Introduction to Media Computation: A new CS1 approach aimed at non-majors and under-represented populations

PI: Mark Guzdial

October 2002

Project Summary

Studies of why students are failing in computer science (or even not attempting it) point out that one of the reasons for the failure are computer science courses described as “narrow,” “lacking creativity,” and “technology for its own sake” (quotes from a 2000 AAUW report and papers at a recent ACM SIGCSE conference). Early undergraduate computer science courses typically emphasize highly-technical problems with few connections to real world examples which allow little room for “tinkering” (AAUW) or creative exploration by students. At the same time, Moore’s Law (e.g., high-speed computers at low cost) make it possible for CS1 students to tackle interesting and creative media computation projects. For example, simple algorithms for splicing sounds or chromakey “blue screen” effects are well within the range of a CS1 course.

At Georgia Tech, we are creating a new CS1 course, “Introductory to Media Computation,” aimed explicitly at non-majors and implicitly at improving the motivation of women and other under-represented groups to take computer science courses. In this proposal, we explain our efforts at creating and establishing this course. We ask for support (a) to integrate media analysis tools in the students’ integrated development environment, (b) to develop Java-based versions of our materials for use by the greater majority of lower-division undergraduate CS courses, and (c) to study the effects of these innovations and the ease of integration of media computation assignments into more traditional courses.

Introduction to Media Computation: A new CS1 approach aimed at non-majors and under-represented populations

Contents

1 Introduction: The Challenge of Recruiting and Retaining Students through CS1 1

2 The Potential for Media Computation Projects in Early Undergraduate Courses 2

3 Project Plan 3

3.1 Current State in Development of "Introduction to Media Computation" 4

3.1.1 Technical details 6

3.1.2 Relevant past NSF experience 9

3.2 Proposed: A new, integrated environment for media computation 9

3.3 Proposed: Java versions of materials 10

3.4 Evaluation Plans 12

3.5 Project Roles and Involvement of Students 13

3.6 Dissemination Plans 13

4 Conclusion: Deliverables and Schedule 14

5 Budget Justification 15

6 Facilities 16

Introduction to Media Computation: A new CS1 approach aimed at non-majors and under-represented populations

1 Introduction: The Challenge of Recruiting and Retaining Students through CS1

"If the number of women in the IT workforce were increased to equal the number of men, even the tremendous shortages of IT workers noted in the ITAA studies could be filled. However, according to the Department of Commerce, only 1.1 percent of undergraduate women choose IT-related disciplines as compared to 3.3 percent of male undergraduates." [8], pg. 111.

Despite the economic downturn (at the time of this writing), the ACM (the largest society for Information Technology (IT) professionals) reports that the shortage of IT workers continues [39]. Thousands of IT jobs go unfilled, much as at the time of the Freeman & Aspray report [8]. When the economy recovers, we can expect the problem to only get worse. While the promise of a "New Economy" remains unmet, it's clear that we are truly in an "Information Economy" where IT workers are a too-scarce commodity.

Part of the cause of this shortage are low success rates in computer science courses, caused both by high drop-out and failure rates (sometimes as high as 30%-50% [36][12])—and by the large numbers of students who don't even attempt Computer Science. A report 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, simply 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 computer science — they didn't expect that, and that expectation was one of the issues that they had to overcome to pursue their degree [34]. Women students are dissuaded from the field by the stereotype of computer science as an asocial, uncreative activity [26].

The issue isn't simply encouraging women—though the quote from Freeman & Aspray above points out that that would be enough to deal with the IT worker crisis. "Women in engineering programs are kind of like 'canaries in the coal mine"', said Stephen W. Director, Chair of the Engineering Deans Council. "If women do well in a program, most likely everyone else will also do well in the same type of program." [1] There is certainly evidence that the "canaries in the coal mine" is an apt metaphor. Not only do men also drop-out or fail computer science courses at unacceptably high rates[36][12], but a recent multi-national study of first and second year computer science students suggests that even those who complete introductory courses successfully are not meeting our programming expectations [28].

We as a field need to show students from the introductory courses how exciting computer science is. We need to engage students and prepare them for the wide range of tasks that IT professionals engage in today. The literature on programming learning shows that *engaging* students in order to get them to spend time programming is the most significant factor in getting them to learn programming—regardless of gender [4].

This proposal is focused on creating a new kind of CS1 course centered around *media computation*, with the hypothesis that, for many students, computation is about *communication* and introducing computation in a communications context will be more engaging and lead to improved motivation. Media computation is the use of computation to

[1] Testimony by Stephen W. Director, Chair, Engineering Deans Council, American Society of Engineering Education and Robert J. Vlasic, Dean of Engineering, University of Michigan College of Engineering, to the Commission on the Advancement of Women and Minorities in Science, Engineering, and Technology Development, Washington, DC. July 20, 1999

create, modify, and transform media. Multimedia has passed the stage of it being a research project to create video effects or to generate novel sounds. The hardware and state of the art make it feasible to move from late-undergraduate/early-graduate courses down into the introductory courses. In the following sections, I expand upon our hypothesis that introducing multimedia early has the potential to improve enrollment and success rates in introductory courses, and then explain what we at Georgia Tech have done so far to act upon this hypothesis. The proposal plan has three components:

- To develop an integrated development environment for students that supports media computation;
- To develop a set of Java-based materials to improve dissemination of our approach;
- To evaluate the impact of these developments, the ease of integration of a media computation approach into traditional CS courses.

