ABSTRACT

The benefits of collaborative programming have been well established. To better understand the problem-solving approaches that students employ during a programming task, a study was conducted at North Carolina State University last year on undergraduate students with novice programming skills working in pairs to build a game with the visual programming language Snap. This REU project has focused on designing and conducting a new study with students who have no prior programming experience working in pairs. The hypothesis is that students’ perception of the ease of use of Snap will be more positive for students with prior programming experience. To investigate this hypothesis, this REU project focused on developing a parser to interpret activity trace logs from the study, which are stored in a database. The information in these scripts will determine how well students completed the task and if their final solution is correct. The time students take to complete the assignment from both studies will also be used in assessing whether the experience level partners possess causes a more positive perception of their use of Snap. The results of this project can help inform how collaborative programming is supported in the future.

1. INTRODUCTION

Pair programming refers to a paradigm in which two students collaborate to complete a programming task. One student acts as the driver, who writes the program, and the other serves as the navigator, who gives the driver instructions and input on how to build the program. Pairs often switch roles and take turns in each role to experience both perspectives of problem solving. Research has been done to assess the benefits of paired programming. Some of these benefits are higher retention rates of students who pursue computer science. An increased meaningful interaction between students and instructors has been cited to be a result of pair programming, as well as an increase in grades of programming assignments and an increase in the enjoyment of programming [1]. This project aims to discover whether prior programming experience affects student pairs’ usability of a visual programming when using it for the first time. To determine this, data was taken from two separate studies at North Carolina State University in which student pairs from two programming courses completed a similar programming exercise within Snap, a visual programming language.

2. RELATED WORK

The present research that involves pair programming has taken many directions in determining its benefits to learning; however, it still remains unproven whether paired programming actually increases learning outcomes, such as increase in exam scores [1].

A study conducted with students from an advanced programming course at North Carolina State University assessed the collaborative strategies students took to complete a programming assignment in the visual programming language Snap. The goal was to evaluate the differences in problem solving approaches for students who
worked individually and students who worked in pairs. Results from the studies found no significant difference in amount of time students took the complete the assignment or in the overall performance of students, suggesting that student pairs performed similarly to their individual counterparts [3].

One research study evaluated the compatibility of pairs. Students from an introductory computing course participated in a study where they were paired based on their mental model consistency (MMC). A mental model is an individual’s cognitive representation of a system or knowledge, which they use to reason and make predictions. An individual's tendency to use the same mental model is their mental model consistency [2]. The researchers also examined whether student pairs migrated, that is whether a student who exhibited an inconsistent mental model moved to a more consistent mental model. One research question proposed in the paper was whether some pairs based on the MMC were more effective at migrating to a consistent model. The results showed that MMC may be a good predictor of success, however it still cannot be claimed that students perform better in these pairs or if some pairing arrangements were more effective than others.

3. RESEARCH WORK

The project this paper describes is based upon the prior research of studies conducted at North Carolina State University [3]. This project was facilitated through North Carolina State University and contributes to ongoing research in modeling the collaborative approaches of pairs.

A study was conducted with undergraduate students from an introductory programming course at North Carolina State University towards the end of this year’s summer courses. Students possessed novice-programming skills and were tasked to complete an assignment using the visual programming language Snap. The hypothesis is that students who have more prior programming experience in will have less difficulty using Snap than students with little to no prior programming experience. For comparative analysis, data and results from a previous study with students working in pairs on a similar task was used in assessing this possibility. The students in that study had one more semester of programming experience than the students in this recently conducted study.

Snap is a browser-based visual programming language tool designed to teach students computer science concepts while avoiding some of the technical errors in traditional programming language, such as syntax errors. Blocks represent coding concepts, such as conditionals and loops. Fig. 1 shows an example of a partial implementation of a rock, paper, scissors game, which students were tasked with implementing for both studies. The interface contains descriptive categories of blocks that pertain to specific actions a student might want to make within Snap. These blocks manipulate visual objects called Sprites. Blocks can be dragged and dropped and “snapped together” onto the scripting area, where the students' programs are assembled. The result of conjoined blocks is presented in area called the Stage, where Sprites perform actions as the program runs.

