World Transactions on Engineering and Technology Education *© 2021 WIETE*
Vol.19, No.2, 2021

Teaching recursive thinking using unplugged activities

Mohammad A. Kuhail, Joao Negreiros & Ahmed Seffah

Zayed University
Abu Dhabi, United Arab Emirates

ABSTRACT: Traditionally, recursion is primarily taught using coding activities, presenting a high cognitive load for novice programmers. A possible method to teach recursion to novice programmers is the use of *unplugged activities*, engaging tasks not involving coding. Despite having been shown to improve students' understanding of computational thinking, it has been argued that unplugged activities alone are not effective to teach recursion. In response, the authors engaged novice programmers who are non-computer science students by utilising unplugged visual activities to illustrate basic recursion concepts. Thereafter, the students were shown how to implement the activities with coding. The authors created three activities that illustrate recursion using breadth-first search (BFS) and depth-first search (DFS) algorithms. An evaluation study with 19 students was conducted in an on-line teaching environment. Empirical results show that most students learned the covered activities. Further, the students reported that the activities were engaging, and helped them learn how to think recursively.

INTRODUCTION

Traditionally, teaching recursion to novice students has been considered as a challenging task [1]. Educators have employed various strategies to teach programming in general and recursion specifically. Examples of such strategies include microlectures [2], open-ended questions [3], concrete examples [4] and games [5]. While promising, these strategies focus on coding early on, which presents a high cognitive load, particularly if the students are new to the syntax [6]. In the authors' experience, it is crucial to capture students' interest prior to involving them in implementation details.

Unplugged activities, tasks that aim at engaging students without the involvement of computers [7], have been proven to enthuse students and improve their understanding of computer science (CS) programming concepts [8]. Some researchers have utilised unplugged activities in teaching recursion [9][10]. However, these studies do not include evaluations that provide statistically significant evidence that substantiate the proposed teaching. Further, it has been argued that unplugged activities alone are not adequate to effectively teach recursion [11].

In response, the authors introduced recursion to novice programmers who are non-computer science undergraduate students using unplugged visual activities initially. To cement their understanding, the students were shown how the activities can be implemented with Python. The objective was to stimulate students' thinking and evoke their curiosity to think recursively, while also show them how recursion can be practically implemented. The activities target recursion topics using commonly known algorithms, such as breadth-first search (BFS) and depth-first search (DFS) of a graph. Further, to bridge the gap of lack of empirical evidence of teaching effectiveness in the previous studies, the authors conducted an evaluation study with 19 students in an on-line teaching environment. The results show that most students learned the activities. Moreover, they reported that the activities were engaging and helped them to understand how to think recursively.

RELATED WORK

Research has shown that unplugged activities can increase interest in computer science when used in various settings [8]. Thies and Vahrenhold showed that unplugged activities could be effective if inserted into a traditional lesson that included worksheets and discussions [12]. The authors used unplugged activities to teach computing concepts, such as binary numbers, binary search and sorting networks. Korhonen and Vivitsou assigned groups of students to use digital storytelling in the form of short films to communicate the concept of recursion [13]. Nevertheless, the developed stories did not include an explanation for how to implement these recursive stories. Gunion et al [9], and Pollard [10] developed unplugged activities to teach recursion, and the results showed encouraging feedback. However, the results were not statistically significant.

To conclude, unplugged activities appear to be promising in motivating and engaging students, but more research is needed to prove their effectiveness especially in a higher education setting. This study builds on the success of unplugged activities in motivating students, but also attempts to reinforce the students' understanding by showing them how to practically implement the activities with Python. Further, this study attempts to provide empirical evidence to substantiate the use of unplugged activities to teach recursion.

METHOD

In designing the study, the objective was to obtain feedback from students on the use of unplugged activities in teaching recursion, to find out if unplugged activities are effective as a teaching tool, and whether they improve students' interest and engagement. Prior to conducting the study, the authors had carried out a survey with 26 non-computer science students to identify their familiarity with recursion. The results of the survey show that most surveyed students were not familiar with recursion. The details of the survey can be found in the survey section.

