Pair- versus solo-programming of mini-games as a setting for learning to program: An Action Research approach

Arash Issaee
Faculty of Computer Science
University of Vienna
Vienna, Austria
issaeea91@univie.ac.at

Renate Motschnig
Faculty of Computer Science
University of Vienna
Vienna, Austria
renate.motschnig@univie.ac.at

Oswald Comber
Faculty of Computer Science
University of Vienna
Vienna, Austria
oswald.comber@univie.ac.at

***Abstract*—This Research to Practice Full Paper depicts and evaluates a secondary school project on using pair- and solo-programming of mini-games in introductory programming classes. In addition to investigating various factors influencing students' problem-solving skills (K9; age 14–15), we introduce the software metric Lines Of Code (LOC) to compare outcomes on that specific measure in the pair- and solo-programming setting. The mini-games were developed with the free personal edition of the game development engine Unity™ and C#.**

In the current study, four different classes at the secondary level were instructed and researched. All classes had approximately the same number of students, the same tasks, the same tutorials, but were using a different social setting for programming. In response to the worldwide pandemic in the years 2020 and 2021, instruction and research proceeded either in virtual or in hybrid-learning mode.

We chose participatory action research to accommodate for the complexity of factors inherent in the field as well as for its iterative, cyclic nature. The current cycle is the third of a series on studies that have investigated various aspects of introductory pair-programming. The evaluation phase employs a digital questionnaire with open and closed questions aimed to capture student's perceptions regarding problem solving. In addition, the software metric Lines Of Code" (LOC), traditionally used to measure the size of a computer program by counting the number of lines of the program's source code, was adapted to measure students' achievement in pair- and solo-programming.

With our research we aim to contribute to make learning to program more effective, engaging, and inclusive, and we would like to promote 21st century competences besides programming skills. In addition, we are eager to share our practice of pair-programming with educators in order to inspire them to experiment with pair-programming as a social setting with high potential, even in times of required social distancing.

Keywords—pair-programming, solo-programming, students, setting, problem-solving

I. INTRODUCTION

Pair-programming at schools is a style of programming in which two students share one computer, work side by side, and alternate in taking suggested roles as "driver" and "navigator". While several studies have investigated pair-programming at professional or higher educational levels [1]–[3], the present research aims to explore whether and under what conditions game-programming in a pair setting benefits students' problem-solving in the context of learning to program in secondary (K9) classrooms. Thus, the subjects of this research were 41 students (age: 14–15) in four computer science classes (CS) located in Vienna, Austria. In brief, the present study investigates students' technical outcomes in the pair- versus the solo-programming setting and explores any differences in students' coding achievements in the two settings. Besides teachers' and researchers' observation of students' approaches to problem-solving, a digital survey using questionnaires was conducted at the end of the programming task.

In order to enable self-paced learning, a tutorial had been prepared to support developing a simple Unity ™ game in the settings of pair-programming and solo-programming. Due to the restrictions caused by the COVID-19 pandemic, face-to-face instruction was strictly reduced and at times even prohibited. As a consequence, the computer science classes CS3 and CS4, who engaged in solo-programming, performed virtually, except for the last session, where students from the CS4 had the opportunity of participating in hybrid mode. Students from CS1 and CS2 who engaged in pair-programming, participated in hybrid mode. Those who worked from home participated via virtual breakout rooms that made pair-work possible.

The first and second cycle of our previous action research on pair-programming of mini-games had emphasized students' motivation, satisfaction, as well as computational thinking [4]–[6].

In the third cycle, which is described here, the focus is on students' approaches and capacities regarding problem-solving, and their acquisition of programming skills in both pair-programming and solo-programming settings. In order to measure students' achievement in coding, we adapted the software metric Lines of Code (LOC) [7]. This metric proved to be a useful instrument for comparing the programming skills acquired in the pair- versus the solo-programming setting.

Originally, pair-programming was practiced by professional software engineers as part of the Extreme Programming software development methodology [8]. A long history of research describes the benefits of collaborative learning in general [9], where pair-programming is a particular type of collaboration. Moreover, pair-programming is a particularly promising means to promote computational thinking because it encourages peer scaffolding, clear roles, and frequent feedback [10]. More specifically, previous research [3] showed that the way students are paired up makes a significant difference: Positive pairings allow students to learn from each other and help each other work through difficult new concepts. "Negative pairings can put students in stressful learning situations, where pairs do not contribute positively" [3]. According to Palmer and Rao, pairing strategies can affect student performance and satisfaction. Pairs in which both students had a similar score in their programming self-efficacy "tended to score their learning experience higher than two students in a group with a lower" similarity in their programming self-efficacy score [3].

