

Introduction

The purpose of this proposal is to create an intervention for an introductory computer science (CS1) course with the goals of increasing the attraction of students to computer science (CS) majors and improving retention of students in first-year programming. The proposed project addresses the Creating Learning Materials and Teaching strategies section of the CCLI Level 1 grant program. We propose to develop a first-year programming course curriculum and instructional materials that use (1) Alice, which teaches introductory computing concepts in the context of animation and movie-making combined with (2) Java, the most commonly used language of instruction in introductory computer science. This blend of Alice with Java will take advantage of the interest and motivation students find in video games and animated films. We believe that this approach will appeal to a wide range of students while covering all the same skills and concepts mandated by curriculum standards [3]. The primary goal is to provide a high impact, motivating context for CS1 at four-year and community colleges. Through our evaluation efforts, we will study the effectiveness of this approach for attracting and retaining majors and minors in CS.

The Alice approach has been highly successful (see the Results of Prior NSF Support section for more details) in pre-CS1 courses. We have been tracking the adoption of Alice in colleges and high schools. To our knowledge, over 100 colleges and more than 100 high schools are using Alice in their curricula. We have an Alice mailing list with 400+ educators. This community of Alice educators is a valuable resource for providing feedback regarding courses and curricula materials needed to support teachers and students.

One message we are hearing (loud and clear) from many teachers at liberal arts and at community colleges is a demand for curriculum and instructional materials that can be used to blend the Alice approach with Java in a regular CS1 course. A primary reason for this demand is the difficulty of adding a pre-CS1 course to their curricula. The liberal arts colleges face the problem of number of courses that can be offered. The number of courses that can be offered is limited by two factors: (1) a small number of faculty members available for teaching courses, and (2) the need to balance the number of courses required for the major with a heavy load of required general education courses. For example, at Saint Joseph's University (a liberal arts school), students are required to take between 20 and 22 general education courses. The computer science major consists of 14 courses (including four mathematics and two physics courses), and the university has a requirement that students have 6 free electives out of their 40 required courses. Thus it is not easy to add a required pre-CS1 course into the existing curriculum. This situation is even more difficult in any liberal arts colleges that have moved to a three-course introductory computing sequence, thus already increasing by one the number of required computing classes majors must take. Community colleges face similar problems, exacerbated by the factors of a 2-year curriculum and the need to adhere to articulation agreements. Thus it would be very beneficial if instructional materials were made available for incorporating Alice into the CS1 course.

Motivation: A primary motivation for this project is the decrease in the number of CS students. According to the Higher Education Research Institute at the University of California at Los Angeles (May 2005), "the percentage of incoming undergraduates indicating that they would major in CS declined by more than 60 percent between the fall of 2000 and 2004, and is now

lower than ... in the early 1980s" [31]. Recent data provided by the NSF CISE division further indicates a significant drop in the number of high-school students that complete advanced placement (AP) tests in CS [7]. While the number of AP tests taken overall has increased by 33%, there has been a drop of 20% in the number of AP tests completed in CS. More interestingly, the number of AP tests taken has gone up in every field except CS. As this data suggests, CS is no longer viewed as a "hot" career. Attracting today's students to computing has become an issue for CS departments across the nation and the situation shows little signs of improvement.

This precipitous drop in interest in CS stands in contrast with job demand data from the US Bureau of Labor Statistics, which predicts that, in the next ten years and after accounting for job losses due to outsourcing, there will be 1.2 million new jobs created in the field of information technology (IT). This is much greater than the number of computing-related degrees granted in the US. Thus, there is a crisis looming on the horizon. Under the current conditions, the US will be unable to satisfy the IT industry's labor demands.

This mismatch between interest and demand impacts the entire US population, not just computer scientists. In his bestselling book *The World is Flat* [15], Thomas Friedman argues for the need for "Versatilists." He points out that preparing students for careers in technology is not enough to compete with nations such as India and China. The greatest need in the global economy is for people who blend their understanding of technology with their knowledge of other domains, such as business or science. The United States is well poised to retain economic leadership by emphasizing an interdisciplinary focus and including technology as part of the mix. This, however, is difficult to achieve if students continue to display little interest in CS.