2 The Potential for Media Computation Projects in Early Undergraduate Courses

"We must broaden our horizons and think of our students not only as potential compiler or operating systems designers but also as implementors of computer-based solutions to non-computing problems." Sharon Lawrence Pfleeger in [34].

Research in Learning Sciences & Technologies shows that students must be *engaged* in order to learn material well [25][3][32]. While it's possible for anyone to memorize just about anything, deep understanding and the ability to *transfer* knowledge (apply the knowledge in a circumstance or domain different than one learned in) requires students to be motivated to explore the material and reflect on it [5][6]. While the evidence is still mixed on whether *consumption* of multimedia facilitates learning [24], there is growing evidence that *constructing* multimedia can facilitate learning [35][20]. This approach to learning is called *constructionism*, a term coined by Seymour Papert to describe the approach to learning of constructing public artifacts.

Consider how we introduce computer science to students in CS1. We'll use the popular introductory text by Deitel & Deitel [7] — not as a critique of that text, but as representative of a fairly common approach. The first program discussed in Deitel & Deitel is producing a line of text, akin to "Hello, World." The second places the text in a window. The next few programs produce numeric outputs in windows and then input numbers and generate calculator types of responses. Do these programs address students' expectations for (a) what makes computer science interesting or (b) what they will be doing in their future IT careers? Does anyone really believe that "Hello, World" is engaging to students?

Not all introductory courses in all fields can give undergraduate students the opportunity to do all the exciting things that graduates can do. If that were true, we would hardly need an entire undergraduate degree! But the more that introductory courses *can* reflect what's exciting and intriguing about a field, the more successful the courses will be at (a) motivating the students to succeed in the course and (b) motivating the students to explore the field further. It's certainly worth exploring how much of the excitement and creativity of computer science can be communicated in the introductory course.

For students today and the kinds of professions they will take on, CS1/CS2 content may not not reflect the most common kinds of activities. For example, very few IT professionals today build sort algorithms. More commonly, as Pfleeger notes in the quote at the beginning of this section, what IT professionals do today involves solving new kinds of problems in new kinds of ways. A large number of IT professionals construct multimedia as part of their careers, and that number may be increasing. However, our goal is not to prepare students for such jobs *only*. Rather, the point is that multimedia manipulation is an activity that can *engage* students into what computer science is about and why it's interesting. Once engaged, the students' potential for learning how to implement those "computer-based solutions to non-computing problems" is limitless.

Our current generation of students has been referred to as the MTV or Nintendo Generations [37]—sometimes in derision, with claims of resultant low attention spans. Perhaps the point of these students' interest in media is not that this is the only thing that these students are interested in, but that this is what these students want to *produce*[18]. Media is what many of these students think computing technology is about.

It may not be a generational issue. In 1977, Alan Kay and Adele Goldberg reported on their success in teaching programming by having students build a wide range of multimedia projects: Animations, computer music compositions, computer games, and even simple forms of music videos [23]. Kay and Goldberg report that students were excited and successful. With the increases in processor speed and decreases in memory prices in the last 25 years, these kinds of projects are even easier to implement—even inefficient algorithms produce interesting results.

We intuitively know that multimedia is engaging for students. The computer science education literature describes introductory courses around graphics effects [27] and virtual reality [41]. But rarely do these courses involve real programming, i.e., to leverage media computation as an engaging context for learning programming. Further, it may be that our current CS majors might be just as motivated by a media computation context. Most significantly, we see that multimedia production activities *do* provide for the "creativity" that the AAUW report sought [1] and has been reported as successful in the past [23]. Overall, we (those developing our media computation course at Georgia Tech) believe that such an approach has potential to engage students not currently motivated to study computing, such as women and other under-represented populations.

We can't know for certain that a multimedia construction approach to introductory computer science courses will be any more engaging than our current approach for women and non-majors. There are at least two threats to the hypothesis:

- Students may not find multimedia production engaging, or not engaging enough to get past the cognitive challenges of learning to program. Later in this proposal, you will read evidence from our design process that gives us confidence that this will not be the case.

- Students may find the domain of multimedia too complex. To manipulate sound, for example, students need to understand issues such as sampling rates and the physics of sound.

Through our evaluation effort, we plan to test these hypotheses. Given the need, we argue that a media computation approach is well worth the gamble.

3 Project Plan

The College of Computing at Georgia Institute of Technology (*Georgia Tech*) has committed to developing the course "Introduction to Media Computation." All students at Georgia Tech are required to take a course in computing. Two courses that meet the requirement currently exist:

- *CS1321 Introduction to Computing* which is aimed primarily at computer science students, electrical and computer engineering students, and other students whose goal is to become software developers. Approximately 1200 students a semester take this course.

