The Impact of Alice on the Attitudes of High School Students Toward Computing

Eileen M. Peluso, PhD
Lycoming College

Gene Sprechini, PhD
Lycoming College

Abstract
In this paper, we present the results of a study conducted to determine the impact of Carnegie Mellon’s Alice on the attitudes of high school students with regards to computing. A two-week unit of instruction in Alice was developed and presented to high school students, and pre and post surveys were administered to assess attitudinal changes. We present the content of the unit and analyses of the collected data.

1. Introduction
Enrollment in undergraduate computer science and engineering programs has suffered from three problems: dropping enrollments of female students since the mid ‘80s (Hill, et al., 2010; Klawe, et al., 2009; Ericson, 2009; Margolis, et al., 2002; De Palma, 2001), a long-standing lack of diversity (Zweben, 2009), and dropping enrollments and retention overall for most of the past decade (Bruckman, et al., 2009; Zweben, 2009; Vesgo, 2006). Research into the reasons behind the first two problems has been sparse but increasing in recent years. There has been much speculation as to the reasons for the third problem but little research. Rather, efforts have focused on ways to (1) engage students in K-12 courses and outreach programs to raise interest in the discipline and (2) modify introductory undergraduate computer science courses to improve enrollment and retention. These efforts may be having the desired effect as the past two Taulbee Surveys indicate modest growth in the number of students enrolled in U.S computer science programs (Zweben, 2010).

Carnegie Mellon’s Alice project (www.alice.org) has produced an interactive, animation-based programming environment that is aimed at addressing all of these problems and it is sufficiently flexible for use from middle school to the first semester undergraduate level. The project reported upon herein involved the development of a two-week curricular unit in Alice and the presentation of that unit in high school classes in north central Pennsylvania. Pre and post surveys were administered to students in an effort to determine if exposure to Alice had an impact on students’ attitudes toward computing. Demographic information was collected as well in an attempt to determine if certain groups of students were influenced to different degrees. The homogeneous nature of the student population prevented us from categorizing responses based on ethnicity, but we were able to classify them by gender and favorite subject area. A second and equally important objective of the project was to advance the use of Alice in regional school districts by introducing the software to middle and high school teachers and assist them in its use.

1.1 Related Work
Results from research to determine what influences a student’s decision to pursue an undergraduate degree in computing support our intuition that there are no simple answers. For example, considerable effort has been put into determining the connection between video game use as a child and subsequent selection of computer science as a college major. Although research shows that computer science majors played video games more frequently than non-majors, the observation that there has been a substantial increase in game playing while computer science enrollments have been decreasing indicates that there are other factors at work (DiSalvo, et al., 2009).

Summer outreach programs are considered to be part of the solution to both low enrollments and lack of diversity (Klawe, et al., 2009), and numerous examples can be found in the literature (Peluso, et al., 2009; Groth, 2008; Hu, 2008; Adams, 2007; Scarlatos, et al., 2007). Efforts to attract and retain students during the school year have been implemented at the elementary (Lambert, et al., 2009), middle school (Marcu, et al., 2010; Hardnett, 2008), and high school (Gal-Ezer, et al., 2009; Goode, 2008; Hazzan, et al., 2008; Owens, et al., 2008; Ericson, et al., 2007) levels. All-girl environments, whether in single-sex high schools (Olivieri, 2005) or all-girl after-school and summer programs (Carmichael, 2008; Doerschuk, et al., 2007; Werner, et al., 2005), have a positive effect on high school girls’ confidence in computer usage, knowledge of computer science, participation in computer clubs, and enrollment in computer electives in high school. In spite of these efforts, the gender gap continues to worsen.

