A Qualitative Look at Alice and Pair-Programming

Elizabeth V. Howard
howardev@muohio.edu

Donna Evans
evansd@muohio.edu
Miami University Middletown
Middletown Ohio 45042 USA

Jill Courte
courteje@muohio.edu
Miami University Hamilton
Hamilton Ohio 45011 USA

Cathy Bishop-Clark
bishopcu@muohio.edu
Miami University Middletown
Middletown Ohio 45042 USA

Abstract

In previous studies, we have used quantitative methods to examine the effectiveness of the Alice programming language in terms of student enjoyment, confidence, and learning outcomes. However, in terms of the overall quality of a learning experience, quantitative measures provide only part of the story. This paper reports on the use of Alice from a student perspective using qualitative data gathered from student reflective exercises and focus groups. Eighty-nine (89) students from six (6) different sections of an introductory computing class for non-majors completed a 2.5-week programming module using the Alice interactive graphical programming language. Students in two (2) sections completed the Alice programming module individually and students in four (4) sections used the pair-programming paradigm, where two programmers work on the same program at the same time using the same computer. At the end of the module, all 89 students wrote a reflective essay on their experience with Alice and focus groups were facilitated in three (3) different sections. This qualitative data indicates that students who used Alice reported that they enjoyed programming, had confidence in their programming ability, understood basic programming concepts, and understood the relationship between algorithms and Alice stories.

Keywords: Alice, pair-programming, attitudes, introductory programming

1. INTRODUCTION

Students (non-majors) enrolled in a survey course designed to introduce a myriad of computing topics have often struggled with programming concepts. Learning to create a substantial program using a traditional programming language requires more exposure than a survey course typically can devote.

Additionally, non-majors often have a negative attitude towards programming. Working with a traditional programming language to produce even a simple program can be frustrating, which merely reinforces students' attitudes that programming is dull and tedious. For many students, a survey course will be the only formal exposure that they have to programming. For others, such a course creates a lasting impression of the computing field and may even be the deciding factor on whether they will major in computing.

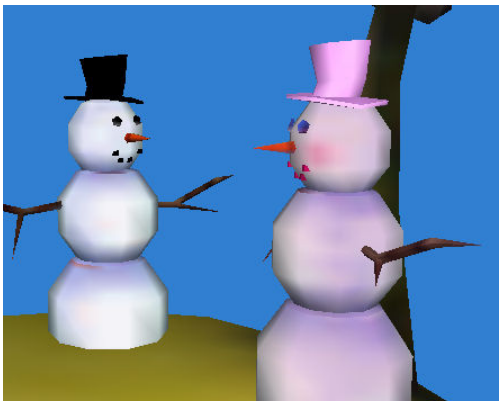

Figure 1. Screen Shot of an Alice Program

There have been many efforts to design introductory programming tools and environments in order to facilitate the difficult process of learning to program, and Kelleher and Pausch (2005) provide an excellent summary of several such tools. One of these is Alice, a 3-D interactive graphical programming language that gently introduces programming concepts. With Alice, programmers create stories in a virtual world while learning common programming constructs (Cooper, Dann, and Pausch, 2000). Figure 1 displays a screen shot of a simple Alice program. Alice programs are constructed of pre-built objects with methods that students can use to create interactions between the objects to build a story (see Figures 2 and 3). For example, to make the snowman move, a student can choose the movement by using the menu structure comprised of simple English terms (see Figure 3). Alice, created by the Stage3 Research Group at Carnegie Mellon University, is freely distributed and may be accessed through the website www.alice.org.

