Media Computation to Motivate Women and Non-Majors in Computer Science

PI: Mark Guzdial

Co-PI: Blair MacIntyre

June 18, 2003

Project Summary

The problem being addressed by this proposal is the disinterest in computer science exhibited by large groups of students, especially non-CS-majors and women—a particular problem at institutions like Georgia Tech where an introductory computing course is required.

We just had a trial offering of a course in *Introduction to Media Computation* aimed at non-CS-majors. 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 have gathered external interest in the course, including a trial offering at a 2-year college and a book contract.

We now propose (a) growing the course to a full-scale rate (of more than 600 students per year) with multiple teachers and at other institutions, (b) assessing the course more objectively, (c) introducing a longitudinal assessment to study how the students use their new knowledge and interests, and (d) introducing a second course to make *Media Computation* a viable path into computer science.

Intellectual Merit: The intellectual merit of the proposal is to scale up, study, and further develop a course holding promise for interesting women and non-majors in computer science.

Broader Impacts: The potential broader impacts is to demonstrate a viable alternative development path for introductory computer science courses that can appeal to students not usually interested in computer sciences.

Media Computation to Motivate Women and Non-Majors in Computer Science

Contents

1 Goal: A New Route to Computer Science **1**

2 Results from Proof-of-Concept Trial **5**
- 2.1 Implementation of the Course . . . 6
  - 2.1.1 Curriculum . . . 8
  - 2.1.2 Technology Development . . . 11
- 2.2 Evaluation Methods and Results . . . 14
  - 2.2.1 Evaluation of Retention . . . 16
  - 2.2.2 Evaluation of Motivation . . . 17
  - 2.2.3 Evaluation of Learning . . . 20
- 2.3 Dissemination and Self-Sustained Distribution . . . 22

3 Project Plan **23**
- 3.1 Development Plan: Completing one course and creating a path . . . 24
  - 3.1.1 Creating a path . . . 25
  - 3.1.2 Media Computation for majors . . . 26
- 3.2 More objective, credible evaluation . . . 27
  - 3.2.1 Longitudinal evaluation . . . 28
- 3.3 Dissemination and self-sustaining distribution . . . 29

4 Conclusion: Deliverables **30**

5 Budget Justification **34**

6 Facilities **35**

Media Computation to Motivate Women and Non-Majors in Computer Science

1 Goal: A New Route to Computer Science

Computer science departments are not currently successful at reaching a wide range of students who are taking introductory computer science. The evidence for this statement includes international studies of programming performance [19], declining retention rates [11], and failure rates sometimes as high as 50% [28]. This comes at a time when the need for Information Technology (IT) professionals is growing [8].

At Georgia Institute of Technology ("Georgia Tech"), all students are required to take an introductory course in computing, including programming skills. Our traditional CS1 course, the only one that met the requirement before our new course[1], is undoubtedly one of the most unpopular courses on campus, especially among those *not* in explicitly computing-related fields. While this is certainly a problem for the College of Computing at Georgia Tech (where the course has its academic home), it points towards a larger problem for the field.

Alan Perlis in April 1961 made perhaps the first argument that programming should be part of a liberal education for *all* students. If Calculus is the study of *rates*, and that's important enough to be part of the liberal education, then so should computer science. Perlis argued that computer science is the study of *processes*, which is certainly relevant to even more fields than those concerned with rates. The argument has been echoed and

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

strengthened over the intervening years—by Seymour Papert arguing for a programming as a way of learning about learning [23][24], to Andrea diSessa's arguments for "computational literacy" as a critical component of many fields [6]. As long as non-CS-majors have such a dislike for computing, the hope is diminished for computer science as an accepted part of a liberal education and for computing generally to meet its potential for intellectual impact across the range of disciplines, not just in computational science and engineering.

