Computers for Communication, Not Calculation: Media as a Motivation and Context for Learning

Andrea Forte
Mark Guzdial
College of Computing, GVU Center
Georgia Institute of Technology
{aforte, guzdial}@cc.gatech.edu

Abstract

As the skills that constitute literacy evolve to accommodate digital media, computer science education finds itself in a sorry state. While students are more in need of computational skills than ever, computer science suffers dramatically low retention rates and a declining percentage of women and minorities. Studies of the problem point to the over-emphasis in computer science classes on abstraction over application, technical details instead of usability, and the stereotypical view of programmers as loners lacking creativity. In spring 2003, Georgia Institute of Technology trialed a new course, Introduction to Media Computation, which teaches programming and computation in the context of media creation and manipulation. Students implement PhotoShop-style filters and digital video special effects, splice sounds, and search Web pages. The course is open only to non-computer science and non-engineering majors at Georgia Tech, such as liberal arts, management and architecture students. The course is supported through the use of a Web-based collaboration environment where students actively share and discuss their digital creations. The results have been dramatic. 120 students enrolled, 2/3 female, and only three students withdrew. By the end of the semester, the combined withdrawal, failure and D-grade rate had reached 11.5%--compared to 42.9% in the traditional introductory computer science course. 60% of the students who took media computation reported that they would be interested in taking an advanced version of the course; only 6% reported that they would otherwise be interested in taking more computer science. Results of the trial indicate that media computation motivates and engages an audience that is poorly served by traditional computer science courses.

1. Introduction

Communication plays such a fundamental role in human lives that the tools of the literate become nearly transparent. Our pencils and pens, newspapers and books, signs and labels are no longer noticed unless they are absent. In the same way, computers are becoming transparent tools, so enmeshed in everyday practices that many of us have already ceased to recognize their fundamental role until some unusual circumstance prevents us from using them. Computers have not only been introduced to our repertoire of communication technologies, they have become indispensable tools, permeating and connecting cultures at astonishing rates.

Andreas diSessa defined literacy as: "a socially widespread patterned deployment of skills and capabilities in a context of material support… to achieve valued intellectual ends" [1]. If we accept that computers are altering the context of material support, and that communication is a valued intellectual end, what new skills and capabilities do individuals need to leverage the power of computers as communication tools?

Prior to the computer, most avenues of communication and most definitions of literacy involved one medium: text. Kay and Goldberg described the computer as a "metamedium," or a medium that is capable of supporting the creation of all other media [2]. As computer technology makes accessible a new range of media, the skills that constitute literacy must likewise evolve. People can now communicate in ways that traditional, text-based literacy did not support. In order to define these new literacy skills, we must acknowledge that literacy implies not only consumption, but also creation. Reading alone does not constitute our traditional notion of literacy: reading and writing are fundamental literacy skills. Resnick, Bruckman and Martin point out that just as the expressive possibilities of playing a piano are far richer than playing a stereo, using a computer to construct and design captures the generative potential of the tool in a way that simply using it to view and listen do not [3]. Since computers represent a variety of expressive media, digital media creation can be numbered among new literacy skills for the computer age.

A new course at the Georgia Institute of Technology, Introduction to Media Computation, combines learning about the fundamentals of digital media with basic programming skills and computer science concepts. Media and computation are complementary—digital media are computationally created and manipulated. In the media computation course, students who are interested in working with digital media can look "under the hood" to see how their art, music and websites come to be. At the same time, a creative context for programming provides students with an interesting introductory computer science experience. The constructionist approach to learning asserts that people learn particularly well when they are engaged in constructing a public artifact that is personally meaningful [4]. Introduction to Media Computation provides students with an environment in which their creative efforts to construct digital media double as programming exercises. When the course is finished, students have not only learned to create pictures, sounds, animations, and Web pages, they have also learned the fundamentals of programming and computer science.

2. Previous Work on Media Literacy

Soloway, Guzdial and Hay begin to define computer skills that augment traditional literacy by pointing out that computers do a lot more than allow us to simply manipulate words [5]. Computers make accessible a range of media that were once difficult to manipulate and create. In terms of education, these new media are a treasure; not only can students write to learn, now they can also create pictures, sounds and interactive documents. The obstacle for educators lies in creating environments that invite students to take advantage of these unfamiliar media in order to learn from them without first investing immoderate amounts of time learning to program the computer. If the only people who are able to take full advantage of computers' expressive capacity are programmers, then many people will lose valuable educational opportunities.