- *COE1361 Introduction to Computing for Engineers* which is aimed at engineering students whose goal is to be *tool builders*: Writing relatively short (less than 100 lines) Matlab scripts to solve problems for themselves or their colleagues. This course is still at a pilot stage, with approximately 90 students a semester, with a vision of ramping up to 900 students a semester.

The media computation course is being developed for a third set of students: Those for whom computation is a communications tool. The College of Architecture, Dupree College of Management, and most of the Georgia Tech's liberal arts college, Ivan Allen College, have adopted the course as an option. (Their students currently take CS1321, and their students will continue to have that option.) These students will make extensive use of computation in their professional careers, often to prepare media to meet communications goals, e.g., normalizing sounds to a maximum volume, altering colors in an image, or using chromakey "blue screen" effects to replace a background in a movie. The course is aimed at teaching computing in general and programming specifically, using exactly these kinds of examples to motivate the course.

The next section describes our current progress in developing the course. The following sections present our proposed work.

- We (the team of developers for the media computation course) have a reasonable collection of tools to support students' programming and media manipulation tasks in the class. However, they are relatively roughly implemented and are not at all integrated. This is a serious detriment in a class for non-major freshmen. We propose to develop a new integrated development environment that would support all the students' tasks.

- We expect that exercises in media computation may also motivate students in major-oriented CS1 courses. We propose to create a version of our course notes (including example programs and descriptions of media details such as acoustics and psychoacoustics) in Java to make it accessible to interested faculty teaching CS1/2 courses. Our goal is to provide a buffet of potential assignments so that an instructor can pick assignments that match their students' interests and the curriculum of the particular course.

- To evaluate our interventions and to disseminate our work widely. Our evaluation will focus particularly on:
  - Student learning and levels of motivation, contrasting with students in the traditional CS1 class at Georgia Tech;
  - Usability of our tools and the impact of the improving usability of the tools on students' success;
  - The ease with which adopting faculty integrate our materials into their classes.

  We plant to publicize our work through web sites, mailing lists, the ACM SIGCSE conference, the IEEE FIE conference, and the *Journal of Educational Resources in Computer Science.*

The following sections provide detail and examples to further explain our project plan.

3.1 Current State in Development of "Introduction to Media Computation"

CS1315 Introduction to Media Computation will be offered for the first time in Spring 2003 to a pilot class of 100 students. We will iterate on the course during Summer 2003 and implement at a full-scale in Fall 2003, with two sections of 250-300 students. It will be offered at a similar scale in succeeding terms. Currently, we are planning the course and developing course materials

The premises and core concepts of the proposed course are:

- All media are being published today in a digital format.

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

- Software is the tool for manipulating digital media. Knowing how to program thus becomes a communications skill. If someone wants to say something that her tools do not support, knowing how to program affords the creation of the desired statement. If she understands what her tools are doing, she may become a more adept practitioner, and more capable of transferring knowledge between tools.

- 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.

Our learning objectives are:

- Students will be able to read, understand, and make functional alterations to small programs (less than 50 lines) that achieve useful communications tasks.
- Students will appreciate what computer scientists do and the key concerns of that field that relate to students' professional lives.
  - Students will recognize that all digital data is an encoding or representation, and that the encoding is itself a choice.
  - Students will understand that all algorithms consist of manipulating data, iteration (looping), and making choices — at the lowest level, these are choices about numbers, but we can encode more meaningful data in terms of those numbers.
  - Students will recognize that some algorithms cannot complete in reasonable time or at all.
  - Students will appreciate some differences between imperative, functional, and object-oriented approaches to programming.
  - Students will appreciate the value of a programming vs. direct-manipulation interface approach to computer use and will be able to describe situations where the former is preferable to the latter.
- Students will be able to identify the key components of computer hardware and how that relates to software speed (e.g., interpretation vs. compilation)
- Students will develop a set of usable computing skills, including the ability to write small scripts, build graphs, and manipulate databases – not necessarily using the common tools, but in a manner that exposes concepts and enables future learning.

The currently planned week-by-week outline of lectures and topics is described in Table 1. We have developed (and are continuing to develop) a set of course notes and lecture slides that support the course. Overall, the course is designed to meet the "Imperative First" CS1 general structure and requirements in the new ACM/IEEE Computing Curriculum 2001 [2]. The order of media covered in the course is arranged to correspond to an increasing level of complexity in data structures.

- A sound is an *array* of *samples*.
- A picture is a *matrix* (two-dimensional array) of *pixels*.
- A directory structure (of media files, to process many files with a single recipe) is a *tree* of files.
- A movie is an *array* of *matrices* (frames, as pictures).

The media thus serve as a way of visualizing and making concrete (and interesting, we believe) the programs that the students are writing. Once the students are writing programs of increasing complexity, we introduce the ideas of algorithm complexity, object-oriented programming, and recursion as techniques for managing that complexity. Java is introduced only briefly at the end as a means for accessing the lowest levels of these data representations. The plan is neither to teach the students Java nor to introduce all the features of Java. Rather, only those aspects of Java that directly correspond to language elements they're familiar with will be introduced. The points are to emphasize the existence of alternative notations and that what the students have learned is applicable in other contexts.