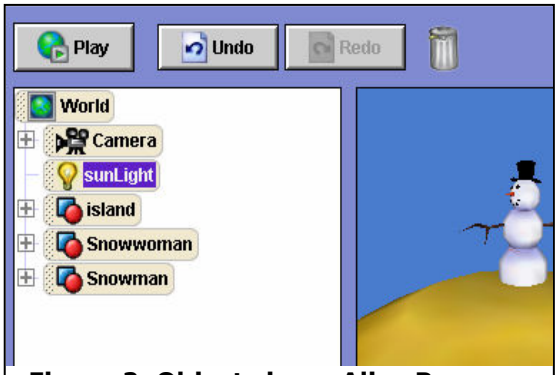

Figure 2. Objects in an Alice Program

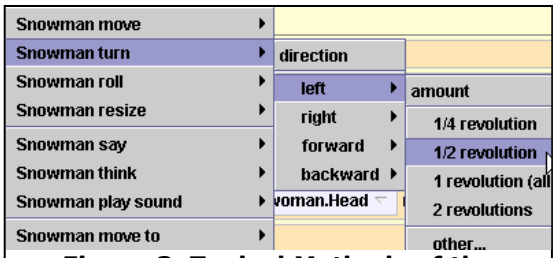

Figure 3 Typical Methods of the Snowman Object

Alice has been used successfully to introduce programming concepts in an engaging manner to students with little or no exposure to formal problem-solving methods (Cooper, Dann, and Pausch, 2003; Courte, Howard, and Bishop-Clark, in press). In one study, students in an introductory survey course reported a positive change in attitude towards programming after just a one-week introduction to programming using Alice (Courte, Howard, and Bishop-Clark, 2006; Bishop-Clark, Courte, and Howard, in press). Alice has also been used to prepare students for introductory computer science courses, particularly for students who are considered "at-risk" for failure (Cooper, Dann, and Pausch, 2000, 2003, 2004; Dann, et al, 2003; Moskel, Lurie, and Cooper, 2004). Additionally, Dann, Cooper, and Pausch (2001) introduced recursion using Alice and Cooper, Dann, and Pausch (2003) taught an "objects-first" approach using Alice.

The story-telling nature of Alice creates a natural collaboration among students. When using Alice, students routinely ask their peers for their opinions of the objects, actions, and stories. When the Alice stories are completed, the majority of students

readily demonstrate their programs. This natural collaboration led us to combine pair-programming with programming in Alice. In pair-programming, two programmers work on the same program on the same computer at the same time. "One person is the 'driver' and has control of the pencil/mouse/keyboard and is developing the design or code. The other person, the 'observer,' continuously and actively examines the work of the driver - watching for defects, thinking of alternatives, looking up resources, considering strategic implications of the work at hand, and asking questions. The observer identifies tactical and strategic deficiencies in the work (Williams & Upchurch, 2001)." Howard (*in press*) describes pair-programming partners as working "together on the same task in much the same way as an actor and a director. An actor delivers the dialogue while the director provides feedback based on a broader view of the entire production. Likewise, the pair-programming 'driver' creates the program under the direction of the 'observer' or 'navigator.'" Studies have shown that pair-programming increases understanding of programming by learning from a peer (McDowell, Hanks, Bullock, & Fernald, 2002; Williams & Kessler, 2000), reduces frustration experienced by novice programmers by having a partner with whom to reason through the program, increases student satisfaction, and fosters positive attitudes (Howard, *in press*; LeJeune, 2006; Preston, 2005; Mendes, Al-Fakhri & Luxton-Reilly, 2005; Hanks, McDowell, Draper, & Krnjajic, 2004; VanDeGrift, 2004; DeClue, 2003; Hedin, Bendix, & Magnusson, 2003; McDowell, Hanks, Bullock, & Fernald, 2002; McDowell, Werner, Bullock, & Fernald, 2003; Nagappan et al., 2003; Thomas, Ratcliffe, & Robertson, 2003).