Introductory computer science classes are also not meeting the needs of women. The rates at which women take computer science classes are falling. While the factors causing women to avoid computer science (and IT careers in general) are multi-faceted, the curriculum does play a significant role [18]. 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 [26]. Additionally, women undergraduate CS students tend to be interested in real applications of computing, as opposed to simply computing for its own sake [18][1]. If we can address the reasons why women are avoiding computer science, we may be able to interest more males, too—all those currently expressing disinterest in computing. "Women in engineering programs are kind of like 'canaries in the coal mine"', said Stephen W. Director, Chair of the Engineering Dean's council. "If women do well in a program, most likely

everyone else will also do well in the same type of program." [2]

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

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. 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. We have gathered external interest in the course, including a trial offering at a 2-year college (Gainesville College) and a book contract with Prentice-Hall.

We do not believe that media computation is the *only* context in which students not traditionally attracted to computer science might succeed. For example, we have described another potential CS1 course organized around information management in which students

[2]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

might build Web harvesting programs and data visualizations [14]. 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. The Media Computation project is a trial and model of that approach.

In this proposal, we describe our proof-of-concept trial offering of the course (funded by NSF CCLI grant #0231176) and its evaluation results. We then describe our proposed plans under this project:

- To continue development of the course as it ramps up to some 600 students/term, has multiple teachers, and is adopted by more institutions. In addition, we are developing Java-based materials to be used in more traditional CS courses.

- To assess the course more objectively. Dr. Donna Llewellyn, Director of Georgia Tech's Center for Enhancement of Teaching and Learning, has agreed to be the external evaluator of the project.

- To explore the impact of the course on the students longitudinally. We want to learn (a) how the course impacts the students' further exploration of computer science (e.g., do they take more CS classes?) and (b) how the course impacts their use of computation in their own disciplines? For example, might we expect students to learn Photoshop better or more easily once they understand better what Photoshop filters are doing?

- Finally, we plan to develop a second course to make Media Computation a viable path into the degree. Currently, students who take *Introduction to Media Computation* and are interested in taking more computing need to start over with our traditional CS1.

While the Media Computation course meets the requirements of a standard CS1 [2], our traditional CS1 covers most of a CS1 and CS2 curriculum. We'd like to create a second course so that students can then enter into the rest of the CS curriculum without starting over.

2 Results from Proof-of-Concept Trial

Introduction to Media Computation (CS1315 at Georgia Tech) is our new introduction to computation and programming contextualized around creation and manipulation of media. 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 by most 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 [21].

- We attempted direct comparisons on problems on exams between our traditional CS1, a prototype engineering CS1 course, and the Media Computation class. We found that such comparisons are fraught with difficulties. Nonetheless, student performance in a problem described in the literature on CS education was comparable between the Media Computation and traditional CS1 students.

- Students found the course relevant, and the majority of respondents to a final survey indicated that they would like to take a second course if it were offered.

- The course met its goal of getting female students to find CS interesting, relevant, and creative where they hadn't previously.

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, and who was successful.

- Collaboration may have played as much 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 and over 20% mentioned collaboration in general.

- The technology built for the course was more effective than we anticipated, given the pilot nature of the course and the software.

- In general, students from Georgia Tech's *Ivan Allen College for Liberal Arts* were the most positive about the course (e.g., had the largest percentage of students who would recommend it to others, who planned to use programming in the future, and who planned to take further CS courses).

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. Several of the academic units at Georgia Tech emphasize that multimedia literacy is a critical component of educated professionals in the future [31]. Multimedia is a real applications

domain—especially true at Georgia Tech, with Turner Broadcasting, CNN, and the Cartoon Network based within a few miles of campus. Multimedia is clearly not computing for its own sake. Further, the multimedia approach allows us to give students assignments that are open-ended and creative. 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 are:

- All media are being published today in a digital format. Digital formats are amenable to manipulation, creation, analysis, and transformation by computer. Text can be interpreted, numbers can be transformed into graphs, video images can be merged, and sounds can be created. We call these activities *media computation*.

- Software is the tool for manipulating digital media. Knowing how to program thus becomes a communications skill. If someone wants to say something that her tools do not support, knowing how to program affords the creation of the desired statement.

- Core computer science concepts can be introduced through media computation. For example, programs can get large and cumbersome. Abstraction is our tool for managing program complexity and allowing programs to become even larger yet more flexible. However, computing has limitations. There are some programs that cannot complete in our lifetime, and knowing that these limitations exist is important for technological professionals.