We have been developing the course in a collaborative process with a board of faculty advisors from across campus and a team of undergraduate and graduate students developing materials. We have been using both on-line and face-to-face forums to gather input on the course—please see `http://coweb.cc.gatech.edu/mediaComp-plan`. Both student and faculty feedback has been very positive:

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

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

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

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

3.1.1 Technical details

The language for the course will be Python (`http://www.python.org`). Python is a popular programming language used today by companies including Google and Industrial Light & Magic. It's most often used for Web (e.g., CGI script) programming and for media manipulation. Python was specifically developed to be easy-to-use, especially for non-traditional programmers.

The specific version of Python that we'll be using is Jython (`http://www.jython.org`). Jython is an implementation of Python in the popular programming language Java. Anything that one can do in Java (e.g., servlets, database programming via JDBC, GUI programming via Swing) can be done in Jython. Jython *is* Python—learning one is the same as learning the other.

We chose Jython in order to enable cross-platform multimedia manipulation. We have written a set of Java classes that encapsulate the kind of multimedia functionality that our examples will require, as well as a set of Jython classes that provide a simple and useful API to those functionalities. The API was designed based on existing literature on challenges that students find in learning to program, e.g., we allow set-based manipulation of samples and pixels before more complex and general iteration structures are learned [29][30].

Our API allows for access to the samples that make up sounds and the pixels that make up pictures.

- Figure 1 is an example program using our API that converts a picture object to greyscale. It computes the intensity of a given pixel by averaging the red, green, and blue components, and then replaces the color of that pixel with a gray pixel (red, green, and blue components

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

Table 1: Outline of the Media Computation course

the same) with the same intensity. Notice that the loop in this example is phrased as a set operation—essentially, "for every pixel *p* in the pixels of the given *picture*, do...." Research on novice programming suggests that this is a simpler concept to begin with, as a way of easing into iteration.

- Figure 2 is a program that normalizes sounds to a maximum volume, by searching for the largest sample, computing a multiplier so that that sample would reach the maximum amplitude, and then multiplies all samples in the sound to raise the amplitude of the overall sound. We continue to use the simpler form of iteration here, but using multiple loops—an increase in complexity.
- Figure 3 takes a filename, then returns the sound in that file in reverse. Here, we use a more conventional `for` loop, with explicit indices in order to copy the array elements correctly.

```
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

```
def normalize(sound):
    largest = 0
    for s in getSamples(sound):
        largest = max(largest,getSample(s) )
    multiplier = 32767.0 / largest

    print "Largest sample value in original sound was",  largest
    print "Multiplier is", multiplier

    for s in getSamples(sound):
        louder =  multiplier * getSample(s)
        setSample(s,louder)
```

Figure 2: An example Jython program using our API to normalize sounds to a maximum volume

We have also created a set of tools to support the students' tasks in this course. Our first and immediate need was for some kind of development environment. Jython is a new language [33], so most developers simply use plain text editors, or make do with Python or Java development environments. We believe that non-CS major freshmen will require more support. A teach of undergraduate senior design students created our tool for students, *JES (Jython Environment for Students)* as a simple editor and program execution IDE (Figure 4). JES runs identically on Windows, Macintosh OS X, and Linux systems. Further JES development is occurring with undergraduates (mostly programming) and graduate students (developing documentation).

We also realized that our students would have a need to visualize and explore media and to prepare media for use in their programs. For example, we would like to be able to look at sounds using a variety of visualizations, record their own sounds, investigate the RGB values in pictures of their choosing, and burst MPEG movies into folders of JPEG frames for ease in manipulation. By using their own media, we hope to make the student programming assignments into a creative activity, and thus, make it more attractive to women and others dissuaded by the stereotype of

```
def backwards(filename):
  source = makeSound(filename)
  target = makeSound(filename)

  sourceIndex = getLength(source)
  for targetIndex in range(1,getLength(target)+1):
    sourceValue = getSampleValueAt(source,sourceIndex)
    setSampleValueAt(target,targetIndex,sourceValue)
    sourceIndex = sourceIndex - 1