Like Papert and Kay [6], [2], diSessa goes even further in suggesting that everyone should learn to program: a computationally literate population will be able to think about and do new things that are unimaginable to us in the same way that our modern, text-based literate culture is practically unfathomable to pre-literate cultures [1]. DiSessa argues that the media we create and our ability to use them to their greatest advantage of expression is a cornerstone of human creativity. Every medium has unique representational qualities; it provides a new and unique way of exploring and expressing ideas. In order to take full advantage of the computer's power as a meta-medium, we must be able to craft new kinds of media that capture the essence of what we want to say.

3. Computer Science Education and Media Literacy

DiSessa's view of literacy in the computer age is not only forward thinking, it looks nearly unachievable to those who are familiar with the failure of computer science departments to reach a diverse student population. Evidence of this can be seen in the growing amount of research on computer science for non-majors at institutions across the United States [7], [8], [9]. Among non-computer science majors at Georgia Tech, introductory computer science is an extremely unpopular required course. While the number of students obtaining computer science degrees has increased nationally in recent years [10]. WFD rates (withdrawal, F, or D grades) remain notoriously high in introductory courses. At Georgia Tech, the overall WFD rate in introductory computer science has averaged nearly 30% over the past three years; other universities have reported WFD rates ranging from 25 to 50% [8], [11]. It seems that traditional introductions to computer science are more likely to frustrate students than attract them to the field. This trend has troublesome implications for other fields, too. If students of computer science are the only group to seek out computational literacy, new media will be slower to develop in life sciences, the humanities, and other disciplines.

High WFD rates are not the only problem faced by computer science educators. Few female and minority students choose to pursue computer science learning. The low number of female students is especially striking, considering the high percentage of female college students overall. In 1999-2000, fewer than 30% of the bachelor's degrees awarded in the United States in computer and information sciences were awarded to women, yet women received nearly 60% of all bachelor's degrees awarded in the same academic year [10], [12]. If educated women are not learning to be computationally literate, what role will they play in a society whose forms of expression are increasingly defined by the computationally proficient?

Clearly computer science education has a lot of work to do before it can live up its potential in supporting students' efforts to achieve literacy in the

computer age. Whether or not we accept that computational literacy is the goal, it is clear that the traditional definition of literacy must be extended to include proficiency with new forms of media and media that have become more accessible with the widespread adoption of the computer. Computer science classrooms are as yet an underutilized resource for the teaching and learning of media literacy.

4. Introductory Computer Science Courses Today

Since computer science first emerged as an academic discipline, educators have fiercely debated the best format for introductory courses. Despite intense discussion and continuing attention, no single approach has emerged as the most effective or popular. The Association for Computing Machinery (ACM) published updated computing curricula guidelines in 2001 that described several types of introductory course implementations: imperative-first, objects-first, functional-first, breadth-first, algorithms-first, and hardware-first. While ACM guidelines acknowledge that certain implementations are better suited to meet non-majors' needs, their recommendations generally involve sequences that span two or three semesters, which exceeds the amount of time that most non-majors spend on computer science.

The increase in non-majors who are required to take introductory computer science courses or who wish to improve their technical skills is drawing more attention to the strengths and weaknesses of traditional course implementations for a diverse student population. Some educators argue that the widespread emphasis on programming in introductory courses causes disinterest or undue anxiety among non-majors or does not impart useful skills [13], [9]. Others believe that a single course and set of content can serve non-majors and majors alike, but that variable levels of cognitive engagement with the material should be required [8]. Examples of non-majors' introductory CS courses that require no programming can be found at the Harvard University Extension School [13], and at Bowling Green State University [14].

Four of the six course implementations described in the ACM Curriculum 2001 emphasize programming; indeed, despite heated debate, most introductory computer science courses do require students to engage in some kind of programming exercises. One particularly innovative introductory CS course uses 3D animation to make certain aspects of object-oriented programming concrete for beginner programmers [15]. Clearly, it is our view that programming is an essential aspect of understanding how digital media are created and manipulated. The course implementation trialed at Georgia Tech in spring 2003 is perhaps best described as "data-first." Students start with data that is interesting to them—their own photographs, music, video or text—and employ programming as a way of leveraging computers to use and transform that data.

