Using Alice in CS1 – A Quantitative Experiment

Ryan Garlick
University of North Texas
Dept. of Computer Science and Engineering
garlick@unt.edu

Ebru Celikel Cankaya
University of North Texas
Dept. of Computer Science and Engineering
ecelikel@cse.unt.edu

ABSTRACT

We present the results of a 2-semester study of using the 3-D graphical programming environment Alice to introduce programming fundamentals during the first two weeks of CS1. One cohort of students was taught basic programming constructs via traditional pseudocode, while a second group used Alice. A student survey was collected, along with performance metrics on a common quiz and first exam.

Students using Alice scored lower than those taught with pseudocode on common performance metrics and responded less-favorably to Alice in a survey. Anecdotal evidence of using Alice with younger students was more positive.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: Computer Science Education.

General Terms
Measurement, Experimentation.

Keywords
Alice, CS0, CS1, pedagogy, retention

1. INTRODUCTION

Declines in Computer Science enrollment are prevalent and well documented [3]. Efforts to increase performance in future classes and aid in retention and recruitment have encompassed a broad spectrum of strategies. The educational community has tried revamped material, objects first [2], changing the language used [5], using games [6], requiring a CS0 class [4], and many other strategies.

While proposed solutions have been numerous, empirical studies of what works and what does not are less common. Other studies have compared CS1 students who took a CS0 course to those who did not [7, 9]. Drawing conclusions about the influence of CS0 content seems difficult, as exposure to any programming related material in an additional course is likely to improve CS1 performance.

This paper aims to measure the effect of using a brief introduction to programming fundamentals via Alice versus a control group using traditional pseudocode (hereafter the pseudocode cohort). The comparison period involved the first two weeks of each CS1 class.

Three CS1 instructors participated in this study (including the authors), which encompassed 5 class sections, 2 semesters, and over 150 students.

From the Alice website [1]: [Alice] allows students to learn fundamental programming concepts in the context of creating animated movies and simple video games. In Alice, 3-D objects (e.g., people or vehicles) populate a virtual world and students create a program to animate the objects.

In Alice’s interactive interface, students drag and drop graphic tiles to create a program, where the instructions correspond to standard statements in a language such as Java and C++. Alice allows students to immediately see how their animation programs run, enabling understanding of the relationship between the programming statements and the behavior of objects in their animation. By manipulating the objects in their virtual world, students gain experience with all the programming constructs typically taught in an introductory course.

UNT conducts summer technology camps for students in grades 8-12 and has had strongly positive anecdotal evidence from using Alice, as similarly noted in [8].

One author of this paper witnessed Alice captivate an 11-year old female in a matter of hours, prompting her to spend days creating a full game.
2. PREVIOUS RESEARCH

Studies have found benefit to using Alice in the classroom [9], positive results with some caveats [10], and reservations [11].

A study at Tufts University [11] used Alice for the first half of a semester followed by a higher-level language (HLL). In a subsequent semester, Alice was interleaved with C++, with each concept introduced in Alice and transitioned to C++. Through anecdotal evidence of student behaviors and attitudes, Powers noted a difficulty for students in transitioning from Alice to a higher level language. The authors noted that several students had difficulty with the precision required by the higher-level language, and became discouraged when their programs did not compile (something made impossible by the Alice interface). Some students voiced the opinion that the Alice environment seemed to be designed for a younger audience.

A recent study by Mullins, Whitfield, and Conlon compared sections of CS1 using Alice to sections of CS1 using C++ [10]. Interestingly, statistics were gathered on the performance and retention of these students as they completed CS2 and CS3. In CS1 (a course required for majors, but available for liberal studies credit by non-CS majors), Alice sections saw “a 4% increase in women enrolled, a 4% increase in the number of students that passed, and a 4% decrease in withdrawals.” Among CS majors (CS2 and CS3), those who used Alice in CS1 had similar lower withdrawal rates and increased success rates, although mean grades were lower for students using Alice.

Johnsgard and McDonald [7] compared the CS1 success rate (a grade of C or better) for students who took CS0 and those who did not. The success rate improved for students with the introductory course (which contained an Alice component), however the authors acknowledge that some of the positive results could have been due to the fact that the CS0 cohort “…had an additional semester to mature before beginning Programming 1.”

An often cited study of Alice in a collegiate environment is by Moskal, Lurie, and Cooper [9]. This study (using a relatively small sample size) noted improvements in at-risk student performance, retention, and attitudes toward computer science. However, as with the Johnsgard study [7], the treatment group had an additional CS0 course devoted to Alice, while the control group did not.

Several previous studies have compared students exposed to additional coursework (using Alice) to students only enrolled in a traditional CS1 course. The authors of this paper feel this was a very limiting factor in determining the effectiveness of Alice versus the effectiveness of exposure to any CS related material in CS0 and desired a measurement of the performance of two student cohorts with equal experience, one using Alice, and one using traditional methods of introducing computer science.

Some authors such as Johnsgard and Powers [7] noted that students spent a good deal of time with Alice focusing on creating smooth character animations rather than programming fundamentals. As discussed in later sections, considerations designed to address this issue (as well as the transition from Alice to HLL) will be present in future releases of Alice.

3. EXPERIMENTAL METHODOLOGY

From the outset, the authors pursued simple research questions – do the students enjoy using Alice more than traditional lessons; do they learn as much or more from it; and does the material make the students eager to explore on their own.

Students were given a lesson in either pseudocode or Alice (based on class section), followed by a homework specific to the material. All students were then given a common quiz over programming basics and a common first midterm based on Java. Furthermore, a survey was also administered to all students.