Traditional introductory computer science courses such as the AP CS A course and college CS1 courses contain an intensive learn-to-program component. Such courses are often difficult for students with no previous exposure to
computational concepts and as such discourage students from further study in the discipline. Alice is one of a handful of tools that is being used to introduce novice students to fundamental programming concepts (Daly, 2009). The use of Alice and Storytelling Alice, the latter being a version designed specifically for middle-school girls, in summer and after-school outreach programs has met with considerable success (Peluso, et al., 2009; Rodger, et al., 2009; Hu, 2008; Adams, 2007; Kelleher, et al., 2007). Such outreach programs are generally voluntary, e.g. summer camps or after-school activities. As such, participants may have been more self-motivated to some extent than the average student. Computer camps or clubs advertised as such would have been attended by students who already had an interest in the discipline. Our project involved the instruction of Alice in high school classrooms during the academic year, in an effort to include students with wide-ranging interests and varying levels of self-motivation.

2. Pedagogical Design
A two-week curriculum unit using Alice was developed to introduce fundamental programming concepts to students in middle and/or high school. Particular attention was paid to maintaining a balance between structure and flexibility. Structure is needed to make sure that the desired programming constructs are covered in a timely fashion and that every student in the class is focused and on task. Flexibility allows the students to create animations that are uniquely their own, thereby keeping them engaged. One of Alice’s strengths lies in its rich gallery of classes, allowing a group of students with widely ranging interests to create animations that are as unique as they are.

Pre and post surveys were administered to gather data on the impact that Alice had on the attitudes of the students with regard to computer science and engineering in general, and computer programming in particular. Demographic information was collected as well, e.g. gender and favorite subject.

All school districts in north central Pennsylvania were given a description of the project and its objectives and were invited to participate in the project. Two school districts accepted our offer to teach the two-week unit with a total of 5 different high school classes, 4 different subjects (Communications/Video, Web Design, Graphics I Communications, and Multimedia), and 3 different teachers. The classes contained a total of 76 students, 70 of whom completed both the pre and post surveys. In all cases, the teachers gave the students additional time following the two-week unit and the completion of the post surveys to develop Alice projects of their own design.

3. Unit Content
The unit consisted of a sequence of four stories. For each, students received a fill-in-the-blank handout that provided the structure for the story while allowing students to fill in the blanks as they wished. These handouts along with corresponding rubrics, lesson plans, and additional “tips and tricks” handouts can be found at Peluso (2010).

3.1 Story 1
The first story was built around a character, standing near some trees, who could do magic. Students began by creating their characters using either the he-builder or she-builder class in Alice’s People gallery and added trees from the Nature gallery. This motivated a discussion of Alice’s galleries, both local and web, and the object tree. At this point, we discussed the need for the camera, light, and in most cases the ground objects. By allowing students to create their own characters and scenery, students were each making something distinctively their own.

Following the storyline, we made the character lower his/her arms (using the character’s idle method), turn toward the camera, wave (using the character’s hello method), walk, and talk using built-in methods. This allowed us to introduce the concepts of method calls, sequential processing, parameters, and Alice’s do-together construct. The storyline then called for user interaction; specifically, our character tells us that he/she can do magic and asks if we would like to see. If we responded with yes, then a magic wand was to appear in the character’s hand. This motivated the use of the search feature in the galleries as we needed to find a magic wand, and the use of quad view to properly place it in the character’s hand. We specifically instructed students to place the wand in the character’s right hand, for reasons to be revealed later. In order for it to appear, it must first be invisible. This motivated a discussion of an object’s opacity property. To make the wand move with the character’s hand, students needed to properly set its vehicle property.

We then turned our attention to coding the communication with the user and introduced the use of the ask user for yes or no function of the world object. This led to the introduction of variables, and for this particular situation, Boolean variables. We then incorporated the ifelse construct to divert the storyline depending on the user’s response. If the user selected no, we used our character’s think method to display an opinion on our disinterest, and then we had him/her turn and exit the scene. We completed the else part of the code first because students had already learned enough about method calls to determine what is needed for these actions, so this served as review. This is a good point at which to introduce the ego-centric nature of Alice objects. When we tell our characters to turn right or left, they turn to their right or left, rather than the viewer’s right or left. This concept is explored in more depth in Story 2.
We then turned our attention to the if part. If the user selected yes, we changed the wand’s opacity to 100%, used the character’s hello method to wave the wand (hence the need for the wand to be in the character’s right hand), and then asked students for ideas on what the “magic” might be. Students were then free to make any “magic” happen that they wished, and ideas ranged from making a tree disappear to having a dragon appear in a puff of smoke to set the trees on fire. We allowed approximately 10-15 minutes for the students to complete their stories.