The authors received an approval from the research ethics committee to conduct the study. The study was conducted in an on-line setting using Adobe Connect, a software application for remote training and Web conferencing [14]. The participants were recruited by sending e-mails to students from previous semesters, as well as making announcements in classrooms. The students who volunteered to participate were given an experiment code. This code was used to refer to the students' data.

The authors conducted the study with 19 undergraduate non-computer science students majoring in technology-related fields, such as multimedia or Web design. The participants had taken a beginner programming course, and hence were familiar with basic concepts, such as defining variables and functions, but had not taken core computer science courses, such as data structures that cover recursion. Eight of the participants were fourth-year students, eight were third-year and three were second-year students. The students rated their interest in programming on a scale from 1 to 5 (where 1 means no interest and 5 means very high interest) as follows: five students: 5 out of 5, eleven students: 4 out of 5, and three students: 3 out of 5.

Study Plan

The study consists of four steps. First, students take a pre-test consisting of four questions, which allows the teacher to assess the students' theoretical and practical knowledge of recursion prior to being engaged in an unplugged activity. Students can take up to 15 minutes working on the pre-test. Second, students are engaged with an unplugged activity for 20 minutes. The authors split the students into three groups, each group learning one of the activities separately. Third, students take the post-test, which is the same as the pre-test. This gives the teacher a benchmark for measuring the improvement of students' understanding of recursion. Finally, students fill out a questionnaire. In the questionnaire, the students report on their experience. The details of the study are explained in more details in the following sections.

Unplugged Activities

Figure 1 shows the three unplugged activities that were used in this study. The activities are animated and illustrated visually to help students recall them more easily [15], and interactive as students are required to participate in them. Instructors may explain a few rules to initiate the activity, but students steadily contribute to the activity. The activities are familiar as they touch on problems that are likely to be known; helping the students in developing the correct mental models, which is crucial for understanding [16].

Yet the activities are thought-provoking because it is likely that students have not thought to solve them recursively. Finally, to encourage reproducibility, the activities are designed to be executed in a short time and at a minimal effort from instructors desiring to adopt the activities in their classes. Each of the activities is stand-alone and designed to last 20 minutes, including showing the students how to implement them with Python code. While the design of the unplugged activities has unique and novel elements, it was inspired by other unplugged activities suggested by the community [17] and examples in data structures textbooks [18].

In an on-line setting, the authors used MS PowerPoint animation to show the students the activities step by step. Further, the students were required to draw the activities and follow up with the steps with a pencil. The full documentation of the activities including the MS PowerPoint animated slides, video, and evaluation can be found in unplugged activities documentation on the Web [19].

Robot Cleaner

Based on maze algorithms [20], the robot cleaner was designed to illustrate depth-first search recursion using a 2D maze. The idea is that a robot needs to clean cells in a maze. The cells in a maze can have one of these four states: dirty, cleaning (currently being cleaned), barrier (a cell the robot cannot be in) and a clean cell. Initially, the cells are marked as dirty or barrier (Figure 1). The robot is initially placed at cell 0,0. For the robot to clean the maze, it can move in four directions prioritised in this order: right, left, up, down. When in a dirty cell, the robot marks it as *cleaning* (visualised with a broom). Thereafter, the robot explores the next possible move, which should only be a dirty cell. If the robot cannot make the next move (because all the movements lead to a cell that is not dirty or out of bounds), the robot cleans the cell (by marking it *clean*) and backtracks. Continuing with this recursive approach, the robot ends up cleaning all the cells, and going back to the location it started with (cell 0,0).