We used the programming language *Jython* for the course. Jython[3] is a variation of the Python[4] programming language which has been designed to be easily used by novices and non-technical users. Jython is a variation of Python written in Java. Jython can instantiate and subclass Java classes, and Jython can be used for virtually anything for which Java can be used. We considered other languages (e.g., Java and Scheme), but selected Jython based on survey responses from students and teachers [12]. Jython's design is based on the research findings on studying novice programmers (e.g., [29], [20], [22]).

2.1.1 Curriculum

The semester outline of the course appears below. The course was offered as a three credit hour course with five weeklong lab activities, six homeworks, three in-class examinations, and two take-home examinations (programming assignments) with optional recitations. 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.

- Week 1: Introduction to the course and the argument for why media computation. Introduction to variables and functions, in the context of playing sounds and showing pictures.

- Weeks 2–3: Pictures as a media type, including psychophysics (why don't we see 1024x768 dots on the screen?), looping to change colors with a simplified **for** loop

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

(Figure 1), conditionals to replace specific colors, then indexing by index numbers to implement mirroring, rotating, cropping, and scaling.

```
def greyScale(picture):
  for p in getPixels(picture):
    intensity = (getRed(p)+getGreen(p)+getBlue(p))/3
    setColor(p,makeColor(intensity,intensity,intensity))
```

Figure 1: An example Jython program using our API to convert a picture to greyscale

- Weeks 4–6: Sound as a media type, including psychophysics (how human hearing limitations make MP3 compression possible), looping to manipulate volume, then indexing by index numbers to do splicing and reversing of sounds. Include discussion of how to debug and how to design a program, as those issues arise. One lecture on additive and FM sound synthesis.

- Week 7: Text as a media type: Searching for text, composing text, reading text from a file and writing it to a file. An example program parses out the temperature from a downloaded weather page.

- Week 8: Networks, including making the temperature-finding program work from the "live" Web page. Introduction to HTML.

- Week 9: Discuss media transitions. Moving from sound to text and back to sound again. Using Excel to manipulate media after converting it to text.

- Week 10: Introduction to databases: Storing media in databases, using databases in generating HTML.

- Week 11: Movies: How persistence of vision makes animations and movies possible, generating frames using the various techniques described earlier in the semester, manipulating whole directories of files.

- Week 12: "Can't we do this any faster? Why is Photoshop faster than Python?" Introduction to how a computer works (e.g., machine language), and the difference between an interpreter and a compiler. Algorithmic complexity and the limits of computation.

- Week 13: "Can we do this any easier?" Decomposing functions, modularity, and functional programming (map, reduce, filter, and simple recursion).

- Week 14: "Can't we do this any easier?" Introduction to objects and classes.

- Week 15: "What do other programming languages look like?" Brief overview of JavaScript and Squeak.

Student homework assignments started with simple Photoshop-style filters (Figure 2) (average score 92% with a standard deviation of 22.1). We moved on to sound, and students' first take-home exam was to splice and reverse sounds (Figure 3) (average score 92.1% with a standard deviation of 15.3). Some of the assignments invited creativity, such as the third homework which required creation of a collage, but without concern for *what* was in the collage (average score 86.3% standard deviation 26.0). These homework results became sizable–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

Write a program named **hw1** to accept a picture as input, and change its pixels as follows:

- Set the green component to 125% of its current value
- Decrease the blue by 25%
- Decrease the red by 75%

Figure 2: First homework assignment by *Media Computation* students

At the Feb. 7 class, we recorded this sound *thisisatest2.wav* ("This is a test."). Using MediaTools, we found the end points for each of the words in the sound: (Table omitted) Write a function **backSpliced** that splices the last word ("test") into the front of the word, but *backwards*, so that the result is:"Tset is a test."

Figure 3: First take-home exam assignment by *Media Computation* students

lines of code. Several students posted their collages in the shared collaboration space as a creative and social space (Figure 4). The final homework assignment involved programmed Web access and creation of animations (Figure 5) (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.