There were 8 student pairs in the "experienced" condition and 6 student pairs in the "inexperienced" condition. Students were recruited voluntarily and screened to ensure they did not have experience with the visual programming language Snap. Student pairs in the experienced condition possessed one full semester of Java programming experience more than the inexperienced condition. The study with the experienced pairs has students collaborate on a single computer. Students engaged in verbal communication and used the pair-programming paradigm: one student acted as the driver and the other student acted as the navigator. Student within a pair would switch roles as described in their task description. Students in the inexperienced condition collaborated remotely using Google Hangouts, a chat and video client to facilitate collaboration and communication. Students in this condition did not switch roles. One student acted as the driver and built the
program and shared the screen with the navigator who could view simultaneously on their computer and type instructions to the driver. All communication was textual.

As students from both studies were completing the assignment, their actions in Snap were being logged to a database for further analysis.

4. RESULTS

Data was used from both these studies to determine the validity of the hypothesis of whether students with prior programming experience interacted with Snap better. Two measures were used in data analysis: the total duration of the activity and the number of actions students made within the Snap interface.

Fig. 2 – Scatter plot of experienced vs. inexperienced

Fig 2 shows a scatterplot of students from both conditions relative to the total time it took to complete the assignment and the total number of actions made within Snap. Students in the experienced condition are closer together in a cluster, took relatively the same amount of time, and did not have much variation in the amount of actions. Students who were inexperienced were not closely clustered and showed more variance in the amount of actions taken to complete the assignment.

<table>
<thead>
<tr>
<th>Inexperienced Pairs</th>
<th>Experienced Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Duration</td>
<td>64.2</td>
</tr>
<tr>
<td>Std. Dev Duration</td>
<td>19.3</td>
</tr>
<tr>
<td>Mean # of Events</td>
<td>379.7</td>
</tr>
<tr>
<td>Std. Dev # of Events</td>
<td>205.6</td>
</tr>
</tbody>
</table>

Fig 3. – Table 1 displaying statistical values by condition

Table 1 shows some of the statistical values for each condition for the two measures, total duration and the number of events.

5. DISCUSSION/CONCLUSION

The Mann-Whitney U-Test, a statistical test used on two samples with small sample sizes, was used to determine if there was any statistically significant difference that could be observed from the data. The test showed that there was no significant difference in the amount of actions students performed during the activity; however, there was a statistically significant difference in the total activity duration. Students in the inexperienced condition took longer to complete the programming activity, which could be explained by their limited experience.

There are some factors that affect the validity of these results. There was a fundamental difference in the way the two studies were conducted. The experienced pairs were allowed to verbally communicate whereas the study with the inexperienced students only allowed textual communication. Since communication through typing tends to take longer than speaking, this is a factor that extended the duration for the inexperienced pairs. There were also technical difficulties intermittent in the study with the inexperienced pairs (four of the six pairs experienced system crashes on their computers). Additionally, the initial sample size for both conditions is too small to be significant and representative of both populations of students, whether experienced or inexperienced. Any future work to be conducted to prove this hypothesis would involve a larger sample size for both conditions. Additional future work would include analyzing the effect of various types of communication, textual or verbal, on students’ problem-solving approaches.

6. ACKNOWLEDGEMENTS
I would like to thank my mentor Dr. Kristy Boyer of the LearnDialogue group as well as Fernando Rodríguez, my graduate student mentor, whose research this project is based upon. I would like to thank Dr. Tiffany Barnes for being a mentor to me and organizing the IIM-REU program. Lastly, I would like to thank the DREU program for giving me the opportunity to do research at North Carolina State University this summer.

7. REFERENCES