  return target
```

Figure 3: Return the sound in the file backwards

computer science as non-creative [26][34]. Another team of undergraduate students have modified the media tools in Squeak [10][17] to create cross-platform media exploration and manipulation tools (Figure 5).

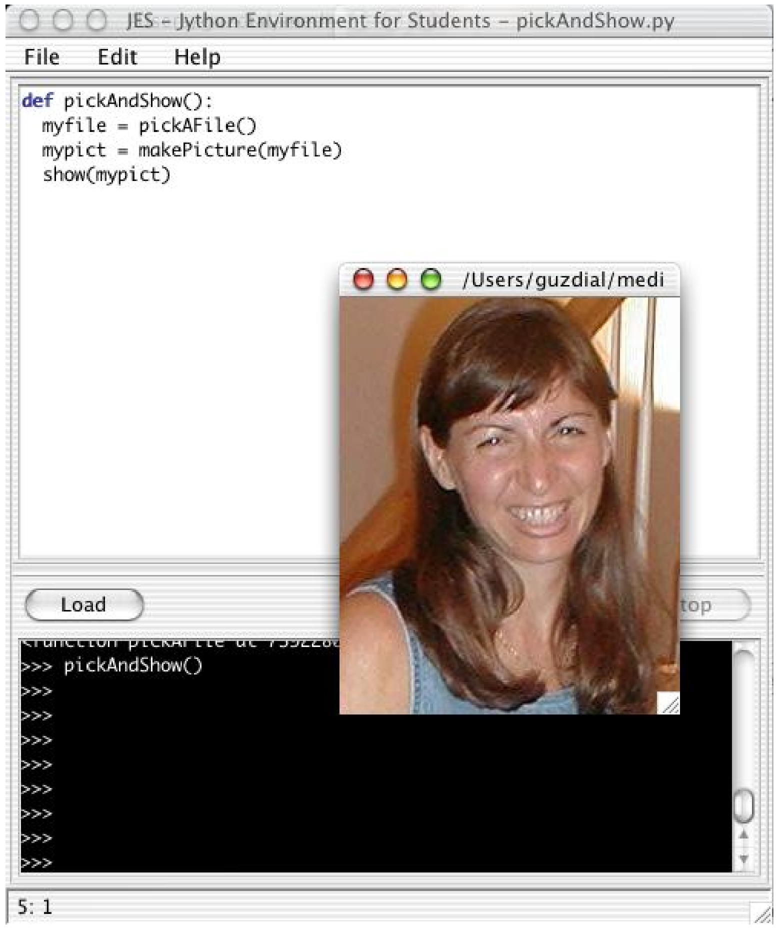
Finally, we plan to use our CoWeb collaboration tool to support a collaborative experience for students in the Media Computation course. The CoWeb has been used successfully in a variety of classes, including computer science. By encouraging students to share their creative artifacts via the CoWeb, we further erode the perspective of computer science as a loner, non-creative activity. Further, we plan to use the CoWeb to support student, e.g. asking questions about the multimedia assignments. Such support may be critical to the success of the project. A factor analysis considering a range of variables influencing CS1 success, with completion as an outcome variable, suggests that comfort asking questions is the most critical factor for succeeding in CS1 [40]. Findings suggest that Web-based collaboration tools encourage much greater participation and comfort than in-class questions [19][21].

3.1.2 Relevant past NSF experience

Guzdial was a co-PI on the *Computer Modelling for Curriculum Integration* project, funded by the NSF REPP program (REC-9814770). That project sought to use computer-supported collaborative learning to integrate the various curricular elements (computer science, mathematics, and engineering) that lead to students' understanding of computer modelling. While we had success in some domains using our collaboration support (e.g., in some computer science classes [11]), it was generally not a successful activity. Our finding was that the culture and classroom practices of mathematics and engineering courses inhibit collaboration [14]. Our focus then shifted to the faculty, working with them to develop collaboration tools that meet their goals [16] which resulted in a tool (the *CoWeb* for Collaborative Website) that has surprisingly good adoption (and even invention of new activities) by faculty into their classes [15]. The PI also has significant experience developing support materials for computer science students [13]. His dissertation work was on a scaffolded environment to support high school students learning programming and physics through development of simulations of kinematics [9].

3.2 Proposed: A new, integrated environment for media computation

JES, MediaTools, and CoWeb form a relatively complete set of functionality for the course. However, they are three separate programs with completely different interfaces. Squeak (MediaTools) and Swing (JES) even have different look-and-feels, so buttons don't even look the same. Our experience and literature in the field suggests that the cognitive dissonance and lack of usability will be a problem for these students [9][31].



```
def pickAndShow():
  myfile = pickAFile()
  mypict = makePicture(myfile)
  show(mypict)
```

```
>>> pickAndShow()
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

Figure 4: JES: Jython Environment for Students, with a graphics example running

We propose to construct a single environment with a single interface standard that integrates the programming IDE and media tools functionalities. The goal is for students to be able to record a sound and then manipulate it, or implement a graphics effect and check the RGB values of specific pixels—without having to switch environments and interfaces.

Currently, if a student wanted to look at the waveform of the sound she just produced, she would:

- Save the sound into a WAV file.
- Open the MediaTools.
- Open the WAV file, then open the sound editor.

A similar process is needed to, say, check the RGB values of a newly generated/transformed picture. We would like to integrate the analysis tools so that they can be used immediately to check the results of the students' programs.

3.3 Proposed: Java versions of materials

As mentioned, our API is built on top of a set of Java classes that encapsulate Java media functionality. Creating Java versions of our examples, course notes, and slides will be technically simple.