3.2 Story 2
In our second story, we focused on writing new methods and manipulating object subcomponents. The story calls for an eskimo which we found in the People gallery and an igloo for which we used/reviewed the search feature. It also calls for a hole in the ice. To model this, we used a circle from the Shapes gallery, changed it to a dark color, and rotated it so that it rested flat on the ground.

The storyline says that the eskimo is walking out of his igloo; however, unlike he-builder and she-builder, the eskimo has no idle method to bring his arms down and no walk method. We needed to create them. This motivates a discussion of subcomponents and their ego-centric nature. Using a tips-and-tricks handout, we demonstrate the use of the axes to determine the orientation of an object or subcomponent and how that knowledge allows us to correctly choose among the move, turn, and roll methods and their direction parameters.

It is worth noting that when selecting the eskimo as the character for this story, we did not allow the students to chose a different character, not even the female eskimo. This was a conscious decision on our part because not all characters have the same subcomponents, and even if they do, their orientations can vary. Dealing with those variations while demonstrating the use of the axes would have likely been chaotic.

Once these methods have been written, we introduce the loop construct to execute our walk method the correct number of times, as our eskimo is to walk to the hole in the ice and fall in. The number of times to loop will vary from one student to another, depending on how far away their eskimo was from the circle object. In some cases, the eskimo and/or circle had to be repositioned to align the objects properly. When we called our walk method multiple times, the motion appeared to be jerky. This led to a discussion of how Alice (and real) objects gradually accelerate from a resting position and gradually decelerate when coming to a stop. To make the motion appear smoother, we needed to set the style parameters on the movements’ method calls to abruptly.

Having the eskimo fall through the ice was accomplished with a move down method call with the distance parameter set to 10 meters followed by a move up of 9 meters. At this point we introduced the use of built-in sounds by accompanying the fall through the ice with the splash sound. We then used an infinite loop to make the eskimo bob up and down in the water, which is actually a move down followed by a move up.

As presented, Story 2 leaves no story components open for student individualization; there are no blanks in the story for them to fill in. In practice, there were times when some students completed the code segment at hand and had to wait patiently while others needed help. Students were allowed to explore the galleries and add additional objects while they waited, which not only resulted in some very unique worlds but also kept the students engaged and discipline problems to a minimum.

3.3 Story 3
Before beginning our third story, we talked about the difference between programs such as our animations that run from beginning to end, and programs that are driven by user actions. All students had experience with programs, e.g. word processors and video games, that once opened simply sit there, waiting for and responding to actions from the user. We introduced the terms event, listening, and event handling.

Story 3 involved a pink minnow that wants to maneuver an obstacle course. We began by demonstrating our completed world. We insisted upon the use of the pink minnow because of its existing swim method. However, students were allowed to change its color property and rename it in the object tree if they wished. We discussed how we had built our brightly colored obstacle course using various 3-D shapes from the Shapes gallery, but students were allowed to create their own obstacle courses using any objects that they liked. We allowed 15-20 minutes for students to build their obstacle courses.

This story did not include all of the event handling available in Alice but once a few have been demonstrated, the others are fairly intuitive to figure out. We chose to have our fish call its swim method while the world was running. We used the built-in let the arrow keys move <subject> to first move the camera, and then move the fish. To give the feel that the user was actually the fish moving about the scene, we set the camera’s vehicle property to be the fish. This was quick and easy, but we wanted to have more control over the speed with which the fish moved and turned. To do this, we deleted the let the arrow keys move <subject> event and created when a key is pressed events for the individual arrow keys. This led to a discussion of how various
combinations of distance and duration could achieve the speed we wanted, and we recalled that the style parameters on the method calls needed to be set to abruptly.