To be successful in today's workplace, engineering and computer science students must possess high levels of teamwork skills [11]. A wide range of studies have considered the benefits of pair-programming in terms of its effect on the quality of the resulting software. These studies have taken place in both academic and commercial environments. In the commercial area, two studies are particularly noteworthy: Nosek [12] showed that pair teams significantly outperformed individuals on program quality. Jensen [13] found that the error rate for a project with pair-programming was three orders of magnitude smaller than for other similar projects. In an academic environment, the most cited study is probably the one described in Williams et al. [14], in which 13 university students worked individually on a project and 28 chose to work in pairs. The findings showed that code produced by the pairs passed more automated tests over four different programming exercises. A study by [15] involved six girls from grade 11 who were already practicing solo-programming in grade 10 and did pair-programming in their following grade. The authors found out that the girls enjoyed the subject more when programming in pairs due to their improved comprehension of the task. They especially enjoyed the socialization and communication brought about by pair-programming. The assistance, support, motivation, focus, and encouragement they received from partners when stuck or while fixing errors made the programming experience more enjoyable for them.

Previous small-scale studies conducted by the authors of the current study indicate that developing a game in pair- programming improves students understanding of programming [6]. Furthermore, pair-programming fosters students' motivation, collaboration, and computational thinking as they consult with each other, help each other and try to find a solution within their own pair [5].

For the current study, we chose a participatory action research (PAR) design [16] to reflect the complexity and the novel focus of the educational intervention. The overall action research framework allowed us to include the experience, perspectives, and learning from all stakeholders, especially teachers, students, and researchers. Additionally, various methods such as digital post-questionnaire surveys, observations, and evaluation of students' learning by considering Lines Of Code (LOC) [7] as a technical outcome, provide a rich picture of the intervention with its effects which is described in this paper. We trace a whole action research cycle encompassing the analysis of the situation, the planning, action taking, evaluation, and specification of learning.

This work aims to be of interest to teachers, instructors/educators, educational researchers, staff responsible for (further) education of computer science teachers, and potentially educational policy makers. We hope to inspire some of these audiences to embark on the challenge of learning to program - potentially even in virtual mode – by using pair-programming settings more effectively.

The findings reported in this study along with our observations in the secondary school classes showed that the pair-programming setting was superior to solo-programming regarding students' learning of programming and problem-solving. In the solo-programming classes a lack of collaboration and problem-solving were clearly observed. Furthermore, those who developed the mini-game in pairs could finish their projects faster than those in the solo-programming setting. We conclude that pair-programming of mini-games, if conducted in a way that takes into consideration specific key factors, has a high potential to keep or raise young students' interest in learning to program.

II. RESEARCH APPROACH

This paper describes action research on students' learning to program in both pair-programming and solo-programming settings in purely virtual and hybrid-learning modes at K-9 level. Furthermore, we were interested to know whether pair-programming contributed to students' problem-solving capacities. The current study builds upon two years of our research on pair-programming in the context of developing mini-games in class and presents the third cycle of action research focusing on the following research questions:

A. Research questions

1. What influence does pair-programming have on students' programming skills?
2. In which ways does pair-programming spark students' problem-solving activities more than the solo-programming setting?
3. Which factors accelerate the development of the "mini-game" when working with a second person? Which factors slow down the development when working with a second person?

B. Action Research as overall framework

According to Susman and Evered [17] cited by Baskerville [16], the Action Research method guides researchers in investigating social systems in the form of an enquiry using an interventionalist's viewpoint. "Researchers both observe and participate in the phenomena under study." [16, p.5] The particular brand of action research in which practitioners are involved as both subjects and co-researchers is referred to as Participatory Action Research (PAR) [16] [18]. PAR has successfully been applied in educational contexts, where pioneering teachers combine research with practice in acting as reflective practitioners in their own courses [19]. This is exactly the situation we encounter with the current research and educational intervention.

In this research, we take up Baskerville's proposal [16], based on [17] that suggests that action research typically proceeds in cycles consisting of five identifiable phases that are iterated: (1) diagnosing, (2) action planning, (3) action taking, (4) evaluating, and (5) specifying learning. Each of these phases is elaborated in the next section. Thus, we view this research intervention as one PAR cycle – actually our third one, as noted above. PAR as overall research framework was chosen since it satisfies our need for a multi-perspective (teachers, researchers, students), participative and iterative approach. Within this PAR framework, we take the liberty to include two complementary research instruments for improving the evaluation of specific skills. These instruments are described below. Note that even though we employ some simple quantitative measures, the observations and qualitative aspects of our enquiry are pivotal and confer with the basic philosophy of action research.