The researchers intentionally chose to collect and analyze qualitative data from the students because "the key concern [of qualitative research] is understanding the phenomenon of interest from the participants' perspectives, not the researcher's (Merriam, 1998)." As educators, we make assumptions about our teaching and the students' feedback, whether in the form of reflective papers or focus groups, forces us to reconsider those assumptions. "The qualitative paradigm includes a reflexive stance that provides the opportunity for the researcher to examine her or his biases. (Auerbach & Silverstein, 2003)." Qualitative feedback allows students to express their opinions outside of the framework determined by researchers in surveys and other quantitative methods. By collecting qualitative data, we can focus on the *entire* experience since quantitative data often examines the individual components of the experience while qualitative data shows how those components interact (Merriam, 1998).

2. METHOD

Participants

During Spring 2006, students in six (6) sections of an introductory non-majors computing survey course participated in the Alice programming module. Reflective essays from all 89 students were analyzed and focus groups were facilitated in three sections. All of the students were undergraduates at regional campuses of a medium-sized, mid-western university. The course fulfills a liberal education requirement in the category of math, logic, and formal reasoning. Three different instructors were involved in the study and all instructors used identical handouts, assignments, and class notes.

Procedure

The Alice programming module spanned five (5) class sessions (6 hours and 15 minutes) during the third week of the semester. In the first class session, the instructors presented a brief introduction to programming terminology. The students then completed the guided online tutorials that accompany the Alice software. During the second class session, students created their first Alice world from scratch. The instructors provided the students with additional paper instructions on the basics of creating a world, adding objects, using methods along with decisions and iterations. In the third class session, students were introduced to algorithms. During this session, students created flowcharts to solve simple computing problems. Students also completed a homework assignment where they produced algorithmic solutions to several different problems. By introducing algorithms in the middle of Alice programming, we hypothesized that students would be able to recognize explicit connections between algorithms and Alice stories. In the final two sessions,

students created Alice worlds and demonstrated them to the rest of the class. In two (2) of the sections, students worked independently (31 students) and in four (4) of the sections, students worked with partners using the pair-programming method (58 students).

Reflective Essays

At the end of the 2.5-week module, we asked students to reflect on their Alice experience and answer several questions. All students answered questions about using Alice (please see Table 1) and on the relationship between algorithms and Alice (please see Table 2). Student working with partners also answered questions about pair-programming (please see Table 3).

Describe what you liked about Alice and what you did not like. Do you feel like you now understand some of the basics of computer programming? What surprised you about computer programming? What did you learn that you expected?

Table 1. Questions on Alice Programming

How are Algorithms related to what you did in Alice? What is the relationship between the symbols in a flowchart and the stories that you created in your labs? Can algorithms help you create better Alice stories? Explain.

Table 2. Questions on Relationship Between Algorithms and Alice Programming

Describe the advantages and disadvantages of programming in pairs? What did you like? What did you dislike? Did programming in pairs help or hurt your learning? Explain.

Table 3. Questions on Pair-Programming

Focus Groups

Focus groups were facilitated in a section taught by each of the instructors for a total of three (3) different sections. Students in two (2) of the sections worked with partners using the pair-programming paradigm and students in the third section worked individually. The instructor was not present during the focus groups. The researchers were trained in holding focus groups and facilitated the focus groups for one another. Students were divided into small groups and asked to discuss their Alice experience. After the small groups had concluded their discussions, the entire class was reconvened and responses were solicited from each small group. After all responses had been recorded, students voted for the responses with which they agreed. By using small groups, the researchers were not directly involved in the conversations so that students might express their opinions more freely.

3. RESULTS

The reflective essays were coded for three sets of variables intended to capture student perceptions of the learning experience and process of using Alice. The first set examined student attitudes regarding confidence, enjoyment, and overall use of Alice (please see Table 4). The second set examined student perceptions of the relationship between algorithms and Alice (Table 5). The third set examined student attitudes toward programming in pairs versus programming individually (Table 6). To establish reliability, two coders coded 25% of the essays and one of the coders coded all of the essays. The coders were reliable with a mean agreement level of .88 (range was .77 to 1.00).

Student Attitudes Regarding Confidence, Enjoyment, and Use of Alice

Table 4 contains the results of the content analysis for variables regarding attitudes, confidence, enjoyment, and overall use of Alice. As shown, this analysis was done separately for students programming in pairs and individually. Two of these dimensions showed significant results between pairs and non-pairs as described below.