2.1.2 Technology Development

Three different software technology developments were used in the course:

- There was no Jython development environment when we started this project. We have developed *JES* (Jython Environment for Students), modelling it after the popular DrScheme environment [7]. Development of JES also included development of a simplified API for accessing the Java media library. We were pleased to receive *no* reports of crashes or failures of JES over the course of the semester.

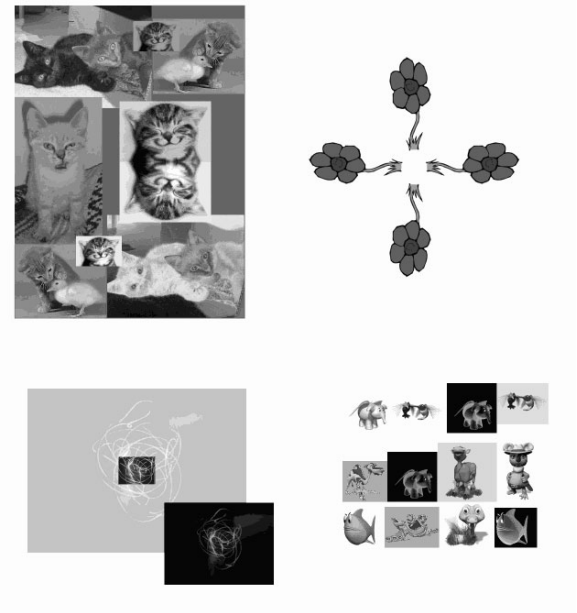

Figure 4: Four student collages shared from the third homework assignment

http://www.cnn.com is a popular news site. You are to write a function (named "hw6") that will input a directory as a string then:

- Visit *http://www.cnn.com* and pick out the *top three* news stories headlines in the **More News** section.
- Create a ticker tape movie on the 640x480 canvas of all three news stories. Have one come across at y=100, another at y=200, and the third at y=300. Generate 100 frames, and don't have the ticker tapes move more than 5 pixels per frame. Store the frames to files in the input directory.

Figure 5: Final homework assignment by *Media Computation* students

- Students also needed tools for exploring media: Recording and viewing sounds (and FFTs), investigating the RGB values of individual pixels, and assembling movies and bursting movies into individual frames. We created a cross-platform *MediaTools* based on Squeak [9].

- We used our existing collaboration technology, CoWeb [13, 10], 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 [25]. We have had good success in the past using our collaboration tools to support sharing of multimedia artifacts to encourage critique and motivate learning [15][5][33].

We have pending a proposal to the NSF CISE Educational Innovations program to fund further development of the technology. In particular, we hope to merge the MediaTools and JES functionalities so that the MediaTools become effective as debugging tools. In our observations of students using JES and MediaTools, virtually no use of MediaTools occurred at all. Currently, students create pictures and sounds, but to study how individual pixel values get set or to see a visualization of the generated sound (e.g., were any samples generated, or is the volume too low to be heard?), students have to save the media out to files, then start up MediaTools and open them there. We hope to integrate media exploration into the programming environment so that students can explore more easily their output media.

CoWeb was particularly successful in the class.The students made extensive use of a collaborative website [5]. For example, each of the assignments (homework, labs, and take-

[5]at `http://coweb.cc.gatech.edu/cs1315`

home exams) had Q&A pages associated with them where extensive discussions took place. For each exam, a review page was posted where students could answer sample questions, ask questions about the questions or concepts, and critique each other's remarks. Each week, a new general "Comment on the week" page was put up for general feedback and discussion, with leading questions on issues in the course.

The CoWeb became a central place for question-asking and discussion. We know that that's a critical role in CS1, since comfort in asking questions has been highly correlated with attrition in CS1 [32]. From interviews, we know that the CoWeb increased that comfort.

> **Q. Have you consistently felt comfortable asking questions?** Student D: 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.
>
> **Q. Do you think the CoWeb is beneficial?** Student D: 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.

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 this 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 PI and the assessment team[6] 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 had demographic data which we used to address the retention question. We had 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 ($n = 75$).