C. Instruments Used Within the Action Research Framework

1) Online survey on students' attitudes toward pair-programming

Data collection. The basic idea behind the online post questionnaire was to find out about students' attitudes and thoughts in connection with the pair-programming versus solo-programming settings. The post questionnaire had a mix of open- and closed-ended questions. Although students had mostly similar questionnaires in accordance with their experiences in each setting, several new questions were added to the post questionnaire. While the full questionnaires (in English or German) can be obtained from the authors, we list examples of questions most relevant to responding to the research question of the current study below:

- Regarding what aspects do you find the development of the "mini game" faster or slower by working with a second person? (PP)
- Regarding what aspects do you find the development of the "mini game" faster or slower? (SP)
- Could you solve any problem in the pair-programming setting (without asking your teacher)? (PP)
- Could you solve any problem in the solo-programming setting (without asking your teacher)? (SP)
- Does pair-programming help you to solve problems independently (without asking the teacher)? (PP)
- What do you learn from programming mini-games in pairs or on your own? (PP and SP)
- How do you rate your partner's contribution to pair-programming? (PP)

Data analysis. For the closed-ended questions we used descriptive statistics to depict the results. The open-ended questions were evaluated by summarizing the most frequent responses and interpreting the findings. To allow readers to better immerse themselves into the field, all students' answers were translated into English and listed. This was possible due to the brevity of most responses.

2) Using interactive tutorials to measure programming skills

In order to evaluate students' learning to program using Lines Of Code (LOC) metric, the first author designed four C# programming questions in two

interactive pdf tutorials as a kind of "interactive test". Each of the first three questions included four commented answers, in which only one of them was correct. The fourth question required writing one code-statement. In this paradigm, it was expected that each student in a pair would answer two questions over the course of taking on the role as driver. Students were allowed to copy the text-format codes from the tutorials and paste them inside their scripts in the Unity™ (Fig.2).

3) Observation

We observed virtual classes, individual students as well as each breakout room in the pair-programming settings and gave advice when questions were asked. By using breakout rooms, which is a new feature of MS-Teams[1], each student in pair-programming had the opportunity of working with their peer in a separate virtual room. Conversely, in the solo-programming setting, the breakout rooms were not used in order to be able to observe all students at the same time and help them faster if necessary. In both settings, we focused on students' behavior, problem-solving, and students' speed during game programming.

I. Experiences and Findings in Following the Action Research Cycle

A. Diagnosing

Sentence and Csizmadia [20] emphasized that some of the difficulties teachers encounter in teaching of computer programming originated in internal and external factors on the side of teachers, while others originated in students. Internal difficulties of teachers were the lack of self-confidence related to computing competencies and their inadequate knowledge in determining the appropriate pedagogy, whereas external difficulties were the lack of the instructional resources.

Our previous action research on pair-programming with Unity™ and C# [4] resulted in the following learnings that we utilize in the current PAR cycle: Students (K-9) who had participated at school "A"[2], had a positive attitude toward the pair-programming setting when programming mini-games using C# and Unity™ as a challenging task. In addition, based on students' answers when asked about their motivation [5], it was evident that students were motivated through the pair-programming setting. The majority of them wanted to learn further coding skills and even other computer software skills via pair-programming. Moreover, a survey question that asked explicitly about students' motivation to develop a computer game in pairs was answered affirmatively by all but one student.

B. Action planning

The authors of this paper have been collaborating with the respective school since 2017 as part of our previous project "learn to proGrAME"[3]. In order to get the permission from the school to work with students, observe them and publish students' outcome, the following two forms have been signed:

- "Declaration of commitment to data protection during implementation a scientific survey at Viennese schools".
- "Application for approval of a scientific survey (Section 46 (2) SchUG)".

The teachers had already taught solo-programming with C# to students. Then our research started in all classes with the collaboration of two CS teachers, one of which coauthors this paper. Based on the stated research questions concerning problem-solving and programming, the authors of this paper had designed a digital post-questionnaire and an interactive tutorial including coding test questions (see also Section 2C) in order to inquire about students' experiences and skills in both settings. Furthermore, clear instructions for the driver- and navigator-roles in the pair-programming setting had been prepared.