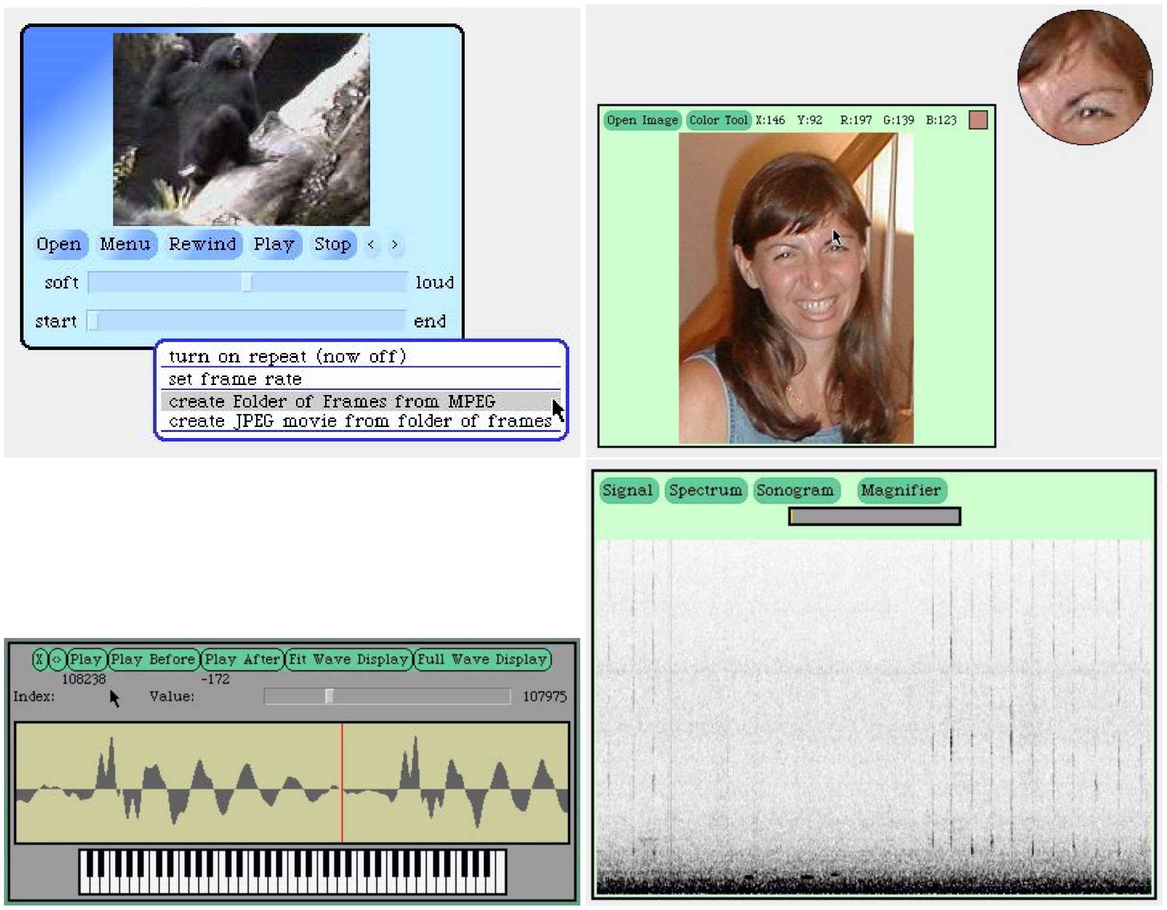


Figure 5: MediaTools: Movie tools (ul), image tools (ur), sound editing (ll), and sound views such as a sonogram (lr)

By creating Java versions, we make our media computation materials available to a much broader audience. While C++ is still the most common CS1 programming language, we expect Java to grow even greater in popularity with the move of the ETS Computer Science Advanced Placement exam to Java in the next year. (We have explored a C# version of our materials, but we have found the multimedia API available in C# to be much more complex than Java's. We may want to develop a C# version in the future, but not as part of this project.)

Our plan is not to create yet another Java CS1 nor CS2 textbook, but to create a companion set of materials that could be used to serve as infrastructure for easily implementing multimedia projects in CS1/CS2 courses (or in non-majors introductions to programming, but based in Java rather than Python), to the extent that the instructor desires. We foresee some teachers using our multimedia projects for almost all examples and assignments, while others may only pick a handful of examples and assignments to engage those students who might enjoy the multimedia. We plan to index the example assignments and sample solutions via CS1 and CS2 core curricular concepts, e.g., the knowledge units of the ACM/IEEE *Computing Curriculum 2001*.

Our hope is that the course materials become published as a book, to further disseminate the materials. Alan Apt of Prentice-Hall Publishing has already expressed interest in such a book[2]. He is already publishing textbooks (a) that use multimedia examples and (b) that support CS1 and

[2]Alan Apt agreed to allow us to re-use the letter of support that he provided for a previous version of this proposal.

CS2. He knows this kind of material and believes that there is interest in it. A letter of support from him is included in our proposal.

Further, we plan to trial these Java-based multimedia projects in our own Java classes. Our CS1 class is in Scheme, but our CS2 is in Java. Some of the array and matrix manipulation projects that students are currently doing could be easily mapped to sound and picture manipulations. By trialing these projects in our own classes, we have the opportunity to evaluate how well the integration is progressing.

3.4 Evaluation Plans

We have a pending one-year proposal with the Division of Undergraduate Education to assess student learning in our pilot study year. Our plans in that proposal are to study:

- Student learning from three perspectives: (a) whether learning objectives are met (e.g., do students understand the media-specific issues?); (b) whether student learning is similar (for common topics) to learning in our traditional CS1 class *CS1321*; and (c) whether students learn compared to literature on novice programming, e.g., on standardized problems like the rainfall problem [38].
- Student motivation, in comparison with the traditional CS1 class, but also differentiated by gender, major, and race. We know from benchmarking studies in our CS1 class that non-majors leave *CS1321* wishing never to take another CS course—a severe detriment to these students' use of computation in their future professional life, we believe. We hope to see that change with this class.