5. Introduction to Media Computation

Georgia Tech's introductory course in media computation is an attempt to address the problems in computer science education described in section 3 as well as introduce media literacy goals to the computer science agenda. We believe that the first problem, high WFD rates, is symptomatic of a failure to communicate the value of computer science to students. If computer science is not perceived as interesting or useful, students fail or drop out. It has become painfully clear that generic, one-class-fits-all computer science fails to meet the needs of a diverse student body. Historians, writers, architects, and engineers (just to name a few) have diverse interests and require different kinds of computational proficiency to perform the tasks that are important to them. The use of domain-specific contexts for computer science learning is being explored by researchers with the aim of improving students' experience in IT courses [16], we propose using this approach for introductory computer science. Because media computation focuses on data that is important to students—their own photographs, recordings, and creations—and allows them to use computation in a personally expressive way, we expect to better engage non-computer science majors than traditional introductory courses and, as a result, improve retention rates.

A second motive for creating a class in media computation is to better engage female students in computer science. Like other institutions, Georgia Tech has attracted appallingly low numbers of female computer science students; even worse, female enrollment has dropped by roughly a percentage point in each of the last four fall semesters [17]. The scarcity of females in computing nationwide has generated a spate of research to better understand why so few women seem to find the field appealing. At Carnegie Mellon, a longitudinal study was conducted from 1995 through 1999 to investigate female computer science students' motivations and experiences. Researchers found that reasons for

women's disinterest in computing include the emphasis in computer science courses on technical detail rather than application, the perception of computing as an uncreative or anti-social field, and a frequently uncongenial culture [18]. Ongoing studies by the American Association of University Women (AAUW) echo these results, naming "computer culture" as a major deterrent for women in computing [19]. Both of the studies named here also suggested that female students are put off by feelings of incompetence, which may be fed by code-competent males who exaggerate their skills or by the fact that females typically have less experience in programming by the time they reach college. Turkle and Papert maintain that the dearth of women in computing is induced not only by historical prejudice or discrimination, "but by ways of thinking that make them reluctant to join in" and that "equal access to even the most basic elements of computation requires an epistemological pluralism, accepting the validity of multiple ways of knowing and thinking" [20]. We hope that a course in media computation will provide a new entrance to computational literacy, an introductory CS experience that overcomes the above-named reasons for female disinterest in computing.

How can media computation hope to solve all of these deeply entrenched problems?

- **By connecting computer science content to application in a relevant domain: media creation.**
  If we can introduce computational skills in a personally meaningful and relevant context, we believe students will be more likely to "stick with it" and, as a result, achieve more. Communication skills such as media creation will resonate with many students who do not respond to conventional introductory computer science exercises.

- **Second, by challenging students to be creative and collaborative in homework assignments.**
  The perception that computer science is asocial or uncreative has been introduced by researchers of gender disparity in computer science achievement. We believe that the coupling of media creation with programming skills and the freedom to collaborate with other students will bring about a more social, creative classroom culture for introductory computer science.

- **Third, by providing liberal arts, architecture, management, and other non-computer science majors with a classroom environment in which they can make computing their own.**
  The "computer culture" described in gender research is perceived by everyone, not just women. Men and women in other fields can benefit from a diversified computer culture that allows them to achieve computational proficiency in a community that shares their goals, background, and frustrations.

5.1. Course Description

Introduction to Media Computation was piloted at Georgia Tech in the spring semester, 2003. The course begins with an introduction to media encoding. Students learn how computers can save and display media in ways that make sense to the human eye and ear. Discussions of psychophysics (Why does high resolution look better than low resolution? What do we mean by CD-quality sound?) relate to students' own experiences with digital photographs, MP3s, and other electronic media.

In the second week, students begin writing simple code in the programming language Python to modify images. At first they simply change the amount of red, blue or green in an image, but by the third week they create more complex code to darken or lighten an image, turn a color photograph black and white, or create reflections. In order to produce these effects, the students must understand what a matrix is (a picture is a matrix of pixels), what a variable is (the variable red could represent the amount of red in any pixel), how iteration works (we alter the picture by looping through each pixel and changing it) and how to use conditionals (only change the pixel if it meets certain criteria).

```
def negative(picture):
  for px in getPixels(picture):
    red=getRed(px)
    green=getGreen(px)
    blue=getBlue(px)
    negColor=makeColor(255-red, 255-green, 255-blue)
    setColor(px,negColor)
```

Fig 1: A program that produces the negative of an image.

In the following weeks, these concepts are revisited and expanded on in the context of sound and text. Students learn to increase and decrease pitch and volume, to search text, to generate HTML, and to manipulate directories and networks. Finally, animation is explored and students use what they learned about pictures to modify animations and create elementary special effects.