Both assessment techniques, namely the quiz and first midterm, had questions that could be solved by students learning Java with or without the Alice tool. Instructors made every effort to ensure no bias was introduced in learning Java with or without Alice.

3.1 Alice Class Sections

Two CS1 class sections (82 total students) were presented with instruction in Alice. A very brief introduction to programming was presented in pseudo-code style statements. However, the core concepts of variable creation, looping, control statements and functions were presented via examples given in Alice, with discussion of how Alice was enabling a graphical representation of logical constructs.

Students worked through the tutorials included with the Alice software and were provided with links to additional Alice resources and tutorials. Two weeks of classroom lecture were
spent covering these topics, introducing Alice, and working through examples in class, followed by a transition to Java. Students were encouraged to explore the interface on their own and create sample animations.

The homework assignment was as follows:

Create an animation in the Alice environment. Your topic may be anything that you choose. Special consideration will be given to interactivity, complexity, and creativity. Your animation must contain the following elements:

- You must create a new variable, function, and method
- You must have interactivity in your animation that is controlled by the user.
- You must have a loop in the program.

Since the public will vote on the best animations, you are encouraged to learn extended techniques to make your program more sophisticated. Prizes will be awarded to the best submissions.

Student turn-ins ranged from the non-functional, to intricate, simulated interactive worlds including a fully functional bowling game.

3.2 Traditional Pseudocode Class Sections

Two separate CS1 class sections (72 total students) were introduced to programming concepts via traditional pseudocode. This involved two weeks of slides and discussion related to pseudocode. The same topics: looping, control statements, and functions were introduced by tracing through pseudocode algorithms rather than via Alice. The pseudocode cohort also transitioned to Java.

The homework assignment in these sections was to create a pseudo code algorithm to calculate GPA.

3.3 Homework, Assignments and Quizzes

A common quiz was given to both the Alice and pseudocode cohorts. It included a given algorithm for finding the average of a group of numbers using a loop and conditional statements presented as blocks with arrows drawn between them to indicate program flow. The assignment was designed to provide a mix of pseudocode and the "tiles" dragged into the Alice environment to form programming constructs. Students were then asked to create a "make change" program, as illustrated by this excerpt from the quiz:

"Create a process for determining the correct number of quarters, dimes, nickels and pennies to give as change. For example, if the change amount is .82 (it will always be .99 or less), your process should end up with quarters = 3, dimes = 0, nickels = 1, pennies = 2".

Students were told they could use any method to solve the problem – drawing flowchart-style blocks similar to the development window in Alice, writing pseudo-code, or any other method that presented a coherent solution to the problem. This flexibility was designed to minimize any possibility of bias to the advantage or disadvantage of either teaching cohort employed in the experiment.

4. OUTCOMES

The “make-change” quizzes were graded on a scale of 50 to 100 in 10 point increments. All quizzes were graded by a single instructor without knowledge of the students’ cohort. All statistics presented in the paper are based solely on the students’ success rate in the given assessments, regardless of considering parameters such as their gender, final course grade, switching to or from other majors, etc.

The mean quiz grade was lower among students in the Alice cohort, however the data did not quite establish statistical significance to a P-value of 0.05 (P=0.0527).

A common first exam was given 2-3 weeks after the Alice / pseudocode instruction. While largely based on Java syntax, the exam included questions that tested the ability to recognize the output of given programs and analyze existing program logic.

All exams were graded by a single instructor without knowledge of the students' cohort. Mean exam grades were significantly higher among students in the traditional cohort (P=.0291).
As mentioned in [10], students also seemed to have difficulty with the transition from Alice to Java, which the authors of Alice are addressing in the forthcoming Alice 3.0. Alice 3.0 allows a smoother transition to Java, with models from the Sims game, allowing students to focus less on creating animations and more on the logic of programming.

The number of respondents to the survey, quiz, and exam vary since not all students were present for the survey or quiz.

### 4.1 Student Surveys

A common survey was given to participants from both groups. Participation in the survey was voluntary, with 70 respondents from traditional sections and 71 from Alice sections. Students rated their answers to the following questions from 1 (low) to 10 (high).

The survey was as follows:

**Please Rate:**

1. Your understanding of variable creation and assignment
2. Your understanding of decisions (control statements)
3. Your understanding of looping
4. Your understanding of function calls and parameter passing
5. Your enjoyment of learning “pseudo-code” as presented (students in the Alice cohort were told that they were learning pseudo-code in a different manner).
6. The effectiveness of teaching programming constructs in this way
7. Your desire to explore on your own to solve more sophisticated problems
8. Your ability to use techniques learned to solve new problems

Mean responses from the Alice cohort were lower than those from the traditional pseudo-code group for every question on the survey.

### 5. CONCLUSIONS

Our results from the use of Alice for a two-week period at the start of CS1 were negative; however the authors view Alice as an important tool. This result could be due to the first impression that students get with Alice. Or, students simply expect a more conventional method for learning Java: a traditional pseudocode approach, which most probably is the technique that they are used to from any earlier experience.

It is not the desire of the authors to cast a negative light on Alice – we still introduce it and allow CS1 students to voluntarily create animations. One instructor had positive results presenting Alice at the end of a CS 1 semester. This gives us a good incentive to continue to introduce students to Alice.

More study on the effectiveness of Alice in a college setting seems warranted. The authors plan on expanding the current study to include more parameters such as students’ gender, transferring to/from other majors, etc. Furthermore, the expanded assessments will be repeated at multiple times during the semester: at the beginning, in the middle, and at the end of the semester to collect and analyze more data.

### 6. ACKNOWLEDGMENTS

This research was funded by a University of North Texas Transformative Instruction Initiative Grant.
REFERENCES