A variety of complex factors contribute to students' lack of interest and participation in CS. Many students are concerned with the CS culture. Some see computing as an asocial activity [26] that is best suited to those men drawn to computers from an early age [20]. Many female students in undergraduate CS programs reported feeling socially isolated from and less capable than their male peers, particularly if they did not have extensive computing experience prior to college [20]. In addition, many women and underrepresented minority students view the practice of CS as tedious, boring, and irrelevant [2, 20] with little room for creativity [26]. Beginning students have difficulty seeing the real-world relevance of topics such as byte representations and algorithmic efficiency [20]. Faced with a difficult curriculum, an unwelcoming culture, and course lectures and assignments that seem irrelevant to real-world problems, many women and underrepresented minorities choose not to pursue CS. We believe that to *get students in the door*, we need to make the introductory CS curriculum more motivating and relevant. And in making CS more motivating, men, as well as women will become more attracted to the discipline.

A second, but equally important, motivation for this project is the problem of retaining the students in CS majors (or minors). On average, at least half of college students majoring in CS withdraw from the major [11], and the majority of these students withdraw during the first year [30]. Thus, there is a small window in which changes can make the greatest impact. Recent studies (such as [10, 18]) have shown that students who enter college without prior programming experience are at a disadvantage in successfully completing a CS degree. Two issues appear to be critical in determining success in the first year. First, many students enter college with a weak foundation in problem solving and logical reasoning [5, 23, 29]. Second, first-year programming courses rapidly introduce a broad range of programming concepts, and this overwhelms many students. And, we believe this second issue has become more of a problem by the adoption of object-oriented programming as the first programming paradigm. Third, there is the lack of

curricular materials that address the previously described challenges. In an age where the demand for IT professionals is rising [14] and the need to think algorithmically [22] in many career occupations is critical, we need to increase the motivation of students to take a first course in programming and to consider combining a minor in CS with their major, in an interdisciplinary manner. CS departments are not currently successful at reaching the wide range of students who are taking introductory CS. The evidence for this statement includes international studies of programming performance [17], declining retention rates [13], and failure rates sometimes as high as 50% [26]. If students cannot successfully complete the first programming course, there is no chance for them to become or remain CS majors or minors.

Context: CS courses draw examples and assignments from a *context* or problem domain [16, 17]. The choice of context can influence both students' motivation [32] and the quality of their learning as it relates to transfer to other domains [12, 19]. A context that students relate to and find relevant can lead to deeper learning, and material that is learned deeply is more likely to transfer to new contexts [4]. Currently, the typical contexts for introductory computing include business applications (e.g., a bank account example for simple classes in object-oriented programming), systems building (e.g., writing functions to format numbers appropriately into strings), and purely abstract problems. Based on studies of female students, we know that many students are not understanding what problems introductory computing is helping them to solve [2, 20], and students often do not see relevance in their CS classes until their sophomore and junior years.

We believe that changing the *context* of introductory courses without changing the skills or concepts introduced will have a dramatic impact on student motivation. We propose to use the context of animation, sound, and gaming. Today's students have grown up watching animated movies, such as *Toy Story* and *Shrek*, and playing video games. The interest these students had in these activities as young children helps to motivate them to want to understand the concepts that underlie their creation. Using Alice and Java, we believe that introductory CS can be presented through animation, sound, and gaming, while maintaining all of the same skills and concepts mandated by curriculum standards [3]. Rather than manipulating strings and numbers in an abstract setting, students can learn iteration, conditionals, objects and classes, and methods and parameters by manipulating data types common in the contexts of digital storytelling, video game production, and movie-making: actors, scenes, sound, time, and score.

The PIs of this proposal have been successful at introducing motivating contexts. Dann and Cooper's project investigated the use of 3-D animation as a program visualization tool (Alice). Slater is an experienced Java programmer who has worked with the AP College Board to develop and evaluate testing tools. In addition, Slater has experience with training teachers in CS faculty development workshops. We propose to create a textbook and supportive instructional materials to provide a combined Alice += Java approach, oriented around the creation of animated movies, sound, and games. The focus of this approach will be on creating a steppingstone from "computing as facilitated movie-making" to "computing as traditional, textual programming." In this manner, students will learn both the algorithmic process of programming and the syntax of a text-based object-oriented programming language (Java). We propose a pilot testing of these materials at three schools (Carnegie Mellon University, Ithaca College, and Saint Joseph's University). Results of our previous work lead us to hypothesize that this combined context is one that will attract and motivate a diverse student population. We propose to evaluate the effectiveness of this approach as a means of creating a context that

attracts prospective students to “peek through the CS door” and encourage them to stay in the room.