- We attempted to develop and distribute polymorphic problems (same problem, but changed in language or other minor details for the course context) that were offered in all three CS1's on exams–for example, on computing an average test grade and determining an appropriate letter grade, and on the infamous Rainfall Problem [30].

- We interviewed a sample of female students in the *Media Computation* class on their attitudes toward computer science and the course. We had seven interviews in total–

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

three early in the semester, and all three were interviewed again at the end of the term along with another student at the end.

- We also observed volunteer students working on their homework so that we could have a sense of strategies and problems, especially with the technology. There were 11 observation sessions, including 6 with just one (male) student to track development. The other observation sessions were all with female students.

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 is rather negative. Completion rates of less than 30% is not uncommon in CS1 [28]. In an email survey on the ACM SIGCSE members list, course success retention rates of less than 50% were reported at several schools [11]. The "best-practice" comparison that we have for non-majors, in a pair-programming CS1 class, had a 66.4% course success retention rate [21] (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.

Our course completion retention rate for Spring 2003 was 98%. Our course success retention rate was 89%. During that semester, our 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 [17] [27] [3].
- 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 ("data-first") approach has potential for attracting potential CS majors.

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

- "I like the feeling when I finally get something to work."

- "Very applicable to everyday life."
- "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!!!"
- "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.

When we asked students "What is something interesting, surprising, or useful that you learned?" we found that students appreciated the relevance of the course and even found the computer science interesting (again, all female respondents):

- "The most useful things I have learned are the basics about computers and pictures/sound...interesting and useful to real life applications."
- "Just general concepts about programming. It's pretty logical, sort of like in math, so it's understandable."
- "Programming is fun and ANYONE can do it!"

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, into something that they could imagine themselves doing in the future. On the final survey, 30% of the students indicated that they believed that they will program again in the future. Table 1 looks at these final survey results by College of major. In interviews, students talk about programming in the future.

> **Interviewer: What do you think about the homework galleries on the CoWeb?**
>
> Student A: 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.

	N respondents.	Will use programming again in the future.	Feel they learned to program.	Plan to take more CS.	Would recommend 1315 to friends who didn't have to take it.
Architecture	8	12.5%	50.0%	0.0%	37.5%
Ivan Allen (liberal arts)	24	41.7%	91.7%	17.4%	82.6%
Management	19	21.1%	68.4%	5.3%	68.4%
Science	3	33.0%	100%	0.0%	50.0%

Table 1: *Media Computation* student responses on final survey by College of major

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.

2.2.3 Evaluation of Learning

We developed several problems that we attempted to put on all three courses' exams and quizzes. Unfortunately, logistical considerations such as different rates of development and sequencing of concepts in each course and different times and numbers of exams and quizzes 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 assistant to such an extent that it is difficult to compare them. 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 could 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 [30]. 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 [19].)

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 get the raw problems to code ourselves and compare. On the first test (which stipulated the use of tail recursion), the result 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 bottomline is that it's still an open question how much the Media Computation students learned about programming 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 similar tutorial and a paper on the planning and development of the Media Computation course was accepted for ITiCSE 2003 [12], which will be presented in July 2003.

- Guzdial spoke on the course at a meeting of representatives from all the CS departments of the University System of Georgia. Several schools expressed interest in adopting the course. Charles Fowler of Gainesville College, a two-year college, is offering the course as a trial this summer, with plans to offer two sections of the course in the Fall (letter of support included). Guzdial was also invited to speak at Kennesaw College and Augusta State College, each four year colleges. Guzdial is actively working with Kennesaw on their plans to trial the course.

- Guzdial also gave a talk at DePauw University on the course.

- Finally, Prentice-Hall has signed a contract with Guzdial to produce a book *Introduction to Media Computation* with no completion date at this time.

3 Project Plan

While the results from the pilot offering of the course have been quite strong, there is still much to do.

- We view our results with some skepticism. The PI was also the teacher for the course. There was probably a strong Hawthorne Effect (though that's not necessarily bad [4]) given the trial offering of the course.