We also wanted our fish to be able to jump over objects. There are several ways to accomplish this. We took this opportunity to review how to write our own methods and had students create a jump method for their fish. We then created a when a key is typed event to call that method. This also provided an opportunity to demonstrate the difference between when a key is pressed and when a key is typed events, and how selecting which to use depends on how we want the program to work.

Invariably at this point, several students would ask how to prevent the fish from going through objects. We explained that this capability, which involves collision detection, is beyond what the students have learned at this point. We demonstrated some rudimentary collision detection that can be achieved in Alice and explained that more sophisticated capabilities are beyond what Alice was designed to do. We took this opportunity to describe how today’s video games are the result of months and years of design and development by a team of individuals with diverse backgrounds and abilities.

3.4 Story 4
Our goals with the fourth story, which was a modification of an exercise from Lewis, et al. (2009), were to introduce students to the use of random numbers to make our animations different every time they are executed and to show students how to add parameters to the methods that they create. When completed, the story had two frogs hopping across a road, with each hop being a randomly selected length. When at least one frog crosses the road, they both stop jumping and, if there is a winning frog, it declares itself the winner. In the case of a tie, both frogs declare that it is a tie.

For this story, we gave the students a starter program containing one frog, for which we had written a parameterless jump method, and a road. In addition to saving a little bit of time, doing so insured that the orientation of everyone’s frogs and roads were uniform. The constructs presented in this final story are challenging for beginners and this uniformity made the instruction more straightforward. In the starter program, the frog jumped 4 meters with each jump and a loop construct calling the jump method 3 times was sufficient for the frog to cross the road.

After running the starter code a few times, our first step was to add a distance parameter to the jump method. We observed that Alice automatically added the parameter to the call to jump located in my first method and gave it a default value of 1. However when running the program, the frog still jumped the same amount. We went on to explain how we needed to actually use the parameter in the appropriate places within the jump method, and we corrected the code.

Our next step was to add a numeric variable for the distance to jump, give it a random value between 1 and 4 (using the world function random number), and then pass it as the parameter’s value. We demonstrated how placing the selection of the random number before the loop caused the frog to make 3 jumps of the same distance while placing it inside the loop caused each jump to be different. The latter is what we were going for. We also had no way of knowing how many times we needed to loop to get our frog across the road. This motivated the use of the while construct.

The condition needed for the while statement, in English, is the frog has not yet crossed the road. In Alice, we needed to use one of the spatial relation functions. To determine which one we needed to use, we reviewed the tips-and-tricks handout on location and orientation and again used an axes object to determine the orientation of the road. From this we determined that we wanted the frog to continue jumping until it was in front of the road, or in other words, while the frog was not in front of the road. We completed the code and students could see that each time they ran the program, the frog jumped random distances and enough times to cross the road.

We chose to wait until this point to add the second frog because the random number and while constructs were easier for students to understand when dealing with just one frog. We also explained to the students how making a copy of our existing frog would give our second frog the jump method that we had written for the first frog. Had we added a new frog from the library, we would have needed to recreate that method for the second frog. As an added bonus, we could now say, “What do we need to do for our second frog?” Most students knew that we needed to define another variable for the second frog’s jumping distance, assign it a random value, call its jump method, and somehow change the condition in the while statement. The compound condition needed for the while statement proved to be challenging for all students, as was expected. Most simply took the code as presented on faith. The same was true with the nested if-else construct needed to determine the winner or if there was a tie. The explanation of why we needed to test for the tie condition first was lost on most students, but all enjoyed seeing the completed program execute.