Alice: *Alice* is a programming environment designed to enable novice programmers to create 3-D virtual worlds, including animations and games [25, 27]. In *Alice*, 3-D models of objects (e.g., people, animals and vehicles) populate a virtual world and students use a drag and drop editor to manipulate the movement and activities of these objects. *Alice* was designed through an iterative process of studying how novices try to describe the motions of objects in a 3-D world, and then modifying *Alice* so that the novices’ expectations would be met [28]. *Alice* makes use of program visualization to allow students to immediately see how their animation programs run, enabling students to easily understand the relationship between the programming statements and constructs and the behavior of their animations. In *Alice*, students learn the basics of computing, but where the objects of concern are actors and scenes in a virtual world. *Alice* programs have a strong object-oriented flavor, allowing students to control the appearance and motion of objects, have objects respond to mouse and keyboard input, or do any sort of computation that would normally be done in an introductory programming class. Students learn about objects and aggregation by addressing the component features within objects—lifting an arm, turning a head. Students learn about sequencing and iteration by constructing a series of actions in a scene. Further details about *Alice* may be found in Appendix A.

Alice [24] has been used successfully in undergraduate classes that draw a diverse range of students into computing [6, 8, 9]. A set of curricular materials and a text [9] were developed and piloted tested in introductory computer programming courses (pre-CS1) offered at Saint Joseph’s University (SJU) and Ithaca College (IC). At SJU, the materials were taught prior to CS1, while at IC the materials were taught concurrently with CS1. Successful results in teaching *Alice* concurrently with CS1 lends credence to our planned *Alice* and Java integration. Details of the results of this study are provided below in the Results of Prior NSF Support section.

Synergism, Alice += Java: The *Alice* approach has strong motivation and appeal for students who have experience with video games and multimedia in their daily lives. *Alice* uses 3-D animation and program visualization to introduce fundamental programming concepts such as sequencing, decision making, and repetition, as well as an intuitive understanding of object-oriented concepts such as classes (including behavior and state), objects, methods and inheritance. Java is, of course, a language used to create real-world applications that can incorporate graphics, sound, and gaming computations. This project will create a synergetic intervention that uses *Alice* to introduce computing in an animated movie-making context and then uses the same concepts in Java programs to produce sound and games.

We believe in “one hard problem at a time.” *Alice* allows students to assemble programs using a drag-and-drop editor. An advantage of the drag-and-drop editor is that students can focus their attention on understanding programming constructs without the initial distraction and frustration of syntax details. Once the programming concept is understood, then Java can be used to learn any syntax necessary to create a text-based program that is contextually related to the animation previously created in *Alice*. For example, *Alice* can be used to create an animation in which a princess has been grounded by the evil Queen. The princess uses her cell phone to call a dragon-taxi service. The dragon flies to the castle and the princess escapes from the castle. In creating this first animation, students learn concepts of program design, statements, sequential execution, classes, objects, methods, and method calls. Java can then be used to apply these

concepts (and learn the syntax) in writing a program that creates a sound file for the cell-phone ringing. A new version of Alice (Alice 3.0) is currently under development. The proposed new textbook and instructional materials will use the latest version of Alice.

Results of Prior NSF Support

The work in this proposal builds on the success of our prior innovative research efforts in attracting and retaining students using Alice as a program visualization tool, as well as from our experiences with assessment.

Alice (NSF-0126833, 0302542, 0339734): Alice has been used successfully as an intervention to draw at-risk students (who are disproportionately female or underrepresented minorities) into computing [6, 8, 9]. At-risk students were defined as those students who had demonstrated less success in math and/or those who had little previous programming experience. A textbook [9] was developed and pilot tested in introductory computer programming courses (pre-CS1) offered at SJU and IC. Additionally, a detailed set of curricular materials [33], including several different curricular models (with complete lecture notes), laboratory exercises, solutions, exams, assignments, sample student projects, and other material, was created. At SJU, the materials were taught prior to CS1, while at IC the materials were taught concurrently with CS1. The primary results of this investigation were:

- 1) The average grade for at-risk students exposed to Alice was a 3.0 GPA in CS1, which is comparable to the grades of students who were at no risk or low risk. The average grade for at-risk students not exposed to Alice was a 1.2 GPA in CS1.
- 2) 88% of at-risk students exposed to Alice enrolled in CS2 after CS1. Only 47% who were not exposed to Alice enrolled in CS2. ($p < .05$, chi-squared)

Further details concerning this investigation are available in [21].