During the final few weeks of the semester, after they have become proficient in the computational

manipulation of media, students are introduced to broader CS concepts: Why is Photoshop faster than the code they write? What are interpreters and compilers? What is object-oriented programming? What do other programming languages look like?

Key computer science ideas like abstraction, hierarchical decomposition, representation and encoding of data, and iteration, are introduced more than once in the class. Whereas a more traditional class presents these ideas in the first few weeks with elaboration later, the media computation class reverses that. When students first face iteration, for example, it appears as a set operation. Students manipulate pixels with a statement like:

for pixel in getPixels(picture):

which looks more like a statement of what "pixel" means than an executable statement. Later, more sophisticated notions of the **for** loop are introduced. The detailed explanation of what iteration is (for example, in contrast with recursion and as part of what leads to algorithmic complexity) does not appear until the end of class. The reason for this inversion is that we know that learning occurs concrete-to-abstract [20], [21]. Non-CS majors cannot be expected to have much experience with programming ideas or even sophisticated math ideas. We aim to give them the opportunity to work with these ideas first in a concrete fashion, and then explain them (with the useful abstractions) later.

Students are expected to turn in six homework assignments over the course of the semester, all of which involve programming and entail the creation of their own original media. Three in-class exams assess students' comprehension of conceptual material and ability to read and create code. Two take-home exams assess programming proficiency. A collaborative website (CoWeb) enables students to post questions, discuss problems, and display their work if they choose to do so.

6. Course Evaluation

6.1. Data Collection

The questions we were interested in answering during the pilot offering of the media computation course included:

- Do students take advantage of the creative aspects of the course?
- How does collaboration affect student performance and perception of the course?
- How do female students respond to the course?
- Do students find media computation relevant and/or useful?

We gathered the following data:

- We gave initial, midterm, and final surveys to consenting students.
- We examined the homework assignments of consenting students.
- Interviews were conducted with female volunteers from CS1315 at both the middle and the end of the semester.

6.2. Who are media computation students?

Introduction to Media Computation fulfills the computer science requirement for non-engineering, non-CS students only. The distribution of our data sample is represented by college major in Figure 2.

On the initial survey, which was distributed during the first week of the semester, students were asked whether or not they had any previous programming experience, and, if so, how many years. Media computation students reported very little programming experience: only 17% of the respondents reported any programming experience at all and of these, none reported more than one year. Research indicates that previous programming experience is one of the strongest predictors of success in computer science [22], [23], [24]. Media computation students, then, can be seen as "at risk" students in this field.

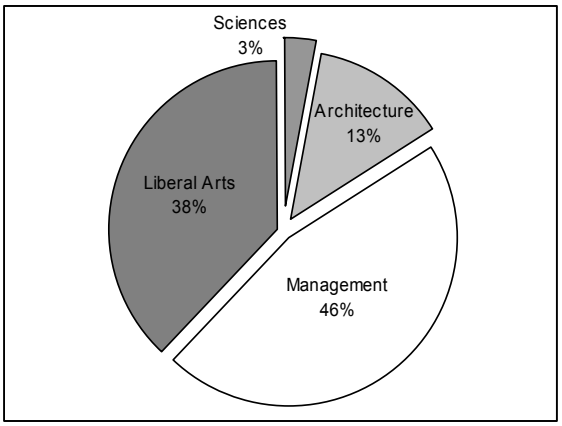

Figure 2: Composition of Intro to Media Computation by major college

6.3. Do students take advantage of the creative aspects of the course?

Survey responses, homework assignments and interviews with media computation students all

indicated that students were not only enjoying the material, they were also taking advantage of the creative aspects of the course and doing interesting things on their own. Some students preferred working in audio, some with images, and many reported that learning how the Web works and creating their own pages was the most enjoyable aspect of the course.

Many students were clearly excited about the potential for using media in ways they had not previously encountered. Two students reported on the midterm survey that they had written programs to reverse popular songs, in order to find out if there were hidden messages. One student reported using Python to create an online scrapbook. Students often turned in homework assignments that included far more complex code than was required. For one homework assignment, students were directed to create a collage in which the same image appeared at least three times with some different visual manipulation each time. As can be seen in the example in Figure 2 (which was showcased by voluntary public posting on the collaborative website), many students accomplished considerably more than the minimum requirements. Some programs reached over 100 lines in length.

