 305 students took the class, still with approximately 2/3 female students. Only six students dropped the course, and the final WDF rate was 13%. The course is being offered currently in the Spring 2004 semester to 400 students, with new faculty teaching it who were not involved in the course's development.

These initial results provide strong proof-of-concept results, which, built upon the research base we cite, make a strong case for moving further with this approach. We see three next steps:

- **Development:** Currently, we have materials to support a single introductory computing course around the media computation approach. While we believe that we are having impact with the single course, studies such as the Margolis and Fisher work [25] suggest that an "alternative path" will have greater impact on the representation of women in computing. We plan to develop a second course, which will be integrated into our CS minor and major. In addition, we are developing introductory materials in Java to make our approach more accessible to undergraduate CS programs that use Java in their introductory course[6]. We hope that these two courses, with their integration into our degree programs will become a national model for an alternative path into computing that is attractive to women.

- **Evaluation:** While our trial results are strong, it's also clear that our support for our hypotheses about changing attitudes and learning is incomplete. We plan to study learning through this approach, attitudinal change of women towards computer science, and long term impact on attitudes and use of technology.
- **Expansion:** We are now working with a set of collaborators at different kinds of undergraduate institutions who are already offering or plan to offer courses using the media computation approach. (See included letters of support.) We plan to hold an annual meeting for sharing course materials, assessment methods, and data collected on the impact on women. In addition, we are requesting consulting funds so that we can pay faculty to work on joint projects.

All of our development and assessment will be overseen by the Georgia Institute of Technology's Human Subjects Review Board. All the researchers working on the project will be certified by the Board as capable of doing human subjects research. All our instruments will be reviewed by the board to respect the rights and safety of the subjects.

We have additional plans for the media computation approach beyond the proposed research:

- We have a hypothesis that the media computation approach may also be motivating and engaging for minority groups in computer science, other than women. Because of the emphasis on a relevant application of computing, creative expression, and a conducive social climate, we believe that the approach may motivate others who do not find traditional computing to be engaging. However, we do not have the research findings at this time to support the hypothesis–there are too few members of minority groups in our pilot study to support the claim. We are developing collaborations with HBCU's and Hispanic groups in the Atlanta area to find testbeds where we can test this hypothesis.
- We are also exploring whether the media computation approach is useful earlier in a student's academic life. We are currently working with the Georgia Department of Education to create workshops using the media computation approach for teachers who need certification in Computer Science in order to teach general and AP computer science classes. The Department of Education believes that the media computation

[6]The Java development is funded by an NSF CISE Educational Innovations grant #0306050

approach may be engaging for teachers, especially those without previous computing experience. The Department of Education has also invited us to be part of the next round of Information Technology curriculum standards discussions next year, to explore how media computation might play a role in IT instruction in high schools across the state.

In the sections below, we detail the work that we are proposing.

3.1 Development: Developing a model for an alternative path

The work of Margolis and Fisher point out that significant curriculum change that impacts women (and potentially other minority groups in computer science) requires true alternative paths into computer science. We don't need to eliminate existing paths, because they are obviously meeting the needs of a large class of students. But we also need paths that are more obviously relevant and that emphasize more creativity and social context. One of our proposed efforts is to grow our single course into a true alternative path into computer science.

The existing course, *Introduction to Media Computation*, does meet the majority of the requirements for a "CS1" course as defined in the ACM/IEEE *Computing Curriculum 2001* standards [2]. For many CS programs, this course would probably work for their CS1. The current Media Computation course covers well the knowledge units in the topics *PF1 Fundamental programming constructs*, *PF2 Algorithms and problem-solving*, and *SP1 History of computing*. However, we only touch on *PF3 Fundamental data structures* (just strings, lists, and arrays, with little trees and no graphs or linked lists) and *AL3 Fundamental computing algorithms* (linear search, some binary search, but no graph traversals or sorts). The second course would need to cover the rest of PF3 and AL3 in order to provide a complete alternative to CS1 topics.