Based on the results of our proof-of-concept study, we were awarded two follow-on grants. The first grant involved the use of Alice with three diverse community colleges to attract and retain majors, and to improve the computer literacy course. The second grant involved the running of regional summer workshops to train faculty on using and teaching with Alice. In summer, 2005, workshop participants included 95 faculty members from 57 schools attended, and more than half taught with Alice during the 2005-2006 academic year. Preliminary results for both of these studies suggest that using Alice has had a statistically significant positive impact on students' understanding of basic programming concepts in both the community college and four-year college setting. A list of the schools using Alice is available in the Appendix. Instructional materials developed as part of these three grants are updated and maintained by Cooper and Dann and have been made freely available to instructors at [33].

Assessment (NSF-0511940, 0512098): Dann and Cooper are PIs in a collaborative research project with Barb Moskal (Colorado School of Mines) and Mark Guzdial (Georgia Institute of Technology). This grant has been to design and validate two assessment instruments. The first is a modern attitudes instrument, focusing on confidence, beliefs in one's ability to succeed in computing, perception of computing as a male field, usefulness, and interest. The need for a standard attitudes instrument in computing is especially significant, as the most recently validated prior instrument dates from the early 1980's, prior to the time when personal computers were widely available, and most computing was done on mainframes. The second

instrument is a concepts instrument for CS1, measuring fundamental programming concepts, algorithmic thinking in programming, and problem solving through programming. The particular need for this instrument is to have a standard way to measure the effectiveness of various interventions (which are most commonly proposed at the introductory computing level). We expect to be able to use these instruments as part of our evaluation plan, as described below.

Goals and Objectives

The purpose of this project is to create a textbook and develop curriculum materials for teaching and learning fundamental programming concepts using simulation and visualization as part of an introductory computing course combining Alice and Java. This will be a proof of concept project, tested at a few schools. The approach used in this proof of concept will take advantage of a high-level of interest in graphics, animation and storytelling, commonly found among students who have grown up in a multimedia world. However, the major emphasis is the use of visualization to teach and learn a strong core of fundamental programming concepts and problem-solving techniques in an OO, interactive environment.

With NSF support, we have previously created highly successful introductory Alice instructional materials, including a textbook. These materials, however, were designed for and tested with pre-CS1 (and pre-AP) courses. Over the last two years, we have conducted several Alice-community “brainstorming sessions.” The feedback obtained in these sessions has provided impetus for developing instructional materials that can be used in CS1. The need is for materials that will assist the teacher and the student in successfully mediating a transfer of concepts from experience with building animations in Alice to writing text-based programs in Java. The main idea is to take advantage of Alice in developing intuitive understanding of both object-oriented and fundamental programming concepts and mediate a transfer of these concepts to build skills with constructing Java programs in a traditional text editor or IDE tool.

The goals of this project are to:

1. Produce a complete draft of an Alice += Java textbook, up to date with Alice 3.0 .
2. Provide a high impact, motivating context for introductory computer science.
3. Apply and integrate the text and curricular materials at (at least) two institutions (Ithaca College and Carnegie Mellon University).
4. Assess the effectiveness of goals 2 and 3.

We propose to create a textbook and supplementary instructional material that integrates Alice 3.0 with Java. Although a pre-CS1 Alice text and several CS1 Java texts each separately exist, we believe that combining the two into one textbook is needed to assist the instructor and student in transitioning between Alice and Java concepts and techniques. We also propose to provide online access to the supportive instructional materials. Use of these materials is expected to be highly motivating to students and result in an increase of students participating in computer-related degrees at the college level. It is further expected to improve the retention rate of first-year computing majors.

Measurable Outcomes:

- i. Completion of textbook and instructional materials that combine Alice 3.0 and Java.
- ii. Creation of online support for instructional materials.

- iii. The majority of participating students will evaluate the courses that use these materials positively as measured by a closed response survey.
- iv. Improvement in retention rates of first-year CS students at the participating schools.
- v. Courses that utilize the above described materials will have a greater retention rate in terms of declared CS majors and minors than comparable courses at the same level.

Detailed Project Plan

To have the maximum impact on the CS pipeline, we believe that it is important to generate student interest in CS early and to support the development of their interest through the first year of their study. We will focus on developing instructional materials and on interventions that attract students into introductory classes, keeping these students involved in CS, and supporting faculty teaching these classes. Details of the project plan are described here in terms of (1) materials development, (2) target groups and (3) focal points, based on our goals and objectives.