- The course materials have now worked *once* at the developer's institution. We will clearly need to adapt the course materials as we get feedback from other institutions.

- We are very interested in the longer term effects of the course. Do students take more computer science? We would like to provide that second course that students state that

they're interested in, and thus provide an entry way into the rest of the CS curriculum. Do some students become computer science majors? How does the course impact their use of computation in their own disciplines?

3.1 Development Plan: Completing one course and creating a path

We plan to continue developing *Introduction to Media Computation* over the proposed three year period. The development of the course will continue to be led by the PI, Professor Mark Guzdial. There are several challenges that we will be facing that will lead to changes in the course materials.

- The course will be ramping up in size. During the Fall 2003 semester, there will be two sections of 120, and three in Spring 2004. The initial pilot of 120 could be taught with incomplete materials by responding to many questions. As the class ramps up, the materials need to be better.

- The course will be taught by others. In the Spring 2004 semester, Professors Blair MacIntyre and Colin Potts will each teach sections of the course (along with Guzdial), and most probably a full-time instructor will teach a section in Summer 2004. The course materials will need to be adaptable easily by other instructors.

- The course will be taught elsewhere. Currently, it's being taught at Gainesville College with close cooperation from Georgia Tech, but eventually, it will be taught by others

without our knowledge. We want to evaluate the initial offerings elsewhere so that we can learn what needs to be changed for other audiences.

3.1.1 Creating a path

In addition, we want to develop a second course that will serve as a follow-on to our Media Computation course. Our plan is for Media Computation plus the second course will be accepted as a pre-requisite for our CS2, so that students can use Media Computation as a path into the major.

The definition of the second course falls out from what is commonly in a CS1 course according to the ACM/IEEE Computing Curriculum 2001[2]. 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.

We would use a media-oriented approach for some of the course, but not just media. The second course would also be a second course for the prototype Engineering CS1, again, as a pathway into computer science. The Engineering CS1 course plus the second course would also meet the pre-requisite for our CS2. In general, the second course will be a *data-first* approach, as the Media Computation course is.

Undergraduate computer science courses tend to emphasize the processing of any kind

of data at all. In a sense, it's *data agnostic*—all data should be treated exactly the same. In this way, focus can be shifted to abstracted data representations laid on top of the data. However, most non-CS-majors come to computers because *they want some processing of data of interest.* The focus for them is on their data. By paying attention to the data that the students care about, we may be able to increase their motivation for learning programming. In the terms of the ACM/IEEE Computing Curriculum 2001, our approach is "data-first"—we start from the data that students care about, then introduce computing as a way of creating, manipulating, and transforming the data that they care about.

Co-PI Professor Blair MacIntyre will be leading the effort to develop the second course during the second year of the grant, and working with us on evaluation during the third year of the grant. Professor MacIntyre has a strong interdisciplinary research background, especially in applications of computer science to liberal arts concerns, and a strong interest in undergraduate education. Professor MacIntyre's research is in augmented reality, and he does extensive work with faculty in our Ivan Allen College of Liberal Arts who are developing new media. In particular, Professor MacIntyre works with Professor Jay Bolter in digital storytelling with augmented reality [16].

3.1.2 Media Computation for majors

We are already developing Java versions of some of our materials, and using them (in our recent SIGCSE2003 workshop). We view the Java materials as being more likely to be adopted in courses aimed at CS majors. Our goal is not to change all CS1 courses into Media Computation, but to make it easier to include media-focused projects that motivate

students into Java-based CS1 courses. We believe that Media Computation may engender similar excitement among CS majors as what we're seeing among non-majors, and may help to raise retention rates in introductory CS courses among majors as well. We agree with the quote from Stephen Director earlier—women and non-majors may be "canaries in the coal mine" and what works better for them might work better for men and CS majors, too.

We plan to continue developing the Java materials, and in years two and three of the proposed project, develop these into a book for a course supplement. These materials will be available throughout the period of the grant on our website and in workshops, and we will seek out a publisher during year three for sustained dissemination of the materials. Prentice-Hall has expressed interest in the Java version of the book.