3.5 Additional Assignments
All teachers participating in the project chose to extend our two-week unit by allowing students to create worlds of their own choosing. Students were given guidelines regarding the use of specific constructs or length of the animation.
4. Impact on Attitudes
Of the 76 students who participated in the project, 70 completed both the pre and post surveys.

4.1 Appeal of Alice
In Figure 1, we see that 49 (70%) of the 70 students had a positive reaction to Alice, based on their responses to the question, “How do you feel about the Alice software?” on the post survey. A chi-square goodness-of-fit test showed that there is at least one statistically significant difference among the proportions in Figure 1 ($\chi^2 = 24.29, p < 0.001$), with the highest proportion of students selecting “Like.” The limits of a 99% confidence interval for the proportion of high school students who like Alice to some degree were found to be 0.559 and 0.841. We were not surprised to find that with 99% confidence, the majority of students had a positive reaction to the software. In Figure 2, we see the responses broken down by gender. It was interesting that none of the girls selected “strongly dislike” while 9 (18%) of the boys had that most negative reaction, although the sample size was not large enough to test for statistically significant differences.

Students were asked on the pre survey to select the category into which their favorite subject falls, with the choices described as follows:
- Fine Arts (such as Art or Music)
- Language Arts (such as English, Reading, or Writing)
- Foreign Languages (such as Spanish, French, or Latin)
- Mathematics
- Science (such as Earth and Space Science, Biology, or Chemistry)
- Social Studies (such as History, Geography, or World Cultures)

Figure 3 shows the breakdown of the students’ responses based on those categories. We expected students who favor math, science, and fine arts to like Alice better than those in other disciplines, and although this appears to be the case from the data, too few students participated in the project to be able to test for statistically significant differences.

Figure 1: Response to “How do you feel about the Alice software?”

Figure 2: Response to “How do you feel about the Alice software?” by gender.

Figure 3: Response to “How do you feel about the Alice software?” broken down by favorite subject area.

Figure 4: Response to “How likely are you to install Alice on your home computer?”

Of the 70 students completing the surveys, 9 responded “yes” to the pre survey question, “Have you used the Alice software before?” The responses to the post survey question...
“How likely are you to install Alice on your home computer?” are shown in Figure 4. Interestingly, of the 9 students who responded “I already have” to the latter question, each of them responded “no” to the former question. This leads us to believe that we sparked an interest in nine students who might not otherwise have been turned on to programming.

Figure 5 breaks the responses from Figure 4 down by favorite subject area. It was interesting to see that the students who had already installed Alice on their home computers were not in just one or two disciplines. Although it appears that students who favor language arts are less likely to install Alice at home, there was an insufficient number of students to test for statistically significant results.

4.2 Attitudinal Shifts

In order to determine the effect of the unit on the students’ attitudes toward computing, several questions appeared on both the pre and the post survey forms. Although there were too few students to test for statistically significant results, we can see a positive impact in some areas. In all cases, the students’ responses were converted to an integer between 0 and 5, corresponding in the natural way to the 6 possible answers to each question. We measured attitudinal shifts by subtracting the pre survey response from the post survey.

4.2.1 Using Computers

Students were asked, “How do you feel about using computers?” We did not anticipate that using the Alice software would have much of an impact on students’ responses to this question and the data we collected confirm this. Figure 6 and 7 show the responses from girls and boys, respectively. The data support the current wisdom that girls enjoy using computers as much as boys. Figure 8 shows the shift in attitude, or more accurately the lack of a shift, that occurred as a result of the unit.

![Figure 6: Responses from females to "How do you feel about using computers?"

![Figure 7: Responses from males to "How do you feel about using computers?"

![Figure 8: Shift in attitudes in response to "How do you feel about using computers?"

4.2.2 Knowledge of Computer Programming

Students were asked, “How much do you know about computer programming?” We were not surprised to see that
participation in the project had a positive impact on the responses to this question. Figure 9 shows the pre and post survey data collected, and Figure 10 shows the shift in attitudes.