Materials development: We plan to create a text that combines Alice and Java. Key to the success of this innovative approach is the interweaving of fundamental, object-oriented programming concepts (creating animated movies using Alice) and learning the details of syntax and textual programming skills a real-world language (Java). Based on our previous work, we believe that such an interweaving can be accomplished. A potential topic outline is shown in Figure 1. We expect to cover each topic by illustrating examples in Alice followed by examples in Java. For example, when introducing methods and parameters, we will program the story of a girl and her dad working out, in which the girl teases her dad about being out of shape. Methods will be invoked on the girl and the father, as she learns her father is quite as out of shape as she had thought. The coding in Alice 3.0 and the coding in Java will be integral and additional code will be seen in Java examples. Please see the supplementary materials for a letter of support from Prentice Hall, publisher of the previous Alice [9] text, and likely publisher of the proposed combined Alice and Java textbook.

In addition to a textbook, we plan to develop instructional support materials. This includes lecture notes (presentation slides), sample quizzes and exams, sample labs, and sample projects.

Introduction

A perspective on programming languages leading up to Alice and Java

Part I: An Introduction to Object-Oriented Programming

A. Objects and Classes

1) Design

- a) Problem solving techniques
- b) Storyboards
- c) Stepwise refinement
- d) Pseudo-code

2) What is an object?

- a) Using methods and parameters
- b) Accessing instance variables

3) What is a class?

- a) Encapsulation (private and protected variables)

- b) Methods
- B. Creating New Classes – three implementation strategies
 - 1) Modifying existing class (making and modifying a copy – Alice-only)
 - 2) Interfaces (Java-only)
 - 3) Inheriting from existing class
 - a) Adding instance variables
 - b) Constructors (Java-only)
 - c) Adding methods
 - d) Polymorphism (Java-only)
 - 4) Creating classes from scratch
 - 5) Static classes and methods
 - a) Alice world-level methods
 - b) Methods called from main – as a driver
 - c) Static classes in Java

Part II: Adding Capabilities

- A. Control
 - 1) Input/Output
 - 2) Data types
 - 3) Decisions & expressions
- B. Functions (methods non-null return type)
- C. Events and GUIs
- D. Exceptions (Java-only)

Part III: More Complex Programming

- A. Repetition
 - 1) For loops
 - 2) While loops
 - 3) Recursion
- B. Concurrency (Do Together in Alice; Java threads may be deferred to the Appendix)
- C. Design patterns (Iterators, Visitors, MVC, and others – Java-only)

Part IV: Data Structures

- A. 1D Arrays
- B. 2D Arrays (Java-only)
- C. Lists
- D. Sets (Java-only)

Appendix

- A. Using the Alice interface
- B. Using the Java Eclipse IDE
- C. Primitive data types
- D. Applets (if not covered in text)
- E. Sound (information beyond the text narrative)

Figure 1. Potential topic outline for integrated Alice and Java text

Target groups: We plan to offer Alice/Java classes for two levels of students: four-year and community college.

Four-year college faculty and students: Two of our institutions (CMU and IC) will pilot at least one version of the new course. Instructors and students in these pilot courses will provide formative feedback for revision purposes, as is described in the evaluation plan of this proposal. We will impact faculty and students through our summer teacher training workshops. Students in our pilot courses will be selected to act as student assistants for the summer workshops.

Community college faculty and students: The first-year results from our current NSF ATE-supported project indicate that using Alice prior to CS1 helps to improve student performance and increase retention in CS courses at the community college level. However, community college faculty members are often concerned about transitioning students from Alice to a more traditional text-based object-oriented language (e.g. Java, C++, or VB). Accordingly, several community colleges have expressed interest in participating in this project by using the Alice/Java class developed in this proposal. We expect at least one community college will participate formatively in this study.

Attracting Students: *Attracting prospective students into an Alice/Java course by enabling them to “peek” into the introductory CS curriculum.* By integrating Alice’s animated movie-making context with traditional Java programming, we hope to make CS more attractive to students, particularly women and underrepresented minorities. However, unless we provide opportunities for students to appreciate what is going on inside the CS classroom, we are unlikely to realize the full benefits of the approach. We propose to provide opportunities for prospective students to “peek” into the world of the CS by holding an “animation fair” on-campus to premier student work.