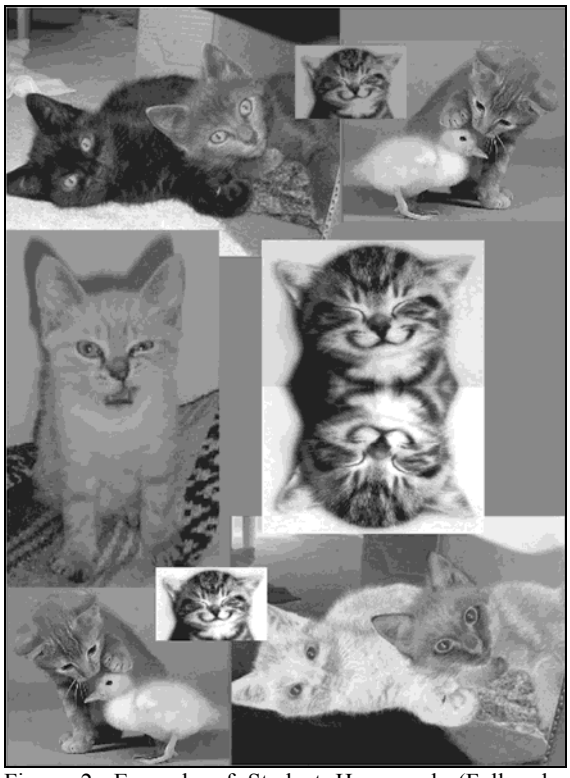

Figure 2: Example of Student Homework (Full color examples of student work can be found at: http://coweb.cc.gatech.edu/cs1315)

In interviews, several students indicated that they considered programming media to be something fun and useful beyond the completion of assignments. Two students revealed that they simply had not had enough time to play with the programming environment as much as they wanted to, and both stated that they would not remove the programming environment from their computers:

> Interviewer: What do you think about the homework galleries on the CoWeb?
> Student: I don't ever look at it until after I'm done, I have a thing about not wanting to copy someone else's ideas. I just wish I had more time to play around with that and make neat effects. But JES will be on my computer forever, so… that's the nice thing about this class is that you could go as deep into the homework as you wanted. So, I'd turn it in and then me and my roommate would do more after to see what we could do with it.

> Interviewer: Have you ever written code outside of assignments?
> Student: 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.

6.4. How does collaboration affect student performance and perception of the course?

We have a basic understanding of the affordances of digital media such as photographs, sounds, animations, and text for computational learning, but still have not addressed the social conditions in which this learning is to take place. Robert Kozma observed that the conditions for learning through media are more complex than the simple presence of that media; thus, he argues, "our media theories and research must reflect both the capabilities of media and the complexities of the social situations in which they arise" [25]. One important social aspect of the media computation course is the liberal collaboration policy: students are allowed to collaborate as much as they wish on homework assignments.

Many students reported that the collaboration policy positively influenced their experience in the course. On the final survey, over 20% named collaboration as the one aspect of the course that should not change. When asked on the midterm survey for their opinion of the collaboration policy, over 95% gave a positive response. Many students explained that collaborating led them to a deeper understanding of the material because it provides an

opportunity to talk about problems and their solutions. (Exam scores averaged nearly 80%, indicating that the students were not simply using the collaboration policy as an opportunity to copy code.) Overall, nearly 40% of the responses to the collaboration question mentioned improved learning. Other responses indicated that students felt more confident about their solutions after discussing them with peers, and that the collaboration policy provided them with an alternate avenue for seeking help. Some students simply felt that collaborating made the class more fun:

> Interviewer: Did you collaborate? Do you have any thoughts on collaboration?
> Student: My roommate and I… we took full advantage of the collaboration. It was more just the ideas bouncing off each other. I don't think this class would have been as much fun if I wasn't able to collaborate.

Whereas we have seen that programming experience is a good indicator of success in computer science [22], [23], one study found that the best indicator is students' comfort in the classroom [24]. We hoped that collaborative activities would help establish a relaxed classroom culture in which students felt comfortable asking questions and seeking help. In order to encourage a less competitive, more supportive learning environment, students were encouraged to ask questions and post comments to a collaborative website (CoWeb). When asked about her use of the CoWeb, one female student made it clear that she felt not having computer science majors in CS1315 made it easier for the non-majors to ask questions and get help:

> Interviewer: Have you ever posted to the CoWeb?
> Student: I think I've posted to everything. Sometimes I'll just make random comments. Sometimes I ask a specific question and he [the professor] asks for clarification. I would feel different in a class with a bunch of CS majors. But since we are there with a bunch of management—other students—it's kind of more comfortable.