Figure 9: Responses to "How much do you know about computer programming?"

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>27</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 10: Shift in attitudes in response to “How much do you know about computer programming?”

We observe from Figure 10 that 39% (27/70) of students felt that they had learned something about programming, 44% (31/70) reflected no change, and 17% (12/70) felt that they knew more before they started the project than when it was over. We wonder if it might be the case with the latter group of students that, as the old adage goes, the more you know, the more you realize that you don’t know.

Figures 11 and 12 show the responses to this question by gender. These results are in line with the current wisdom that girls tend to rate their technical abilities lower than boys do, even when they are not.

4.2.3 Choosing a Technical Career

Students were asked, “How likely are you to choose a career in the computer field (e.g. programmer, network specialist, technician, etc.)?” Figures 13 and 14 give the responses and attitudinal shifts, respectively. Again, we observe a shift in the positive direction, but the number of participants was too small to draw any definitive conclusions. An analysis by gender is shown in Figure 15.

Figure 11: Responses from females to "How much do you know about computer programming?"

- Before: 19, 4, 4, 0, 0
- After: 3, 6, 8, 0, 0

Figure 12: Responses from males to "How much do you know about computer programming?"

- Before: 10, 13, 11, 9, 2
- After: 7, 8, 3, 0, 0

Figure 13: Responses to "How likely are you to choose a career in the computer field (e.g. programmer, network specialist, technician, etc.)?"
Figure 14: Shift in attitudes in response to "How likely are you to choose a career in the computer field (e.g. programmer, network specialist, technician, etc.)?"

![Figure 14](image-url)

Figure 15: Responses to "How likely are you to choose a career in the computer field (e.g. programmer, network specialist, technician, etc.)?" by gender.

![Figure 15](image-url)

5. Conclusions and Future Work

We agree with Bruckman, et al. (2009) and Goode (2008) that to improve recruitment, retention, and diversity, improvements must be made at every stage of the pipeline. A student attending a summer camp may become excited about computer science and want to take additional computing classes, but if none are offered at her high school, or if she is the only girl enrolled, her enthusiasm is likely to wane. Identifying those tools and approaches such as Alice that have a positive impact on students’ attitudes toward pursuing further coursework in computing is an important part of the process.

Providing a sufficient amount of quality computing coursework throughout the pipeline poses significant challenges. The Computer Science Teachers Association (www.csta.acm.org) has published a model computer science curriculum for K-12 (ACM, 2003), but changes to curriculum requirements must be addressed on a state-by-state basis, and adding computer science to the curriculum means that something else must be removed. Teacher preparedness must also be addressed. “For computer science teachers, the challenge of becoming and remaining exemplary educators is hampered by systems of pre-service education and teacher certification that are profoundly disconnected from the discipline of computer science and the needs of teachers and students.” (CSTA, 2008). Continued efforts by computer science education researchers and involvement of computer science educators at all levels are needed to address and resolve these problems.

References


Acknowledgements
We are indebted to the teachers and administrators who allowed us into their schools and classrooms. We are particularly grateful to Mark Berger at Lewisburg Area High School, and Linda Gutkowski, Chris Ulrich, and Al Stola at Montgomery Area High School.

Author Information
Eileen M. Peluso, Ph.D.
Associate Professor
Mathematical Sciences Department
Eileen enjoys teaching computer science at the undergraduate level and has had a long-standing interest in gender and diversity issues in the discipline. Since 2007, she has presented sessions using Alice at a math and science summer outreach program for middle school students at Bloomsburg University, her undergraduate alma mater. Since 2008, she has included Alice in her CS1 course at Lycoming College.

Gene Sprechini, Ph.D.
Associate Professor
Mathematical Sciences Department
Lycoming College

Gene enjoys teaching statistics courses at a variety of levels and also serves as the coordinator for the interdisciplinary Actuarial Mathematics major at Lycoming College.