We plan to use a media-oriented approach for some of the course, but with an emphasis on representation. We will argue to the students that the first course focuses on *surface* features of objects (their looks, sounds, and animation). Most professional media work today also captures *structure*–a network to represent the motifs in a song, or a graph to represent the placement of objects in a graphical scene. We will introduce dynamic data structures as a way of describing structure in the world. We plan to continue on to talk about describing behavior as well as structure, an example of which is the method that complex animations like the villagers in Disney's *The Hunchback of Notre Dame* were created and filmed. From there we can introduce object-oriented design, and eventually lead to discrete event simulation where algorithms issues like insertion into ordered lists versus sorting event lists become important topics. This order of events is much the same as in the original object-oriented programming curricula developed by Kay and Goldberg in the 1970's [22].

We are planning to teach this course for the first time in Spring 2005. The two courses (*Introduction to Media Computation* and *Representing Structure and Behavior*) will be considered equivalent to our current majors' CS1. Students who complete the two term media computation sequence will then be allowed to take from there our traditional CS sequence of classes. Thus, we will have a complete *alternative path* into computing, through media computation.

Based on interviews and surveys in our pilot study, and in informal discussions through our subsequent offering of the first course, we do believe that students will use this path to take more computing classes and perhaps consider a Computer Science major. We believe that a *minor* in CS (that would include either the majors' CS1 or the media computation sequence) would also be popular, and we have a proposal for such a minor now.

We hope that this strategy will prove to be a national model. Through our dissemination efforts, we hope to publicize this approach and convince other CS programs both to try our media computation path and to create alternative paths in general. We describe our dissemination approaches later in this document.

3.2 Evaluation: Learning, Attitudes, and Longitudinal

As the class moves out of it's trial period, and as others start to teach the class, we will probably see some diminishing of the euphoria around a non-majors CS1 course. In the next year to two years, we should see results that better reflect the effectiveness of the course.

3.2.1 Learning

We have found it difficult to establish what students are learning in an innovative CS1 course. We can anticipate that many new educational developments will differ from traditional CS1 courses on features such as order of topics, kinds of topics covered, context for topics, and programming language used—as ours does. Because of differences in the order in which topics are covered, it's difficult to compare learning on midterm exams. Even when we simply focus on end of term measurements, simple programming activities are difficult to compare when there are significant language differences (e.g., as in our case, Python vs. Scheme vs. Matlab).

We propose to construct an instrument to measure concepts that one would expect to be covered in a CS1 course, as defined by the *ACM/IEEE Computing Curriculum 2001* [2], in a manner as language independent as possible. We would expect this instrument to be used at the end of a course. Example topics that we would like to include follow:

- We can ask what concepts like *iteration*, *conditionals*, and *assignment* mean, without presenting their syntax, but perhaps with asking the students to present the syntax of the language they learned (making the instrument language-independent, but the evaluation of the instrument langauge-dependent).

- We can ask about data structures such as *arrays*, *matrices*, and *trees* without requiring language explanations.

- We can ask about algorithmic complexity and similar theoretical concepts.

We will use this concept inventory to develop a measure of comparison of whether the students in the innovative course are learning the same kinds of computer science as students in the traditional course, and if the learning is improving as the class develops. We plan to use this concept inventory at the end of the term in successive years and in both the traditional and media courses, in order to develop a basis of comparison.

3.2.2 Attitudes

Many students' attitudes toward computer science today, especially women and non-CS majors, are abysmal as described by several studies [25] [1]. Our results in the *Media Computation* course have been contradictory on the students' attitudes about computing. Our students emerge feeling that programming to manipulate samples in arrays and pixels in matrices is relevant and interesting, and worthy of additional courses—but they are still overwhelmingly disinterested in "computer science."

We plan to develop a survey that asks for fine-grained distinctions about what interests the students about their current course and its content and about what they perceive as computer science. We want to understand issues such as:

- What are their perceptions of what computer science is? How relevant is computer science as they perceive it to their current and future careers?
- How has their attitude toward computer science changed with respect to the interventions they've participated in? How do students think about computer science with respect to the known inhibiting factors to engagement with CS that have been identified such as social setting, tediousness, and creative expression?

We plan to ask the same question in multiple ways to establish internal validity [7]. Our plan to develop and validate the attitude survey is through an iterative process of interviews and survey development using methods described in [13] [3] [8]. We plan to use interviews to develop the survey questions. We will then use these surveys, and interview a sample of students who answer the surveys to check our development process.

3.2.3 Longitudinal evaluation

An additional area of evaluation that we plan to introduce is longitudinal tracking of Media Computation students as they continue their careers. We want to know:

- Do these students take more computer science courses?
- Do these students become computer science majors?
- How does the course impact the students' use and learning of computation within their own majors? For example, do Liberal Arts majors who use Photoshop learn it more easily or use it better (by their own estimation) than students who do not take the course? Do Architecture majors learn AutoCAD more easily or use it better?