During the three year time frame of this proposal, we would hope to continue studying these issues in the scope not covered in the DUE proposal. Learning and motivation are at the core of our hypotheses. Our methods will include:

- Comparable (potentially isomorphic) problems in the media computation course final and midterm exams to those used in the traditional CS1 final and midterm exams. We have had good success using an isomorphic problem approach in the past to compare groups [13].
- Other learning objectives will also be tested in these exams. The coding of all exam problems used for research assessment will be done separately from the grading effort and anonymously, to insure a lack of conflict and maintain subject privacy.
- We plan end-of-term questionnaires on motivation issues.

Additionally, the proposed work raises two other research questions to evaluate:

- Are the tools that we are creating for the students usable, and what is the impact of that usability (or lack thereof) on student progress?
- How difficult is it to integrate media computation projects into Java-based courses?

We will address the first question with three kinds of methods, starting from the first year:

- We will invite student volunteers to do some of their work under the observation of researchers (no one connected with the course) in a private lab setting for a financial incentive. In this way, we can observe interface breakdowns and prompt students for their intentions and problems.
- We will have an anonymous feedback forum on the CoWeb through which students can give us feedback. We have used this kind of approach successfully for student feedback in the past—students do believe (correctly) that they can post in the CoWeb anonymously if they choose to.

- Finally, we will *ask* students in the final questionnaire about their satisfaction with the tools, the interface to the tools, and the impact on their performance.

The second question will be studied during the third year of the proposed project, after the Java materials have been created and are being disseminated—both internally and externally. Our focus for the evaluation will be on our internal adoption process, since we have some control over the adoption actually occurring here. Our first Java course is our *second* course, so our focus will be on use of the multimedia examples for driving data structures and algorithms issues, as well as introductions. We will invite external adoptors to complete our measures, but we can't guarantee that. Our methods will be:

- We plan to develop a journal or log for teachers and teaching assistants working on integrating the media computation projects to record effort. We plan to use the *Personal Software Process* as the model for measuring the integration effort [22]. We don't plan to have a comparison group, e.g., is it harder or easier to integrate media computation assignments and content than, say, concurrency assignments and content. However, such measures will give us a sense of the complexity.
- We will also develop a questionnaire for teachers and teaching assistants using our materials on satisfaction, how difficult the integration was, how well the students understood the content, and how successful the students' programming media computation was.

IRB panel review materials have already been submitted, and we expect to be exempted from further review.

3.5 Project Roles and Involvement of Students

Our budget requests funding for one graduate and two undergraduate students. The graduate student will be primarily responsible for executing the evaluation effort (data collection and analysis) but will also help the PI in managing the undergraduates. The undergraduates will be responsible primarily for implementing the integrated environment, but we also hope to involve some set of undergraduate students in the data analysis process. (Since our subjects are also undergraduates, perhaps peers, we do not want to involve them in the data collection process.) Though it may be unusual to plan such a development effort solely around undergraduate student programming, our experience has been overwhelmingly positive. Since we were successful in developing our original functionality with undergraduate programmers, we believe that we can continue using undergraduates for the next iteration of our tools.

The PI will be responsible for:

- Overall management.
- Designing the evaluation instruments and methods (Year 1), and overseeing the implementation and analysis of evaluations (Years 1-3);
- Developing the Java-based versions of the course materials (Years 1-2).
- Dissemination.

3.6 Dissemination Plans

The PI already publishes frequently in the computer science education community: In ACM SIGCSE (Computer Science Education) conference, the international ITICSE (Innovation and Technology in Computer Science Education) conference, and the IEEE/ASEE FIE (Frontiers in Education) conference. We will continue to publish our work in these forums as part of this project. Since very little work in this community has real research data to back it up, we feel that our evaluation effort will make our papers and materials particularly attractive in these communities. We expect to submit one-to-two papers per year in some combination of these CS Education conferences.

We will make our materials freely available on a project website, which will be referenced in our interactions on the ACM SIGCSE and other relevant (e.g., CS Ed research) mailing lists. We also plan to submit a paper and materials to the new ACM *Journal of Educational Resources in Computing*[3] for review of the material and dissemination at a broadly accessible (and indexed by the ACM Digital Library) level.

We also expect research results from our use of collaboration technologies (a) in CS1/CS2 courses and (b) in support of multimedia exchanges. We plan to submit at least one paper over the three year project plan in each of *Computer Supported Collaborative Learning Conference* and *International Conference of the Learning Sciences*.

Finally, we believe that if we are successful in turning our course materials into a book, that will be a concrete and continuing source of dissemination, even after the project ends.

4 Conclusion: Deliverables and Schedule

The deliverables that we will generate in this project will include:

- An integrated environment for students for media creation, exploration, and Python programming; and
- A supplementary book of Java media computation course materials for CS1/CS2.