C. Action taking

First, we explained the pair-programming approach to both computer science classes (CS1 and CS2) and emphasized how important collaboration was in this setting. MS-Teams was used for both classes as a tool to build a bridge between students, teacher and the researchers. In the action research cycle that was conducted during the COVID-19 pandemic and its social distancing regulations, students from CS1 and CS2 classes participated in the form of hybrid-learning. In this educational model half of them stayed at home, while the other half participated personally in the physical classroom at school. In the solo-programming classes CS3 and CS4, however, all students participated in virtual mode from their homes (except for one hybrid lesson in CS4) since the research happened during a lockdown.

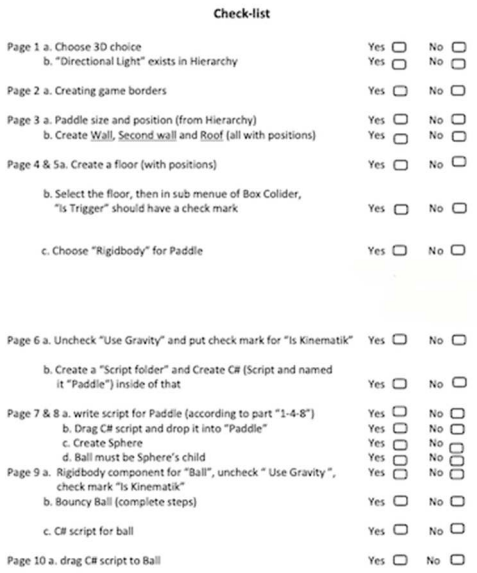
Students were asked to complete a mini-game with Unity™ and C#. All students in both settings dedicated the first part of their class time to download and install the Unity ™ and Notepad++ software as a prerequisite of our mini-game project. We instructed students on how to use the tutorial, the checklist and the "Assignment" menu as part of MS-Teams. From the second week onward, students started with developing the mini-game. Before starting with the task of developing the mini-game, we showed students in both the pair-programming and the solo-programming setting the final version of the mini-game as a demo. In

[1] Microsoft Teams is a proprietary business communication platform developed by Microsoft, as part of the Microsoft 365 family of products.

[2] Pair-programming with C# and Unity had been implemented at school A [4].

[3] https://www.sparklingscience.at/en/projects/show.html?--typo3_neos_nodetypes-page[id]=1008

the authors' view, it is vital that the students know beforehand what is asked of them. In the pair-programming classes CS1 and CS2, students were paired based on their programming self-efficiency. The programming self-efficacy of the students was estimated by their teachers, who observed and taught them programming skills from the beginning of the school semester. Importantly, in the pair-programming setting, driver and navigator roles were switched after each class. Both driver and navigator knew about their own responsibilities which had been explained to them at the beginning of the class. A driver role is taken on by students who have mouse and keyboard at their disposal, while a navigator role has specified responsibilities such as consulting the driver in the given pair, offering advice, controlling duties, and making sure that their partner develop the mini-game correctly. A checklist had been prepared (Fig2) for each navigator to have better control over the driver's part and make sure that the game was developed step by step according to the tutorial files.



Check-list

Page 1 a. Choose 3D choice	Yes ☐	No ☐
b. "Directional Light" exists in Hierarchy	Yes ☐	No ☐
Page 2 a. Creating game borders	Yes ☐	No ☐
Page 3 a. Paddle size and position (from Hierarchy)	Yes ☐	No ☐
b. Create Wall, Second wall and Roof (all with positions)	Yes ☐	No ☐
Page 4 & 5a. Create a floor (with positions)	Yes ☐	No ☐
b. Select the floor, then in sub menue of Box Colider, "Is Trigger" should have a check mark	Yes ☐	No ☐
c. Choose "Rigidbody" for Paddle	Yes ☐	No ☐
Page 6 a. Uncheck "Use Gravity" and put check mark for "Is Kinematik"	Yes ☐	No ☐
b. Create a "Script folder" and Create C# (Script and named it "Paddle") inside of that	Yes ☐	No ☐
Page 7 & 8 a. write script for Paddle (according to part "1-4-8")	Yes ☐	No ☐
b. Drag C# script and drop it into "Paddle"	Yes ☐	No ☐
c. Create Sphere	Yes ☐	No ☐
d. Ball must be Sphere's child	Yes ☐	No ☐
Page 9 a. Rigidbody component for "Ball", uncheck " Use Gravity ", check mark "Is Kinematik"	Yes ☐	No ☐
b. Bouncy Ball (complete steps)	Yes ☐	No ☐
c. C# script for ball	Yes ☐	No ☐
Page 10 a. drag C# script to Ball	Yes ☐	No ☐

Fig. 1- Checklist for navigators

In order to measure students' coding skills (via the LOC metric) in the course of game-programming, students were asked to uncomment the single correct answer from the "interactive-tutorials test" in order to execute the mini-game correctly (Fig. 1). After they finished the tests, they were asked to upload a response-sheet either as screenshots or text files in the assignment.