A majority of students (69% overall) reported that they were *comfortable* programming in Alice and found it *easy* and *simple*. Additionally, students working in pairs reported significantly more comfort and ease of use, 78% (pairs) versus 52% (non-pairs) with a significant difference ($\chi^2(1) = 6.32$, $p = .012$).

Variable	pairs	non-pairs	overall
easy, simple, comfort	78%[1]	52%[1]	69%
did more than expected	34%	29%	33%
lack of confidence	22%	32%	26%
fun, entertaining, enjoy, like	64%	48%	58%
imaginative, creative, interesting	66%	65%	65%
dislike the programming process	2%	10%	4%
difficulty or complexity, respect for the process	40%[2]	65%[2]	48%
educational, learning, understanding	74%	77%	75%
Alice language limitations	88%	97%	91%
Alice language positives	43%	35%	40%

[1] ($\chi^2(1) = 6.32$, $p = .012$)
[2] ($\chi^2(1) = 5.00$, $p = .022$)

Table 4. Content Analysis for Questions on Alice Programming.

Overall, 33% of all students thought that they were able to *do more complex programming than they had initially expected*, while conversely, 26% reported *lack of confidence in programming ability*. Also overall, 58% of all students reported that they *enjoyed programming* in Alice while only 4% overall reported that they *disliked programming* in general. Nearly half of all students (48%) commented on the *complexity and difficulty* of programming with a significant difference ($\chi^2(1) = 5.00$, $p = .022$) between pairs (40%) and non-pairs (65%).

Overall, the majority of students thought that they were *creative and imaginative* (65%), reported that the *Alice module promoted learning and understanding* about programming (75%), and felt that *Alice software had limitations* that hampered their projects (91%). Despite this perception of limitations, 40% of all students commented on the *positive attributes of Alice*.

Student Perceptions of the Relationship Between Alice and Algorithms

In addition to use of Alice, the programming module included instruction and assignments on developing algorithms. It was hoped that the step by step nature of Alice would facilitate algorithmic thought. Table 5 contains the results of the content analysis for questions on the relationship between algorithms and Alice programming. As shown, this analysis was done separately for students programming in pairs and individually. None of these dimensions showed significant results between pairs and non-pairs.

Variable	pairs	non-pairs	Overall
what is an algorithm	97%	90%	94%
words like input/output, sequence, process, decision	84%	81%	83%
appreciate complexity	45%	32%	40%
how a story line in Alice relates to an algorithm	91%	87%	90%
positive algorithm comment	84%	81%	83%
negative algorithm comment	5%	10%	7%

Table 5. Content Analysis for Questions on Relationship Between Algorithms and Alice Programming

Overall, 94% of all students *correctly defined an* algorithm, and 83% of all students *included words typically used to describe algorithms*, such as input, output, process, sequence, or decision. 40% of all students commented on the *complexity of algorithms* and Alice programming. The majority (90%) of all students demonstrated that they *understood how a story line in Alice related to an algorithm*, and a majority (83%) also commented that *generating an algorithm*

helped them to create more complex Alice stories. Conversely, only 7% of all students reported that algorithms did not help them create better Alice stories.

Student Perceptions of Pair-programming with Alice

Table 6 contains the result of content analysis regarding comments about pair-programming from students who worked with a partner during the Alice module. Nearly all students (95%) reported that *working with a partner increased their learning*. A majority of the students (67%) thought that they were *more creative working with a partner* than they would have been had they worked independently. Reports of difficulty with pair-programming included the idea that *having a partner slowed down the pro*cess (34%), problems can arise if *partners had different ideas* about the story (38%) and that *only one (1) person is hands-on at a time* (29%).