3.2 More objective, credible evaluation

The current evaluation will become more credible simply by continuing the current evaluation protocols! We would hope to have more success on the learning evaluation when it's across multiple terms, i.e., it was harder to coordinate among the other course instructors and TA's since it was just a single term effort. As the class moves out of it's trial period, and as others start to teach the class, we will probably see some diminishment of the euphoria surrounding the first offering of a CS1 course for non-majors. In the next year to two years, we should see results that better reflect the effectiveness of the course. We also plan to continue comparisons with the existing, traditional CS1 course.

We also plan to expand our interviews to males, as well as females. Though we're pleased

to see that we are meeting the goal of keeping females' interest, we hope that we are not achieving that goal at the expense of the men! We want to understand, generally, the gender impact of the course.

The evaluation will also become more objective and credible by separating the PI from the evaluation process. For that reason, evaluation will be taken over by the Director of Georgia Tech's Center for Enhancement of Teaching and Learning (CETL), Dr. Donna Llewellyn. Dr. Llewellyn has extensive experience in evaluation and in studying issues of gender at a technical institution. The PI and his team will be involved in design of the studies, but will not be overseeing the data collection and analysis.

We also plan to work with schools adopting our course, who are willing to be part of our evaluation. Gainesville College is working with us now, using similar surveys to ours, so that we can evaluate how well the class is working in other situations, and can adapt the materials accordingly.

3.2.1 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 Dissemination and self-sustaining distribution

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. For SIGCSE2004, we plan to submit papers on the course results overall, on the female attitudes toward the course, and on the comparison between the three CS1 courses that we studied this year. Depending on acceptances and travel budgets, we plan to present at one-to-two 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. We plan to continue developing the materials during year one and publishing the book sometime in year two of the proposed project. As mentioned, we also plan to seek a publisher for our Java materials that we plan to develop in years two and three.

4 Conclusion: Deliverables

At the end of the funding period, several resources will be available publicly to others who would like to use this method at their own institutions, such as the books, lecture slides and technologies, as well as the course planning website, where the rationale for the course decisions is made explicit, and the assessment instruments and evaluation results. All of these materials will be made available at the development website, at a project website that we are setting up, and at the course CoWeb.

We believe that Media Computation is a viable approach to making CS more relevant and improving retention rates in introductory computing courses. It's not the only approach, and we see that our *process* of development (which has involved faculty advisors from across campus, on-line surveys of students, with close ties to evaluation) is part of what we are studying and disseminating. That's the point of our ITiCSE paper [12] and a recent SIGCSE invited editorial [14]. We hope that, by developing this approach, we can improve the success of both non-majors and majors in computer science.

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

[4] Brown, A. L. Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences 2*, 2 (1992), 141–178.

[5] 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.

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

[7] 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.

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

[9] Guzdial, M. *Squeak: Object-oriented design with Multimedia Applications*. Prentice-Hall, Englewood, NJ, 2001.

[10] 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.

[11] Guzdial, M. Summary: Retention rates in CS vs. institution. Message posted on ACM SIGCSE moderated members list, Georgia Tech, April 23 2002.

[12] 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.

[13] 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.

[14] 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.

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

[16] MacInytre, B., Bolter, J. D., Moreno, E., and Hannigan, B. Augmented reality as a new media experience. In *International Symposium on Augmented Reality* (2001), pp. 197–206.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

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

[26] 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.

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

[28] 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

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

[30] 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.

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

[32] 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.

[33] 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 three year project.

- One summer month of the PI's time in Year One, and 0.75 of a summer month in each of Years Two and Three to develop the course materials (both for the Media Computation class and the Java-based materials) and work with evaluation,
- One summer month of the co-PI's time in Years Two and Three to develop the follow-on course and work with evaluation on the pathway into CS through Media Computation,
- Two graduate student research assistants to focus on evaluation with Dr. Llewellyn,
- One graduate student research assistant to help development of the course materials,
- Our costs for these personnel include fringe, computing charges (for support within the College of Computing), and graduate student tuition,
- Travel support for attending conferences to disseminate materials and results.
- Materials and supplies, 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.