[4] Microsoft Forms is an online survey creator, part of Office 365. Released by Microsoft in June 2016, Forms allows users to create surveys and quizzes with automatic marking.

D. Evaluating

The digital German version of the post-questionnaire was sent via "Microsoft Forms"[4], immediately after students had finished the task of developing a mini-game. Although 41 students participated in this research, only 31 completed the questionnaire forms.

```
    // Update is called once per frame
    void Update () {

// Question 2:  if (Input.GetButtonDown("Fire1") And ballInPlay is
//false
// Please uncomment just one of the code which has the same meaning
//with the question 2:

// if (Input.GetButtonDown("Fire1") & ballInPlay == false)
// if (Input.GetButtonDown("Fire1") && ballInPlay == false)
// if (Input.GetButtonDown("Fire1") and ballInPlay == false)
// if (Input.GetButtonDown("Fire1") && ballInPlay = false)

{

            transform.parent = null;
            ballInPlay = true;
            rb.isKinematic = false;
            rb.AddForce (new Vector3 (ballVelocity, ballVelocity,
0));
        }

    }
}
```

Fig. 2- Programming question inside of the tutorial

1) Problem-solving

Pair-programming setting. Regarding RQ2 about problem-solving among students, 14 out of 17 students in CS1 and CS2 expressed positive attitudes toward problem-solving in the pair-programming setting. In response to the survey question "Could you solve any problems in the pair-programming setting (without asking your teacher)?", students who responded positively wrote: "Yes, because you remember mistakes, your partner may also notice which mistakes you make frequently.", "Yes, a lot, because you can discuss your problems in pairs.", " Yes, because you have the opinion of a second person.", and – reflecting about the process – "Yes because I can talk to someone before asking the teacher and it makes me think." Only three students were rather reserved or had a negative response about problem solving in pairs. Their responses were, respectively: "Not really!", "Difficult to say, most of the problems just occurred to me after a while.", and "No, because I don't know the program or my computer that well yet to be at the level where I can solve things by myself. I lack basic knowledge."

Solo-programming setting. Two students were identified who did not engage with their tasks in virtual mode. Although it seemed that they were online, they did not show any reaction whenever their names were called. In addition, another student also sat behind her computer but was not developing the mini-game. We simply asked her "Did you finish the first part of the

tutorial?" and we received no answer from her side. Seven out of 14 students in the CS3 and CS4 classes answered "Yes" to the survey question (Could you solve any problem in the solo-programming setting (without asking your teacher))? Just one of these seven gave an explanation by writing "Yes, because you have to work it out yourself, you learn more." In CS3 and CS4, three students responded that they could not solve any problem on their own. Additionally, one could solve some parts of the problems but was not able to handle everything alone. Examples of responses that were on the rejecting or doubting side were "Not quite, there may still be situations where you need help.", "I do not know.", "I don't think so. I think it is better to work with another person.", and "No-> There are also problems that cannot be solved alone."

Comparison. To compare the results of both settings, pair-programming helped students to understand their tasks better as they consulted each other and engaged in collaborative problem-solving. Amazingly, one student in the solo-programming setting (CS3) who stated "I don't think so. I think it is better to work with another person" clearly felt the lack of collaboration in this setting.

2) Speed of programming

Pair-programming setting. RQ3 addresses student-perceived factors influencing the speed of developing a mini-game in pair-programming mode. The respective survey-question (Regarding what aspects do you find the development of the "mini-game" faster or slower when working with a second person?) had been posed only to students in the pair-programming classes (CS1 and CS2). 13 out of 17 students in CS1 and CS2 expressed aspects that accelerated programming was better when working in pairs. One student, however, pointed to some advantages as a result of less speed (No.17). In the following, we list the whole variety of responses that were given to the respective survey question because we think that the responses nicely reflect various aspects of students' rich experience and willingness to reflect upon it:

CS1: 1. It definitely goes much faster. 2. Much better because you can exchange ideas with each other and it's more fun that way. 3. I find it faster because if you write something wrong, the 2nd can still check you. and fix it. 4. Faster, because you don't have to search everything alone. 5. Faster. 6. If you have only one monitor/screen it goes much faster and also if you have two monitors it is easier because you can concentrate more/better on programming. 7. Faster because you learn more in pair. 8. I think it's faster because I don't know much about it (programming/game developing), so I can exchange ideas with someone else.