Variable	pairs
increased learning	95%
more creativity (more ideas)	67%
slowed down process	34%
difficult if two people had different ideas about the story	38%
only 1 person is hands-on at a time	29%

Table 6. Content Analysis for Questions on Pair-Programming

Focus Groups

As mentioned previously, focus groups were facilitated in three (3) different sections. Students in two (2) of the sections worked with partners using the pair-programming paradigm and students in the third section worked individually. All students were asked to comment on what they enjoyed and did not enjoy about Alice. Overall, student comments in the focus groups were similar to comments in their reflection papers. Students in all three (3) sections enjoyed being able to create their own stories. Many students reported that the Alice environment was user-friendly, that they enjoyed Alice's graphical nature, that they were able to be creative, and that Alice was an interesting way to learn basic programming concepts. Students reported that they did not like the limited choices of methods and objects, that the tutorials did not provide sufficient understanding, and that testing and debugging were time-consuming.

Students in the two (2) pair-programming sections were asked what they enjoyed and did not enjoy about pair-programming. Students in both sections reported that they enjoyed the increased creativity resulting form merging the ideas of both partners. Students also reported that working with a partner resulted in less pressure on each individual and that finding solutions to difficulties was much faster. Students reported that they thought their own experience was limited since the hands-on work was done by one (1) person at a time, that partners working at different speeds was frustrating, and that scheduling to meet outside of class was difficult.

Students in the section where they programmed individually were asked what they enjoyed and did not enjoy about working alone. Students reported that they enjoyed the individuality including not having to rely on others and in the uniqueness of their stories. Students found that working alone was challenging especially when they encountered difficulties and did not have a partner with whom to collaborate to solve the problem. Students did not like that they were unable to divide the responsibility and labor. Some students thought that working alone was less creative.

4. DISCUSSION

Overall, students in the non-majors survey course responded favorably, both in the focus groups and in the reflective essays, to their experience of programming in Alice. Perhaps the most noteworthy response was that *only* 4% of all students indicated that they disliked the programming process. That response is a drastic departure from prior students' reactions to programming in a traditional programming language. When asked to describe what they did not like about the Alice programming module, most students (91%) commented on a limitation of the Alice software. It is because students find Alice entertaining and easy to understand that they were able to critically consider any limitations in the software, which

is another drastic difference from using a traditional programming language. By interweaving algorithms and Alice programming, students were able to make a strong connection between algorithms and the stories that they created in Alice. Students who worked with a partner overwhelmingly reported (95%) that they had learned from their partner and that they felt that they were more creative than had they worked individually.

Chi-square analysis was performed on the reflective essays to test the hypotheses that pairs would find Alice programming easier, enjoy it more, experience more creativity, better understand algorithms, and have fewer difficulties than non-pairs. As previously mentioned, the frequency of responses for pairs differed from non-pairs on *easy, simple, comfort* of the Alice Programming question ($\chi^2(1) = 6.32$, $p = .012$) and *Difficulty/complexity, respect for the process* of the Alice Programming question ($\chi^2(1) = 5.00$, $p = .022$). There were no statistical differences between pairs and non-pairs on the other variables. Although the other variables were not statistically significant, they still support a pattern of pairs being more positive about the Alice programming module than non-pairs.

5. CONCLUSION

Students in a non-majors introductory computing survey course successfully completed a 2.5-week module in computer programming using Alice. Analysis of qualitative data from reflective essays and focus groups suggests that students enjoy programming in Alice and report a positive attitude towards programming. Students were also able to understand and describe the relationship between algorithms and their Alice stories. Students working with partners believed that they had learned from their partner and that they were more creative because of the partnership. Because of the natural collaborative nature of story-telling, Alice is a good choice for introductory courses where teamwork is emphasized. Based on the results of this study, we conclude that non-majors can improve their initial computing experiences by using software such as Alice and by working in pairs. Students clearly demonstrated improved learning and attitudes toward computing, and it may be hoped that these students would take further classes in computing based on this positive first experience.