Our plan is to use a combination of email surveys and in-person or phone interviews.

- At the end of each year, 10% of the students who have taken the Media Computation course will be surveyed by email to ask about the CS classes they've taken and their use of computational tools.
- At the end of each year, some small sample of students who have taken the course (perhaps 3–5 students) will be interviewed (by phone or in person) to understand how the course has effected their later careers and interaction with computation. These data will also be used to inform further development of the course materials.

3.3 Expansion: Involving other schools and dissemination

We are developing a set of colleagues at other schools who are already trying a media computation approach or who are planning such innovations in their courses. We have identified a set of these colleagues with whom we will focus our attention in order to explore the implementation of the media computation approach to introductory computing across a relevant range of school settings. We know that private schools tend to encourage *more* women in computer science, and research institutions tend to *discourage* (have fewer) women in computer science [4].

- **Public two-year**: Charles Fowler at *Gainesville College* has already offered sections of *Introduction to Media Computation* in the Summer and Fall 2003 semesters.
- **Public four-year**: *Kennesaw State University* is planning to use a media computation approach in summer sessions during this next year and is considering use of a media computation approach in its traditional CS1.
- **Private four-year**: Brian Howard at *DePauw University* has already offered a short course on *Art and Algorithms* using the media computation materials. He is considering use of a media computation approach in their Freshman Seminar.
- **Private research university**: Tim Hickey is already teaching an innovative non-CS majors introductory course at *Brandeis University* [21] in Scheme. We have found that our materials (including the media classes we created for our Python students) could be used in his Scheme class. He tried a lecture on media computation in the Fall 2003 class, and is planning to teach a summer session using the media computation approach.

Our plan is to hold an annual two-day meeting of the faculty involved in these efforts, paid for by grant funds. We want these faculty to share their methods and materials with one another, but in addition, to work with us on use of evaluation methods and to share evaluation data. The goal is to learn how media computation as a context for introductory computing succeeds (or doesn't) in schools with different characteristics. Other schools have contacted us about using our approach (including Spelman College. Xavier College, and University of California, Santa Cruz). We plan to invite others to join our annual workshops, but our development and evaluation focus will be on these schools to explore the public vs. private, and two-year vs. four-year vs. research university characteristics that have been identified as important in the research literature.

At these schools, we will use the evaluation approaches that we have developed. We will track WDF rates to see if the media computation approach is having an impact on retention of women in computing. We will also use our developing methods to study learning in computing, attitude change towards computing, and the longitudinal impact of the innovative approach on the students' relationship with computing technology.

We are asking for two other items in our budget to support the expansion and exploration of the media computation approach:

- We are asking for travel budget, beyond what is necessary for conference dissemination, in order to be able to send instructors and evaluators from Georgia Tech to collaborating schools to give guest lectures and to help development of materials and use of evaluation methods.

- We are also asking for consulting funds that we can use to pay faculty from other schools who are working on joint projects. We are requesting $10,000/year for the four years of the project in order to pay for several week-long collaborations. Such joint projects we envision include:
  - Adapting materials from Georgia Tech courses for their own context.
  - Working with the Georgia Tech team to develop new materials.
  - Faculty at two different institutions who want to develop a common assignment to be tested at both institutions.
  - Non-CS faculty (e.g., in art, literature, or film) at any of the institutions who might work with us to develop lecture or homework materials that are more authentic for that discipline.

3.3.1 Dissemination

Our further plans for dissemination are:

- We will continue to write papers as results arise and propose workshops/tutorials at ACM SIGCSE, ITiCSE, and IEEE/ASEE FIE conferences, as well as conferences on learning sciences such as the International Conference of the Learning Science (ICLS) and Computer-Supported Collaborative Learning (CSCL). For SIGCSE2004, we are presenting a workshop on media computation in CS classes, a paper on the results of our pilot study, a birds-of-a-feather session on innovative approaches to non-CS majors computing, and in two panel sessions based on the media computation approach. Depending on acceptances and travel budgets, we plan to present at two-to-three conferences per year.

- We will continue to make materials available on our development website[7], and we

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

plan to create a project website (with a more easily remembered URL) for wider dissemination.