CS2: 9. Faster. 10. Reading the code aloud, as opposed to looking at the code myself, was much slower. 11. Both because you can help better but also get distracted. 12. It is slower when you do it with a second person, because you usually distract each other. 13. It is faster. 14. Faster, because it is easier to understand some things in two. 15. When creating objects and programming. 16. You don't have to switch back and forth between desktops. 17. I find it slower, but it has its advantages. Furthermore, concerning the survey question regarding students' speed in the pair-programming settings, 76.47% of the respondents agreed that working in pairs, speeded up the process of developing the mini-game.

3 Interactive test on programming code (LOC).

Ultimately, to respond to RQ1 (What influences does pair-programming have on students' programming skills?), technical outcomes of students' programming were examined by using the Lines of Code (LOC) metric. In this paradigm, there were four C# programming tasks in each setting. As mentioned before, the tasks were specified inside the tutorial and in the case of the first three tasks all coding alternatives had been commented inside of the tutorial pdf files (Fig. 2). Students were asked to uncomment the one correct answer out of four (one correct and three false) C# statements. The fourth task required students to code one statement. Overall, with counting the number of students' correct answers in each setting, we were be able to evaluate an essential aspect, namely coding, of students' learning to program.

Pair-programming. As a collaborative task, each pair answered four questions after consulting with their peers.

CS1 included nine students. While the number of students showed an odd number, we could make three pairs plus a group of three students as an exception. In this way, we collected 16 answers from students:

((3 pair * 4 questions) + (1 group * 4 questions)).

CS2 included twelve students (six pairs). Thus, we collected 24 answers from this class (6 pairs * 4 questions). After counting the number of the correct answers (LOC), 19 correct answers have been identified. To sum up, 35 correct answers out of 40 answers have been collected from pair-programming classes. This amount to 87.5% of correct answers.

Solo-programming.

CS3 included seven students. As a consequence, each student answered four questions, hence 28 answers have been collected. After counting LOC, 16 answers were correct.

CS4 had 13 students but only eleven students responded to the programming tasks. Using the same procedure to determine the correct LOC as in CS3, in the solo-programming setting of CS4 29 out of 44 answers were correct.

In brief, 45 correct answers out of 72 answers (amounting to 62.5%) have been collected from both solo-programming classes. Last but not least, as we had a different number of students in each setting as well as different methods of collecting their data, we calculated the percentages of correct answers in both settings in order to compare students' outcome from pair-programming classes with solo-programming classes. The Table.1 and Table 2. depict students' details regarding LOC and percentages of correct responses in pair- programming and solo-programming settings. As we had assumed, pair-programming helped students to understand programming more thoroughly as they consulted each other and engaged in a collaborative setting. In general, 87.5% of the students in CS1 and CS2 answered the programming questions correctly, whereas just 62.5% of answers in CS3 and CS4 were correct (besides three students dropping out), which shows that the pair-programming setting resulted in a higher number of correct LOC in comparison to the solo-programming setting.

TABLE I. TABLE. 1 LINES OF CODE (LOC) IN PAIR-PROGRAMMING

	CS1	CS2	Sum (PP)
Total answers	16	24	40
Correct answers	16	19	35
Correct answers in percentage	-	-	87.5%

TABLE. 2 LINES OF CODE (LOC) IN SOLO-PROGRAMMING

	CS3	CS4	Sum (SP)
Total answers	28	44	72
Correct answers	16	29	45
Correct answers in percentage	-	-	62.5%