One advantage of introducing programming through creating animated movies is that the artifacts students create can be readily understood and appreciated by prospective students. We have noticed that students completing an introductory programming class based on the Alice system often invest many hours perfecting their Alice projects and frequently show their completed Alice projects to friends. The practice of students showing their work to friends is, in essence, an informal recruiting program for the class. Further, the fact that students with no programming experience can readily appreciate programming artifacts of other students creates the opportunity to use these artifacts (with written permission from the student creators) as tools for recruiting new students.

To attract students to our course, we will work with the teachers who participate in the project to create local animation fair events. An animation fair is similar to an “opening night” of a movie, except that instead of a movie, students will be viewing the virtual worlds created by their peers. Students will have the opportunity to see what their peers are doing, and thus, what they could be doing. Only programs for which written permission from the student developer was acquired will be viewed at these events.

Project management: The project co-PIs will act as a team to manage the three branches of this project: 1) development of a text and instructional materials 2) providing support for animation fairs, and 3) project evaluation. To coordinate these efforts, the team will schedule monthly teleconferences and face to face meetings twice a year.

In this collaborative project, Cooper, Dann, and Slater will be the co-PIs for the project, with Cooper taking on the role of managing the progress of all activities to satisfactorily complete the project. Development of the text and creation of the instructional materials will be completed by Dann, Cooper, and Slater. Cooper will manage the website for hosting instructional materials. Slater will serve as overall Java expert. Dann and Cooper, experienced in assessment techniques, will manage the evaluation of the project.

Timeline:

Summer 2008	<ol style="list-style-type: none"> 1) Complete initial draft of Alice + Java text 2) Create initial online repositories of curricular materials 3) Evaluation: Obtain baseline data from participating schools
Fall 2008	<ol style="list-style-type: none"> 1) Pilot round of course offerings 2) Propose workshop at SIGCSE 3) Presentation and workshop at CIT (national community college conference) 4) Formative Evaluation: Data collection in pilot courses
Winter 2008	<ol style="list-style-type: none"> 1) Revisions to text based on formative feedback from Fall, 2007, usage 2) Revisions to instructional materials based on formative feedback from Fall, 2007, usage 3) Evaluation: Obtain peer reviews of text and instructional materials
Spring 2009	<ol style="list-style-type: none"> 1) Second round of course offerings at participating schools 2) If accepted, run workshop at SIGCSE 3) Limited pilot with at least one community college and 1 or 2 other schools 4) Evaluation: Data collection in participant courses
Summer 2009	<ol style="list-style-type: none"> 1) Final modifications to text and instructional materials based on Spring, 2008, feedback 2) Evaluation: Additional data collection and analysis
Fall 2009	<ol style="list-style-type: none"> 1) Evaluation: Summative data collection and analysis 2) Submit text to publisher 3) Widespread dissemination of the curricular materials via the Alice e-newsletter, the www.alice.org website, the SIGCSE listserv, and the AP CS listserv

Evaluation Plan

As is suggested by the timeline above, assessment and evaluation activities will be ongoing throughout the proposed project. Attention will be given to both the formative and summative aspects of evaluation. The formative assessment plan described in the section that follows is designed to acquire information that may be used to improve project-related activities as they are being implemented. In other words, this project uses the cyclical model of assessment that was described in the RFP. The summative portion of this assessment plan is designed to measure the attainment of the project outcomes. These outcomes were discussed early and were constructed based on the project goals and objectives.

All appropriate human subject procedures will be followed throughout the investigation. This includes receiving approval from internal Human Subject Review Boards at all institutions, obtaining all necessary written consents, and the maintenance of strict confidentiality throughout the study.

We do not intend to use an external evaluator for this project. An external evaluator adds an increased sense of independence to the project. However, the scope of this project is to provide for proof of concept and thereby has funding limits. In addition, Cooper and Dann have experience in working with an external evaluator for similar types of projects, in which we have developed necessary skills for proof of concept formative and summative evaluation. This leads us to believe that an external evaluator is not necessary for this project. If this project yields positive results, we will apply for a full CCLI Level II grant to more broadly test the materials under the supervision of an external evaluator.

Formative: Qualitative research techniques will be used for formative evaluation purposes. Qualitative techniques have the advantage of providing detailed descriptive information and this type of information is useful for project improvement purposes.

Peer Review of Curriculum Materials. This project will result in the development of a textbook and instructional materials. At least three external experts in the fields of computer science and computer science education will review all materials and revisions will be made based on their recommendations. One of these experts is expected to be Randy Pausch, developer of the Alice tool. Peer review is a widely accepted technique for examining the content, construct and criterion validity of instructional materials [1].