We already have a plan for self-sustained distribution of the materials, through a contract with Prentice-Hall. Prentice-Hall has already expressed interest in a Java version of the materials and in a text for the second media computation course.

4 Conclusion and Timeline

We believe that media computation is a viable context for introducing introductory computing, and in particular, is an alternative path in computer science that will attract women into computing. Our approach is based strongly on the existing literature on women in computing, and we believe that our pilot study is proof of the viability of the context. Through the proposed work, we plan to implement a complete alternative path, carefully evaluate the impact, and explore the impact as we expand the approach into new contexts. We believe that the potential broader impact of this work is to attract under-represented groups (such as women) to computing and to create a broader base of computing-literate citizens by helping students who were not succeeding to be more successful at computing.

We are proposing a four year effort. Our timeline follows:

- *Year One*: We plan to develop our new evaluation methods for learning, attitudes, and longterm impact. We will have our startup workshop with collaborating faculty at other institutions. We will also develop and teach the second course as a pilot.

- *Year Two*: We will continue to iterate on the instruments and gathering those results, but we expect that the most significant effort will be in helping the collaborative institutions to develop their own media computation classes. The first *Media Computation* text should be published this year. The second media computation course will be made into a standing course during this year.

- *Year Three*: In this year, we will be in full swing: A complete alternative path into computing established at Georgia Tech, media computation approach courses taught at all the collaborating institutions, and our evaluation methods tested and validated. Year three will be our year for really understanding what's happening with this approach, across multiple measures, in schools with different characteristics.

- *Year Four*: Year four is where we anticipate the most significant dissemination will take place. All of the schools will have interesting results to share, which we plan to disseminate with an eye toward national impact. But just as important, each of the schools will be looking for what to do with their new media computation approach courses. We will work with them to construct alternative paths and/or to develop new, additional courses as is appropriate for their context–and to serve for other CS programs in similar contexts.

References

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

[2] ACM/IEEE. Computing curriculum 2001. *http://www.acm.org/sigcse/cc2001* (2001).

[3] Allen, D., Ed. *Assessing student learning: From grading to understanding.* Teachers College Press, New York, 1998.

[4] Barker, L., and Garvin-Doxas, K. The effect of institutional characteristics on participation of women in computer science bachelors degree programs. In *ITiCSE 2003: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, D. Finkel, Ed. ACM, New York, NY, 2003, p. 242.

[5] 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.

[6] 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.

[7] Cronbach, L. Test validation. In *Educational Measurement (2nd Ed.)* (Washington, D.C., 1971), R. Thorndike, Ed., American Council on Education.

[8] Cross, K. P., and Angelo, T. P. *Classroom Assessment Techniques: A Handbook for Faculty.* NCRIPTAL, University of Michigan, Ann Arbor, 1988.

[9] Dann, W., Dragon, T., Cooper, S., Dietzler, K., Ryan, K., and Pausch, R. Objects: Visualization of behavior and state. In *ITiCSE 2003: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, D. Finkel, Ed. ACM, New York, NY, 2003, pp. 84–88.

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

[11] Felleisen, M., Findler, R. B., Flatt, M., and Krishnamurthi, S. *How to Design Programs: An Introduction to Programming and Computing.* MIT Press, Cambridge, MA, 2001.

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

[13] Gardner, H. Assessment in context: An alternative to standardized testing. In *Changing Assessments: Alternative Views of Aptitude, Achievement, and Instruction* (Boston, 1992), B. R. Gifford and M. C. O'Connor, Eds., Kluwer Academic Publishers.

[14] Guzdial, M. Use of collaborative multimedia in computer science classes. In *Proceedings of the 2001 Integrating Technology into Computer Science Education Conference*. ACM, Canterbury, UK, 2001.

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

[16] Guzdial, M. A media computation course for non-majors. In *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE) 2003 Conference* (New York, 2003), ACM, ACM, pp. In–Press.

[17] 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.

[18] 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.

[19] 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.

[20] Guzdial, M., and Soloway, E. Computer science is more important than calculus: The challenge of living up to our potential. *Inroads – The SIGCSE Bulletin 35*, 2 (June 2003), 5–8.

[21] Hickey, T. Incorporating scheme-based web programming into computer literacy classes. In *The Scheme 2002 Workshop* (Pittsburgh, PA, October 2002).