Another student suggested that the anonymous nature of the CoWeb made it easier to ask questions, particularly early in the semester when concepts are still new and students may still feel that their questions are "dumb":

> Interviewer: Have you consistently felt comfortable asking questions?
> Student: Not at the beginning. One on one, yes. In lecture, not at the beginning because I felt that I was so far behind other people and the ones who were putting things on the web were the ones who really know stuff. But now I have no problem.
> Interviewer: Have you felt comfortable posting on the CoWeb?
> Student: Yes.
> Interviewer: Do you think the CoWeb is beneficial?
> Student: Yes. And there's no reason to feel uncomfortable because if you feel dumb, just don't put your name at the end! I did that a few times.

6.5. How do female students respond to the course?

When asked what they liked best about the class, female media computation students were more likely to name content than any other feature of the course. When asked what they liked least, the most common response was a specific activity such as exams or homework, followed by the amount and pace of the work. Overall, survey and grade data indicates that female students' responses to the pilot course offering were extremely encouraging, but we were interested in a more qualitative and comprehensive view of female students' experiences in media computation. We wanted to know whether they felt comfortable in class, and whether they felt competent; whether they felt that media computation was worthwhile and engaging.

A series of interviews allowed us to capture the experiences of a few female students. All of the interviewees reported a positive experience in introductory computer science. One student summed up her media computation experience in the following transcript:

> Interviewer: What is the most surprising, or interesting thing you've learned?
> Student: That it's fun. I know that's not specific.
> Interviewer: "It" being CS, or programming, or what?
> Student: Both. The history of the computer and the Web. And that programming is not scary, it is actually pretty cool and when you make a program that actually runs it's a really good feeling. I didn't expect to enjoy it at all because all I'd heard were just the bad stories…

When asked whether or not media computation should continue to be offered, all of the female students interviewed at the end of the semester made a direct comparison to the traditional introductory computer science course, suggesting that the content of media computation is more meaningful to them. One student even stated that:

> 1321 [traditional CS], from what I've heard from people, is not cut out for everybody and it's not fair for the university to demand that we do that stuff when there is so much other exciting stuff you could be doing.

Some studies suggest that computer science classrooms have developed a culture that is uncongenial to women, or is perceived as being so [19], [18], [20]. The reasons for this perception are not well understood, and some may have more to do with popular culture (the unpopular "geek"), or gender differences in social practice than with actual classroom practices. Still, we expect the kinds of collaborative practices described in the preceding section to help alleviate some of the negative perceptions of computer culture identified in the research. Female interviewees repeatedly expressed wonder at the discovery that computer science did not have to be either boring, asocial or uncreative:

> Interviewer: Has the class provided any additional insight into computer science?
> Student: Most certainly. I have seen that there is humor, there's color, there's more than just lines and lines and lines of code that have lots of colons and dots and parentheses that no one could really understand.
> Later: [the professor] is so good at what he does, especially expressing that computer science is not some black screen computer nerd type thing.

All of the students interviewed at the end of the course expressed feelings of competence and indicated that they felt they understood enough to go farther in computer science. When asked whether they plan to take more computer science, very few female media computation students responded affirmatively (6%); however, when asked whether they would be interested in taking advanced media computation, which is not currently offered, the number rose to over 60%. Given the traditional selection of computer science courses, most of these students apparently do not see a compelling reason to take more than what is required. This indicates that media computation has captured the interest of many female students who otherwise would not choose to pursue computer science learning. A particularly striking example of confidence can be found in the remarks of one interviewee at the end of the course:

> Interviewer: Has this class changed your opinion of computer science?
> Student: YES, I'm not intimidated by it anymore. My mom was SO surprised when I told her I want to be a TA she almost fell on the floor, cuz she's heard me complain for years about taking this class and now I want to go do it to myself again!

6.6. Do students find media computation relevant and/or useful?

One indicator of students' attitudes toward a class is the rate at which they drop the course. By drop day, only three students (all male) had dropped media computation—2.5% of 120 students. By the end of the semester, the overall WFD rate had only reached 11.5%. (See Table 1.) Low withdrawal and high success indicates that students' attitudes toward media computation were generally positive.

	Drop Rate	WFD Rate
Media Computation	2.5%	11.5%
Traditional Intro to CS	10.1%	42.9%

Table 1: Comparison of WFD and drop rates for two introductory CS courses, Georgia Tech, spring 2003

Many students reported that they found the content of the course useful. When asked what they liked best about the course at midterm, approximately 20% of media computation students indicated that they enjoyed the content, while about 12% named the usefulness of the content. In comparison, no students in the traditional introductory computer science course named usefulness of content as an attractive feature of the course. In fact, 18% of traditional CS students reported that there was nothing that they liked about the course at all, while no media computation students reported a complete aversion to the course. These results confirm that media computation is engaging many students that traditional computer science does not.