3) Findings from observation

Over the course of developing the mini-game in all the four computer science classes, the researchers and the teachers observed the behavior and collaboration between students. In addition, pair-programming in hybrid-learning mode was a completely new experience among students and the researchers. On closer inspection, it requires professional management in order to build a connection between students who were paired together and the task of transferring their projects to each other in virtual modes. We observed that both driver and navigator were playing significant roles while they were consulting most steps in developing the mini-game. We realized that pair-programming encompassed computational thinking for students in the pair setting as they were consulting each other, learning from each other, and trying to find a solution with their own peer in this approach. Furthermore, they were trying to solve their problems as a pair and avoided asking their teacher or the researcher too many questions. Conversely, most of the students in CS3 and CS4 classes were not able to manage sophisticated tasks during programming in the solo-programming setting. Moreover, it seemed that some of them did not intend to concentrate or even work alone in the virtual learning mode. They felt more freedom because they were far away from their teacher, the researchers and even other classmates. We observed that six students in the solo-programming setting were not working on their tasks because they did not show environment. In this paradigm, lack of collaboration and motivation was observed in the virtual mode. Considerably, one of the teachers shared his observation regarding the pair-programming setting as: "The pair-programming setting looks very interesting to me and my students. It seems that students know what they should do because they have different responsibilities with a similar goal".

E. Specifying learning

Observing the working process brought valuable insights. Students in the solo-programming setting faced more problems during developing the mini-game, as they were not allowed to consult with their classmates. To make the process of assisting students with their problems more effective, the teacher created a virtual breakout room for the researchers. Then whenever more than one student needed help, the teacher sent one of them into the breakout room so that the researchers could help them. In parallel, the teacher helped the other students on the main stage of MS-Teams, where all but one student could listen. In this paradigm, we were able to manage students' problems efficiently despite the challenge of dealing with a virtual or hybrid-learning setting.

As another consequence of the participatory action research, we suggest using a cloud storage for the pair-programming settings in the hybrid-learning mode. A cloud drive not only gives students the opportunity to save their projects in a safe and portable storage, but also gives the navigators the right to access the saved project file in order to be able to continue with the project when they take over the driver's role. Moreover, students would not lose their project whenever they would shift from home to school and also in reverse. Furthermore, it is also important to consider the risk of a weak or broken internet connection, especially for those who work from home. It is worth to mention that using a cloud storage could have some difficulties for students, due to size of a project as well as the internet download speed and the necessity to open the project as an "existing project" on their computers. Another issue we encountered was that one student had a 32bit OS and could not install

the latest version of the Unity™, which was not such a big deal for this research. However, these kinds of issues should have been predicted and organized in the hybrid settings in order to improve the experience of users of 32bit operating systems.

II. Discussion and Further Work

Our study comes with a number of limitations. Due to the COVID-19 restrictions, which occurred during our research, one of our classes had to be changed from hybrid mode to virtual mode. Thus, we were able to investigate the classes only in virtual and in hybrid-learning mode, resulting in small sample sizes. Another limitation was that students' programming self-efficacy had been estimated by their teacher; a pre-test to be completed by the students beforehand had not been used to evaluate their programming self-efficacy.

We see yet another limitation in the fact that all classes stemmed from the same school and the same level (K9), and all worked on the same mini-games using the same tutorials. Hence our results may be biased in terms of the influence of the school, the two teachers, the didactical approach, the wording of the survey, and the virtual/hybrid settings. Additionally, the school and the classes were not selected randomly but depended on the researchers' access to the teachers. In addition, we had to make one exception when pairing the students because one pair-programming class had an uneven number of students. We, therefore, formed three pairs and one group with three (a total of nine students).

Despite these limitations, our action research, surveys and programming skills tests showed strong tendencies and convincing observations in the features we explored. Thus, this research cycle provided valuable orientation in the field and motivation for sharing our findings and following up the pair-programming instruction as well as the research approach. Last but not least, the pandemic-induced necessity to switch classes into pure virtual mode brought about a new valuable insight that otherwise would have stayed concealed: Pair-programming in introductory programming courses proved to be effective even in the purely virtual mode! This opens up a new opportunity to pair students across different locations and, thus, to extend the reach of this highly potent way to learn to program to students with restricted mobility and/or distant locations and diverse cultures.

III. Conclusions

In a nutshell, secondary-level students (K9) who developed mini-games in pair-programming classes had superior results regarding coding - and problem-solving skills than students who solved the same tasks on their own. Moreover, the vast majority of students found that pair-programming accelerated coding. Intriguingly, in their qualitative responses, students identified numerous influential factors that contributed to their positive and occasionally also negative experience in coding in the pair- as well as solo-programming setting. For combining observations with qualitative and quantitative responses, participatory action research proved to be very helpful in illuminating multiple perspectives of the complex task of learning to program. As authors, we hope that our mixed methods research on collaborative ways to learning to program provides inspiration for like-minded educational innovators.