Our detailed schedulefollows:

- **Year One:**
  - Guzdial develops evaluation methods and instruments.
  - Team starts implementing evaluation of learning, motivation, and usability.
  - Team starts design and implementation of integrated environment.
  - Guzdial starts creation of Java version of materials.
  - Publish the plans and results as developing in either SIGCSE and FIE with links to website.
- **Year Two:**
  - Team continues to evaluate learning, motivation, and usability.
  - Team completes integrated environment.
  - Guzdial completes Java version of materials and begins dissemination.
  - Publish further results at some combination of SIGCSE, ITICSE, and FIE with links to developing website.
- **Year Three:**
  - Team continues to evaluate learning, motivation, and usability.
  - Team evaluates the ease of integration of Java-based materials.
  - Final publications on project submitted.

[3] `http://www.acm.org/pubs/jeric/homepage.html`

5 Budget Justification

- **Personal Services**
- Guzdial is funded at one summer month per year.
- We will share one graduate student research assistant (GSRA) with other labs for hardware support.
- We will hire one GSRA for evaluation and supporting undergraduate management.
- We will hire three undergraduates to develop the integrated environment and to aid in evaluation.
- **Other**
- We are asking for travel funding to attend conferences for dissemination, increasing as we have more results to present at more places.
- We are requesting M&S funding for software and other costs.
- We are also requesting funding for GSRA tuition and charges for computing infrastructure support.

Our cost sharing component is:

- An additional month of the PI's time during year one is provided by the College of Computing (approximately $11.5K).
- Georgia Tech Technology Fee program is contributing $13K toward purchase of webcams with microphones to be provided in laboratory computer clusters and for student sign-out, so that students have the hardware to gather their own media.
- The Georgia Tech Vice-Provost for Undergraduate Education is providing $26K to support course development during Summer 2003, which includes a month of the PI's time.
- Finally, the Georgia Tech Vice-Provost for Research is providing $24.5K to purchase media stations, to be manned by teaching assistants in the media computation course, in the non-major colleges (architecture, management, and liberal arts). These stations will include high-end computers, scanners, and digital cameras to support media creation and student assistance.

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 Sony Mavica 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.

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] Bruckman, A., Jensen, C., and DeBonte, A. Gender and programming achievement in a cscl environment. In *Proceedings of the 2002 Computer Supported Collaborative Learning conference*, G. Stahl, Ed. University of Colorado at Boulder, Boulder, CO, 2001, pp. In–Press.

[5] Bruer, J. T. *Schools for Thought: A Science of Learning in the Classroom.* MIT Press, Cambridge, MA, 1993.

[6] Collins, A., Brown, J. S., and Newman, S. E. Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, L. B. Resnick, Ed. Lawrence Erlbaum and Associates, Hillsdale, NJ, 1989, pp. 453–494.

[7] Deitel, H., and Deitel, P. *Java: How to Program.* Prentice-Hall, Upper Saddle River, NJ, 1999.

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

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

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

[11] 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.

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

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

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

[15] 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.

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

[17] Guzdial, M., and Rose, K., Eds. *Squeak, Open Personal Computing for Multimedia.* Prentice-Hall, Englewood, NJ, 2001.

[18] Guzdial, M., and Soloway, E. Teaching the nintendo generation to program. *Communications of the ACM 45*, 4 (2002), 17–21.

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

[20] Hay, K. E., Guzdial, M., Jackson, S., Boyle, R. A., and Soloway, E. Students as multimedia composers. *Computers and Education 23*, 4 (1994), 301–317.

[21] Hudson, J. M., and Bruckman, A. Irc francais: The creation of an internet-based sla community. *Computer Assisted Language Learning (CALL) 15*, 2 (2002), 109–134.

[22] Humphrey, W. S. *Introduction to the Personal Software Process (sm)*. Addison-Wesley, Reading, MA, 1995.

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

[24] Kozma, R., and Russell, J. Multimedia and understanding: Expert and novice responses to different representations of chemical phenomena. *Journal of Research in Science Teaching 43*, 9 (1997), 949–968.

[25] 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.

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

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

[28] 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.

[29] Miller, L. A. Programming by non-programmers. *International Journal of Man-Machine Studies 6* (1974), 237–260. Participants strongly preferred to use set and subset expressions to specify the operations in aggregate.

[30] 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.

[31] Miller, P., Pane, J., Meter, G., and Vorthmann, S. Evolution of novice programming environments: The structure editors of carnegie-mellon university. *Interactive Learning Environments 4*, 2 (1994), 140–158.

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

[33] Pedroni, S., and Rappin, N. *Jython Essentials*. O'Reilly and Associates, 2002.

[34] 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.

[35] Resnick, M., Bruckman, A., and Martin, F. Pianos not stereos: Creating computational construction kits. *Interactions 3*, 5 (1996), 41–50.

[36] 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

[37] Soloway, E. How the nintendo generation learns. *Communications of the ACM 34*, 9 (1991), 23–28. use of multimedia technology in education, mediatext.

[38] 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.

[39] Techlinks, A. Reports of it worker shortages continue. *ACM Technlinks for Week of October 14* (2001).

[40] 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.

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