6. REFERENCES

Auerbach, C.F. & Silverstein, L.B. (2003). *Qualitative data : an introduction to coding and analysis*. New York: New York University Press.

Bishop-Clark, C., Courte, J., & Howard, E.V. (*in press*) "Programming in Pairs with Alice to Improve Confidence, Enjoyment, and Achievement." *Journal of Educational Computing Research*.

Cooper, S., Dann, W., Pausch, R. (2003 February). Teaching objects-first in introductory computer science. *Proceedings of the 34th SIGCSE technical symposium on Computer Science Education*, Reno, Nevada.

Cooper, S., Dann, W., Pausch, R. (2000 April). Alice, a 3-D tool for introductory programming concepts. *Proceedings of the 5th annual CCSC northeastern conference on The journal of computing in small colleges*, 107-116, New Jersey.

Cooper, S., Dann, W., Pausch, R. (2000 November). Developing algorithmic thinking with Alice. *The Proceedings of ISECON 2000*, *(17),* 506-539, Philadelphia, PA.

Courte, J., Howard, E., & Bishop-Clark, C. (*in press*) "Using Alice in a Computer Science Survey Course." *Information Systems Educators Journal* (also published in *The Proceedings of ISECON 2005*).

Dann, W., Cooper, S., Pausch, R. (2001 June). Using visualization to teach novices recursion. *Proceedings of the 6th annual conference on Innovation and technology in computer science education,* 109-112, United Kingdom.

DeClue, T. (2003) Pair programming and pair trading: effects on learning and motivation in a CS2 course, *Journal of Computing in Small Colleges*, 18(5), 49-56.

Hanks, B., McDowell, C., Draper, D. & Krnjajic, M. (2004). Program quality with pair programming in CS1. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer*

science education (ITiCSE 2004), 176-180.

Hedin, G., Bendix, L. & Magnusson, B. (2003). Introducing software engineering by means of Extreme Programming. *ICSE 2003: Proceedings of the 25th International Conference on Software Engineering*, 586-593.

Howard, E.V. (*in press*) "Attitudes on Pair-Programming." *Journal of Educational Technology Systems,* 35(1).

Kelleher, C., Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37:2, 83-137.*

LeJeune, N. (2006). Teaching software engineering practices with Extreme Programming. *Journal of Computing in Small Colleges*, 21(3), 107-117.

McDowell, C., Werner, L., Bullock, H., & Fernald, J., (2003). The impact of pair programming on student performance, perception and persistence. *ICSE 2003: Proceedings of the International Conference on Software Engineering*, 602-607.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education,* 38-42.

Mendes, E., Al-Fakhri, L. B., & Luxton-Reilly, A. (2005). Investigating pair-programming in a 2nd-year software development and design computer science course. *Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in Computer Science Education*, 296-300.

Merriam, S.B. (1998). *Qualitative Research and Case Study Applications in Education, 2nd edition*. San Francisco: Jossey-Bass.

Moskel, B., Lurie, D., Cooper, S. (2004 March). Evaluating the effectiveness of a new instructional approach. *Proceedings of the 35h SIGCSE technical symposium on Computer Science Education,* 75-79, Norfolk, Virginia.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., et al. (2003). Improving the CS1 experience with pair programming. *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, 359-362.

Preston, D. (2005). Pair programming as a model of collaborative learning: a review of the research. Journal of Computing in Small Colleges, *20(4)*, 39-45.

Thomas, L., Ratcliffe, M., & Robertson, A. (2003). Code warriors and code-a-phobes: a study in attitude and pair programming. *Proceedings of the 34th SIGCSE technical symposium on Computer science education,* 363-367.

VanDeGrift, T. (2004). Coupling pair programming and writing: learning about students' perceptions and processes. *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, 2-6.

Williams, L. & Kessler, R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM, 43(5),* 108-114.

Williams, L. & Upchurch, R. L. (2001). In support of student pair-programming. *Proceedings of the thirty-second SIGCSE technical symposium on* Computer Science Education, 327-331.