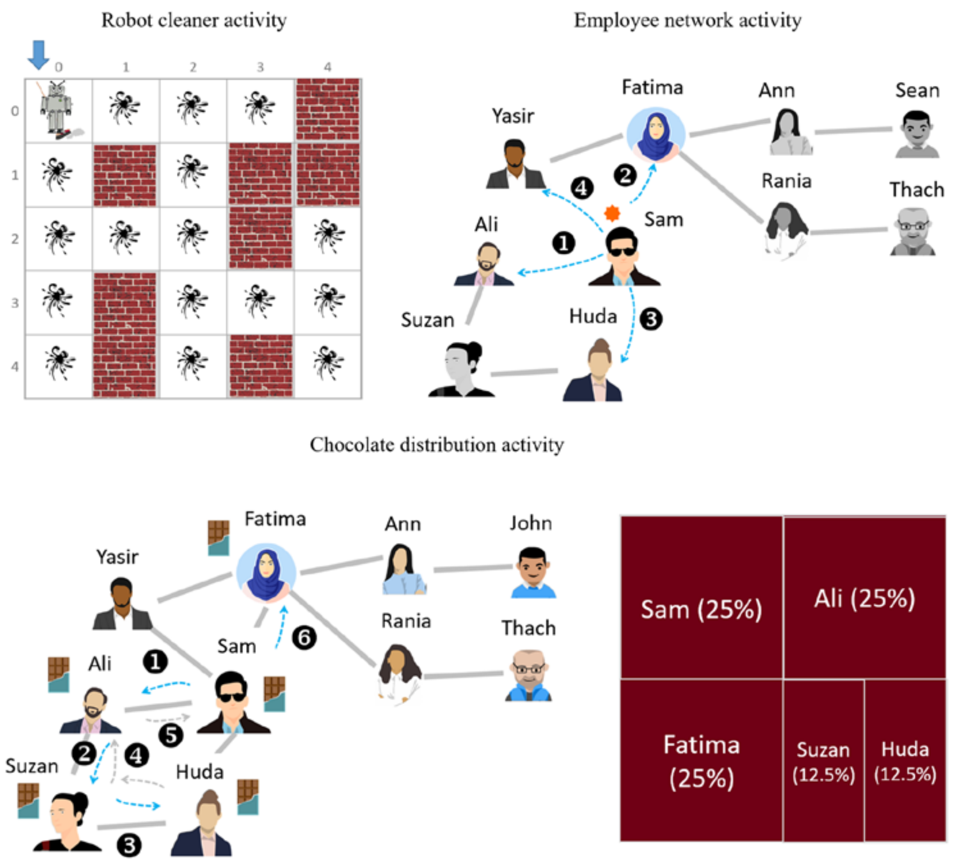


Figure 1: The unplugged activities used in this study.

Employee Network

The employee network unplugged activity illustrates recursion with BFS. The activity shows how news spreads in a network of employees. A random employee, for instance Sam, hears some news. As a result, he is added into a queue. Thereafter, the employee is removed from the queue, and then he shares the news with other co-workers sorted in alphabetical order. Every time a new employee learns about the news, he/she is added to a queue. Once done with passing the news to all his/her co-workers, the step is repeated recursively for the next employee on the queue. The next employee will execute the same procedure with his/her co-workers, but he/she will only share the news with those who have not heard it. Following this recursive path, eventually everyone in the network will hear the news (if it is a connected network environment). Visually, a blue arrow is used to illustrate that the news spreads to an employee. Further, the queue is visually changed, where the grey colour indicates that the employee is removed from the queue, whereas the black colour indicates that he/she is still in the queue.