Student Interviews. The instructional materials will be pilot-tested starting in the fall of 2008. At the end of the semester, five randomly selected students in the pilot classroom will be interviewed. This interview will be conducted by a co-PI who is not teaching at the school. The interview will focus upon their experiences with the instructional materials and how these experiences influenced their future educational goals and decisions. The individual interviews will allow the investigators to acquire a detailed understanding of students' experiences with the instructional materials.

Student Focus Groups. Students not selected for individual interviews will participate in student focus group activities. Questions asked in the focus groups will be the same as those asked during the individual interviews. The focus group activity will ensure every student has an opportunity to express his/her viewpoint.

Summative: The summative evaluation will primarily use quantitative research techniques to support the statistical analysis of our efforts. At each location, pilot and control classrooms will

be selected, if possible. The pilot class will use the materials developed here as part of their first programming course; the control classes will participate in a traditional first-year programming course. If a control classroom is not available, historical data will be collected before the intervention is implemented for comparison purposes. In pilot and control classrooms, a survey will be given to determine student background in mathematics, computers, and CS. The data collected by the readiness survey will be used to examine relationships between mathematics and CS preparedness, student performances and rates of attrition.

Attitudes Survey. At the beginning and end of the semester, students in both the pilot and control classrooms will also complete a computer science attitude survey. This survey will help to determine whether students' participation in their first computer science course has further stimulated their interest in pursuing a degree in computer science. We are planning to use the newly validated attitudes survey, as described in the Prior NSF Support section above.

Statistical comparisons will be made between the pilot and the control classroom, providing evidence as to the influence of the newly designed course on students' computer science interests. Special attention will be given to the relationship of the students' entering level of mathematics and computer science readiness and their continued interest in a computer science degree. Comparisons will also be made between female and minority students that are in the pilot and control classrooms. We do not expect these numbers to be large enough for statistical inference. But we will use descriptive statistics as a method to compare these subgroups. Though the information acquired from these comparisons cannot be generalized beyond the current population, it will be used in the development of a full proposal.

Pretest/Posttest: We intend to use a newly validated concepts survey (as described in the Prior NSF Support section above) as a pretest/posttest assessment instrument. Again, attention will be given to students' entering readiness and their performance on the pre and post instrument, and comparisons will be made between female and minority students in the pilot and control classrooms. The gender and ethnicity comparisons will be descriptive in nature and can only be used to make speculations for the development of the full proposal.

Retention Statistics: Statistical comparisons will also be made between the number of students in the pilot and control classrooms that select to continue to pursue a computer science degree. This data will suggest whether the newly developed course contributes to student retention in computer science. Readiness will be examined to determine whether it was a factor in outcomes of these statistics. Descriptive comparisons between female and minority students in the pilot and control classrooms will also be made. As previously stated, these comparisons can only be used to make speculations that will guide the development of the full proposal.

Capabilities of the PIs

We believe that we are uniquely qualified to carry out this project. Cooper has been teaching introductory CS for 12 years. He also has 10 years of industry experience and is familiar with what industry requires from CS graduates. Dann has 16 years experience teaching introductory CS. Her expertise spans applied research in program visualization, curriculum development, and workshop presentations. Dann has served as a member of the Visualization Working Group, studying the effectiveness of visualization in CS education. Slater has extensive experience in teaching with Java and has served as an AP trainer and grader for many years.

Drs. Cooper and Dann have been working with Alice for 8 years. Slater is in residence at Carnegie Mellon and has been working with Alice for 2 years. Their experiences with using and teaching computing concepts with Alice enable us to complete this project. We are working synergistically with the developers of Alice at CMU. Cooper and Dann make biannual visits to meet with the development team to discuss the design of Alice and how to better tailor it to the needs of novice programmers. In the past, this team has always been responsive to implementing our requests and suggestions. We have written a book published by Prentice Hall [9] concerning the use of Alice in the CS classroom. Our Alice work is receiving strong support from the CS community. Several of our papers have been accepted for conferences. (See the PIs biographical sketches for a partial list.) As part of our current CCLI grant, we have offered workshops to assist faculty in learning to teach with Alice, and have had more than 200 faculty requests to participate. We have demonstrated on our college campuses that teaching Alice really does make a difference for our at-risk CS majors.