Judging by survey responses, students seem to think that media computation as a subject is relevant and interesting. A number of students commented on the fact that they feel it is important to understand how the programs they use work. Others were intrigued by the fact that they now understood the digital effects they see on television and in movies. When asked on the midterm survey what they liked best about the class so far, students responded:

> I dreaded CS, but ALL of the topics thus far have been applicable to my future career (& personal) plans—there isn't anything I don't like about this class!!!"

> I like that we work with useful applications of code: processing sounds, images, movie clips, etc. I like that the professor is most interested in explaining everything so that the class

understands it."

I think that we're doing things that I could actually use as an architecture major- I like dealing with pictures and sounds.

6.7 What did students learn?

We know that students do learn programming skills, based on their performance on take-home examinations where they had to program on their own. Several of their programs were in the range of 25-75 lines of code, which is not insignificant in a CS1 course. We did attempt to compare student performance across three different CS1 courses offered in Spring 2003, but were unsuccessful. The variations in sequencing in the courses, differences in how the problem was presented to the different classes, the use of different languages, and the choice of a known hard problem from the literature [26] led to a floor effect for all courses and no significant differences.

Based on performance on examinations, we believe that the students learned about key computer science ideas, especially those that were most relevant to the media computation context. Whereas all CS1 courses discuss representation and encoding, it was particularly relevant to students who care about the number of colors available to them in a pixel color encoding and what range of sounds can be encoded given a sampling rate and sample size. Student performance on these problems, dealing with issues like number of bits in a sample or the Nyquist theorem, was particularly strong, especially considering that these are not topics that are typically emphasized in CS1 courses.

7. Questions still unanswered

The pilot offering of the course offered glimpses of a promising future for media literacy in the computer age. We seem to have reached an audience that may otherwise have been left behind as broader, more computationally intensive forms of literacy become important. But does familiarity with media computation actually affect students' abilities to communicate through various media? Does computational know-how transfer to other contexts? In the future, we would like to investigate relationships between general media computation skills and students' abilities to learn media-specific applications like PhotoShop or to use digital audio equipment. The advantages of traditional, text-based literacy in our society are clear. When we can define the advantages of media literacy with the same confidence and clarity, we will be able to better motivate and engage students.

Does media computation have any affect on students' long-term decisions regarding computer science learning? In particular, are female students who take media computation as an introduction to computer science more likely to pursue advanced learning than those who take a traditional CS course? We hope to continue this research in the form of a longitudinal study; following female media computation students throughout their undergraduate careers. We are exploring the possibility of an advanced media computation course, and have hopes that the development of such a course will lead to record numbers of non-CS, non-engineering students taking elective computer science at Georgia Tech. In addition, we are in the process of exploring a high school level media computation course. While it often does, media literacy should not begin at the university level. All high school students can benefit from a positive experience with computer science and we believe that female high school students have as much (if not more) to gain from an improved introductory computer science experience as their undergraduate counterparts.

Finally, if the course succeeds at Georgia Tech with the current instructor, can it succeed at Georgia Tech with other instructors and at other institutions? Two other institutions have already implemented their own versions of the media computation course that use Georgia Tech material as a foundation. We plan to collect data from other institutions that implement courses using the materials developed at Georgia Tech, and from future sections of Media Computation at Georgia Tech that will be taught by a variety of instructors.

8. Conclusions

Literacy is a hard-won competency. Those who are literate read and write with a natural ease that belies the effort they put into learning to read and learning to write. For college students, the difficulty of achieving media literacy is an even more difficult feat, competing with other interests and other demands on their attention. Media computation is not a panacea for the ills of computer science education and it does not make learning to program an easier accomplishment. Still, results from this pilot course offering indicate that media and computation together provide a motivating framework for many students, that it may encourage some students to excel who would otherwise prefer not to try.

9. References

[1] A. diSessa, *Changing Minds: computers, learning and literacy*, Cambridge, MA: MIT Press, 2000.

[2] A. Kay and A. Goldberg, "Personal dynamic media," *Computer*, 1977, pp. 31-41.

[3] M. Resnick, A. Bruckman and F. Martin, "Pianos, not stereos: creating computational construction kits," *Interactions*, vol. 3, no. 6, 1996, pp. 41-49.