[22] Kay, A., and Goldberg, A. Personal dynamic media. *IEEE Computer* (1977), 31–41.

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

[24] 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.

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

[26] 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.

[27] 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.

[28] McDowell, C., Bullock, H., Fernald, J., and Werner, L. The effects of pair-programming on performance in an introductory programming course. In *The Proceedings of the Thirty-third SIGCSE Technical Symposium on Computer Science Education, 2002*, D. Knox, Ed. ACM, New York, 2002, pp. 38–42. Pair-programming (social) reduces attrition rates.

[29] 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.

[30] Nagappan, N., Williams, L., Ferzil, M., Wiebe, E., Yang, K., Miller, C., and Balik, S. Improving the cs1 experience with pair programming. In *Twenty-fourth SIGCSE Technical Symposium on Computer Science Education* (New York, NY, 2003), D. Joyce and D. Knox, Eds., ACM, pp. 359–362.

[31] 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.

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

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

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

[35] 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.

[36] Pintrich, P. R., and Schunk, D. H. *Motivation in Education: Theory, Research, and Applications.* Prentice-Hall, 1996.

[37] 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

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

[39] 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.

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

[41] Wilson, B. C., and Shrock, S. Contributing to success in an introductory computer science course: A study of twelve factors. In *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, R. McCauley and J. Gersting, Eds. ACM, New York, 2001, pp. 184–188.

[42] 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.

[43] 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.

5 Budget Justification

To fund this effort, this proposal includes a budget request for a four year project.

- 12 months of Allison Tew per year, for all four years of the grant. Allison Elliot Tew will be a Research Scientist on the project. For four years, Allison has been the Director of Student Services. She is also a Ph.D. student in the College of Computing in the area of Learning Sciences and Technologies. She also has prior experience in project management and software engineering. She'll handle the day-to-day management of the project, and will be the main person developing the measurement instruments used in the project, reviewing the data and results, and working with faculty to implement the interventions.
- We request funding and tuition for the two GSRA's on the project. Our plan is for one of the GSRA's to focus on the evaluation effort, and the other to focus on development of materials and interaction with the collaborating institutions.
- We also request funding for a portion of a shared research scientist for maintenance of laboratory equipment in the GVU Center.
- We are requesting fringe and computing charges for these personnel.
- One summer month of the PI's time in year four.
- We are requesting $5,700/year to pay for travel, lodging, and meals for faculty attending our annual workshop.
- We are requesting $10,000/year of consulting fees to pay faculty for week-long collaborative projects during the four years of the grant.
- Travel support ($10,000/year) for attending conferences to disseminate materials and results and conduct workshops.
- Materials and supplies ($4,000/year), to support the development and evaluation effort.

6 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, many of which are quad-processor machines. In addition, there are more than 1,000 workstation class machines from Sun, Silicon Graphics, Intel, and Apple especially for student use. A number of specialized facilities augment these general-purpose computing capabilities. The hardware that will be purchased for this project will be of similar quality to what the students use, for testing purposes, but will be set up to facilitate development.

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. The affiliated Multimedia, Computer Animation, Audio/Video Production, Usability/Human Computer Interface, Virtual Reality/Environments, Electronic Learning Communities, Computational Perception, Software Visualization, Biomedical Imaging, Collaborative Software, and Future Computing Environments labs provide shared facilities targeting specific research areas. These laboratories' equipments will be of use in developing our multimedia projects.

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 Nikon digital camera.

All of the College's facilities are linked via local area networks which provide a choice of communications capabilities from 10 to 1000 Mbps. The College's network employs a high-performance OC12C (622 Mbps) ATM and GigabitEthernet (1000 Mbps) backbone, with connectivity to the campus ATM network provided via OC12C. The primary campus Internet connection is provided by a direct 100 Mbps link to the service provider's Atlanta switching center, augmented by OC3C ATM and OC12C connections, respectively, to the NSF vBNS (very high performance Backbone Network Service) and Abilene research networks. Georgia Tech is also leading southern regional gigabit network efforts (SoX.net, the Southern Crossroads) as part of Internet2.

Additional computing facilities are provided to the Georgia Tech campus by the Institute's Office of Information Technology (OIT), including five public-access clusters of Sun, Apple, and Dell workstations, a collection of Sun multi-processors which are treated as a single computational resource via login load sharing, and various mainframes.