We have significant project management experience with NSF grants. Dr. Cooper is PI and Dr. Dann co-PI of a million dollar ATE grant (concluding next summer) involving 7 schools, approximately 80 faculty, and more than 3000 students. While we do not yet have final results, our initial results, in terms of student attitudes, retention within, and attraction to the major, are promising.

Our experience in working with female and minority students: Dr. Cooper has been running Pathways to Careers in Mathematics and Computer Science (PACMACS) for the past 6 years. PACMACS enables 10 African-American students from (non-magnet) Philadelphia public high schools per semester to take a programming class with Alice (as well as a mathematics class) at SJU, as well as receiving mentoring, and other preparations for college. PACMACS has been quite successful, with many of its graduates either majoring in, or graduated with college degrees in CS and/or engineering. Dr. Dann has been working with the CS and Hawaiian Studies faculty at the University of Hawaii to create a combined Alice and Hawaiian Studies course for native Hawaiian college students. In this project, Alice is being used to introduce fundamental and object-oriented programming concepts to native Hawaiian students by creating animations that tell the stories of Hawaiian folklore.

Our experience in working with community college faculty: Drs. Cooper and Dann are just completing a successful three-year grant to work with the Community College of Philadelphia (an urban HBCU), Camden County College (a suburban campus), and Tompkins Cortland Community College (a rural school). We have learned the importance of providing detailed course and supporting materials, as community college faculty typically have much higher teaching loads than four-year school faculty.

Dissemination

Over the course of this grant, we expect to generate knowledge in two main areas: 1) research results concerning students' decisions to continue with or leave CS and computing-related fields, and 2) practical experience in how to use an integrated Alice/Java approach to increase enrollments of students in first-year CS courses.

Reaching the Larger CS Education Community: We plan to present our research findings at professional conferences to help inform the design of CS courses that address the needs of a wide range of students. In addition to submitting journal articles, we expect to present the results of our work at SIGCSE, ITiCSE, FIE, CHI, HCC, and CIT. To reach community college faculty, we plan to give talks and run workshops at the Conference for Information Technology and Innovations conferences. Additionally, we expect to present at several Prentice Hall Information Technology (PHIT) conferences.

To enable instructors at other academic institutions to begin using the Alice/Java approach, we plan to produce a textbook and create an online community to provide support and materials for educators teaching Alice/Java courses. We will also send announcements about the availability of the new curricular materials to the SIGCSE members list, the AP CS listserv, and to our 750+ faculty Alice community.

Future Plans

As we continue to pursue Alice and Java, we plan to begin work that focuses on Hispanic students and Pacific Islanders. Because the focus of this work is on teaching programming, rather than teaching the English language, we plan to produce versions of our systems and materials in Spanish and other relevant languages. Other possible groups that may well benefit from our Alice and Java approach include Native Americans who have a strong storytelling culture and the deaf community where the ASL is used as a visual means of communication. Alice and Java will allow this group to benefit from the strong visual nature of our approach.

Summary

The proposal describes a project to merge a successful curricular approach (Alice) to teaching pre-CS1, introductory programming at the college level into a CS1 course in Java. Alice is a software environment that supports 3-D programming using a drag-and-drop editor. Through the use of Alice, students can learn the basics of algorithmic programming without the burden of syntax errors. The primary motivation for this project is based on feedback and requests from faculty at liberal arts and community college schools where the ability to add a pre-CS1 course to the curriculum is limited by 1) a small number of faculty members available for teaching courses, and 2) the need to balance the number of courses required for the major with a heavy load of required general education courses. The need is for materials that will assist the teacher and the student in successfully mediating a transfer of concepts from experience with building animations in Alice to writing text-based programs in Java. The main idea is to take advantage of Alice in developing intuitive understanding of both object-oriented and fundamental programming concepts and mediate a transfer of these concepts to build skills with constructing Java programs in a traditional text editor or IDE tool.

The measurable outcomes of this project will be 1) completion of a textbook and instructional material that combine Alice 3.0 and Java, 2) creation of online support for instructional materials, 3) evaluation of the effect of using this approach in CS1-type courses, and 4) improvement in retention rates of first-year CS students at the participating schools.

We have a proven track record with our pre-CS1 *Learning to Program with Alice* text and supplementary instructional materials, developed under previous NSF support. We will build on these materials and teaching strategies to forge an innovative approach that uses the

excitement of Alice to generate student excitement with learning to program in a real world language and to pursue a major or minor in computer science and other computer-related career paths.