[4] S. Papert, "Situating constructionism," *Constructionism: research reports and essays, 1985-1990,* Harel, I. and Papert, S. Eds.Norwood, N.J.: Ablex Pub. Corp., 1991, pp. 1-11.

[5] E. Soloway, M. Guzdial and K. Hay, "Reading and writing in the 21st century," *Communications of the ACM*, vol. 36, 1993, pp. 23-27.

[6] S. Papert, *Mindstorms: Children, Computers and Powerful Ideas*, New York, NY: Basic Books, 1980.

[7] J. Comer and R. Roggio, "Teaching a Java-based cs1 course in an academically-diverse environment," *SIGSCE'02*, pp. 142-146, 2002.

[8] N. Herrmann, J.C. Popyack, Zoski, Paul, C.D. Cera, R. N. Lass and A. Nanjappa, "Redesigning introductory computer programming using multi-level online modules for a mixed audience," *Eighth Annual Innovation and Technology in Computer Science Education (ITiCSE)*, 2003.

[9] M. Urban-Lurain and D. Weinshank, "Is there a role for programming in non-major computer science courses?," *30th ASEE/ IEEE Frontiers in Education Conference*, 2000.

[10] N. C. Statistics, *Postsecondary Institutions in the United States: Fall 2000 and Degrees and Other Awards Conferred*, Washington DC: U.S. Department of Education, 2001.

[11] N. Nagappan, L. Williams, M. Ferzil, E. Wiebe, K. Yang, C. Miller and S. Balik, "Improving the CS1 experience with pair programming," *SIGCSE*, pp. 359-362, 2003.

[12] D. Guerer and T. Camp, *Investigating the Incredible Shrinking Pipeline for Women in Computer Science: final report*, 2001.

[13] J. Marks, W. Freeman and H. Leitner, "Teaching applied computing without programming: a case-based introductory course for general education," *32nd SIGCSE technical symposium on Computer Science Education*, pp. 80-84, 2001.

[14] G. W. Zimmerman and D.E. Eber, "When worlds collide! An interdisciplinary course in virtual-reality art," *Thirty-second SIGCSE Technical Symposium on Computer Science Education*, pp. 75-79, 2001.

[15] W. Dann, T. Dragon, S. Cooper, K. Dietzler, K. Ryan and R. Pausch, "Objects: visualization of behavior and state," *ITiCSE 2003: Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, pp. 84-88, 2003.

[16] N. Kock, R. Aiken and C. Sandas, "Using complex it in specific domains: developing and assessing a course for nonmajors," *IEEE Transactions on Education*, vol. 45, no. 1, 2002, pp. 50-56.

[17] G. I. Planning, *Enrollment*, <http://www.irp.gatech.edu/apps/Enrollment>.

[18] J. Margolis and A. Fisher, *Unlocking the Clubhouse: women in computing*, Cambridge, MA: MIT Press, 2002.

[19] A. E. Education, *Tech-Savvy: educating girls in the new computer age*, Washington DC: American Association of University Women Educational Foundation, 2000.

[20] S. Turkle and S. Papert, "Epistomological pluralism and the revaluation of the concrete," in *Constructionism* Harel, I. and Papert, S. Eds. Ablex Publishing Corp., 1992, pp. 161-191.

[21] Committee on Learning Research and Educational Practice, *How People Learn: bridging research and practice*, no. 09/04/2002, Donovan, S. et al. Eds. Washington DC: National Academy Press, 1999, pp. 9-22.

[22] Z. Kersteen, M. Linn, M. Clancey and C. Hardyck, "Previous experience and the learning of computer programming: the computer helps those who help themselves," *Journal of Educational Computing*, vol. 4, no. 3, 1988, pp. 321-334.

[23] H. Taylor and L. Mounfield, "Exploration of the relationship between prior programming experience and gender on success in college computer science," *Journal of Educational Computing Research*, vol. 11, no. 4, 1994, pp. 291-306.

[24] B.C. Wilson and S. Shrock, "Contributing to success in an introductory computer science course: a study of twelve factors," *SIGSCE'01*, pp. 184-188, 2001.

[25] R. Kozma, "Will media influence learning? reframing the debate.," *Educational Technology, Research and Development, 5th EARLI Conference*, no. 2, pp. 1-31, 1993.

[26] E. Soloway, J. Bonar, & K. Ehrlich. “Cognitive strategies and looping constructs: An empirical study.” *Communications of the ACM*, vol 26, no11, pp. 853-860, 1983.