Acknowledgment

Thanks are due to the students of the 5th grade of the four participating secondary classrooms in Vienna, Austria, as well as to their computer science teachers for their participation in the study! Furthermore, the authors thank the University of Vienna for supporting the research and its presentation at the Frontiers in Education Conference 2021 (FIE).

References

[1] C. Mcdowell, L. Werner, H. E. Bullock, and J. Fernald, "The impact of pair programming on student performance, perception and persistence," 25th International Conference on Software Engineering, 2003. Proceedings., 2003.

[2] C. McDowell, L. Werner, H. E. Bullock, and J. Fernald, "Pair programming improves student retention, confidence, and program quality," Communications of the ACM, vol. 49, no. 8, pp. 90–95, 2006.

[3] J. D. Palmer and J. Rao, "Pair Programming and Self-Efficacy in CS1", Proceedings of the 49th ASEE/IEEE Frontiers in Education Conference (FIE), October 16-19, Cincinnati, Ohio, USA, 2019.

[4] A. Issaee, "Under what conditions does the pair-programming setting lead to kind of improvement in developing a game at secondary school?". In E. Langran (Ed.), Proceedings of SITE Interactive Conference 2020 (pp. 486-498). Association for the Advancement of Computing in Education (AACE)., 2020.

[5] A. Issaee, R. Motschnig, O.Comber, 2019. "Pair-Programming of video games at a secondary level classroom - concept and case study. "World Conference on Educational Media and Technology (EdMedia), Amsterdam, Netherlands, 2019

[6] A. Issaee, R. Motschnig, O.Comber, "LEARNING TO CODE IN CLASS BY PAIR-PROGRAMMING OF GAMES." , DisCo: E-learning: Unlocking the Gate to Education around the Globe - 14th conference reader, Prague, 2019

[7] D. Spinellis, G. Gousios, V. Karakoidas, P. Louridas, P. J. Adams, I. Samoladas, and I. Stamelos, "Evaluating the Quality of Open Source Software," Electronic Notes in Theoretical Computer Science, 25-Mar-2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066109000632. [Accessed: 27-Jun-2021].

[8] K. Beck, "Embracing change with extreme programming" Computer, vol. 32, no. 10, pp. 70–77, 1999.

[9] B. Barron, "Achieving Coordination in Collaborative Problem-Solving Groups", Journal of the Learning Sciences, vol. 9, no. 4, pp. 403-436, 2000. Available: 10.1207/s15327809jls0904_2.

[10] K. D. Ciani, J. J. Summers, M. A. Easter, and K. M. Sheldon, "Collaborative learning and positive experiences: Does letting students choose their own groups matter?," University of

Arizona, 15-Jul-2015. [Online]. Available: https://arizona.pure.elsevier.com/en/publications/collaborative-learning-and-positive-experiences-does-letting-stud. [Accessed: 27-Jun-2021].

[11] R. Lingard and S. Barkataki, "Teaching teamwork in engineering and computer science," 2011 Frontiers in Education Conference (FIE), 2011.

[12] J. T. Nosek, "The case for collaborative programming," Communications of the ACM, vol. 41, no. 3, pp. 105–108, 1998.

[13] R. Jensen, "A pair programming experience. The Journal of Defensive Software Engineering" 16 (3), 22–24, 2003.

[14] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," IEEE Software, vol. 17, no. 4, pp. 19–25, 2000.

[15] J. Liebenberg, E. Mentz, and B. Breed, "Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT)," Computer Science Education, vol. 22, no. 3, pp. 219–236, 2012.

[16] R. L. Baskerville, "Investigating Information Systems with Action Research," in Communications of the Association for Information Systems, vol. 2, available online: http://cais.isworld.org/articles/2-19/, 1999.

[17] G. I. Susman and R. D. Evered, "An Assessment of the Scientific Merits of Action Research," Administrative Science Quarterly, vol. 23, no. 4, p. 582, 1978.

[18] R. Motschnig, D. Pfeiffer, A. Gawin, P. Gawin, M. Steiner, and L. Streli, "Enhancing Stanford Design Thinking for Kids with Digital Technologies A Participatory Action Research Approach to Challenge-Based Learning," 2018 IEEE Frontiers in Education Conference (FIE), 2018.

[19] S. Ottosson, "Participation action research - A key to improved knowledge of management," Technovation, vol. 23, pp. 87-94, 2003.

[20] S. Sentance and A. Csizmadia, "Computing in the curriculum: Challenges and strategies from a teacher's perspective," Education and Information Technologies, vol. 22, no. 2, pp. 469–495, 2016.