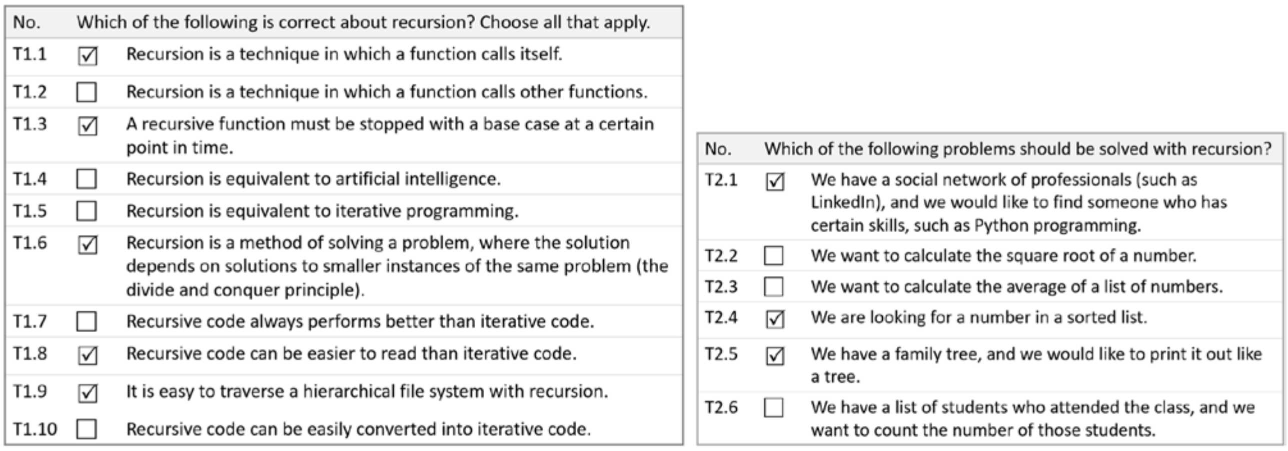
Chocolate Distribution

This activity illustrates recursion using DFS. The idea is that a bar of chocolate is recursively shared (by splitting it) with all the employees in the network (Figure 1). First, a random employee, for instance Sam, receives the bar of chocolate. He splits it into two halves, keeps a half, and gives the other half to his first co-worker alphabetically (Ali). Next, Ali will do the same, i.e. share half of his portion with Suzan, etc, and then Suzan will share her half with Huda. However, Huda has no one to share the chocolate with (because the other co-worker Sam already has his portion of the chocolate), so she goes back to Suzan (backtracking), and Suzan goes back to Ali, and Ali to Sam. Now Sam will share half of his with the next employee alphabetically (Fatima), and so on until everyone in the network has got a portion of the chocolate. Further, a blue arrow is used to indicate that the chocolate was shared between co-workers, whereas a grey arrow represents backtracking (the employee goes back to her co-worker who shared the chocolate telling her that the sharing is done).

EVALUATION

Theoretical Evaluation

Figure 2 shows the two theoretical questions that were used, as well as the answers to these questions. The students were asked to select all possible answers to questions 1 and 2. All the 19 students worked on the two theoretical questions in the pre-test and post-test. The questions were inspired by foundational recursion questions presented in popular data structures textbooks [18]. Some questions have been directly touched on during the activity (such as T1.1), while others were not explicitly mentioned (such as T1.9 and T1.10) to allow the teacher to evaluate whether students would deduce the answer. In grading the students' attempts, 1 point was given for each correct answer and -1 point for each wrong answer. Hence, the maximum grade a student could get was 10 points for question 1 and 6 points for question 2, whereas the minimum grade was -10 points for question 1 and -6 points for question 2.



No.	Which of the following is correct about recursion? Choose all that apply.	
T1.1	☑	Recursion is a technique in which a function calls itself.
T1.2	☐	Recursion is a technique in which a function calls other functions.
T1.3	☑	A recursive function must be stopped with a base case at a certain point in time.
T1.4	☐	Recursion is equivalent to artificial intelligence.
T1.5	☐	Recursion is equivalent to iterative programming.
T1.6	☑	Recursion is a method of solving a problem, where the solution depends on solutions to smaller instances of the same problem (the divide and conquer principle).
T1.7	☐	Recursive code always performs better than iterative code.
T1.8	☑	Recursive code can be easier to read than iterative code.
T1.9	☑	It is easy to traverse a hierarchical file system with recursion.
T1.10	☐	Recursive code can be easily converted into iterative code.

No.	Which of the following problems should be solved with recursion?	
T2.1	☑	We have a social network of professionals (such as LinkedIn), and we would like to find someone who has certain skills, such as Python programming.
T2.2	☐	We want to calculate the square root of a number.
T2.3	☐	We want to calculate the average of a list of numbers.
T2.4	☑	We are looking for a number in a sorted list.
T2.5	☑	We have a family tree, and we would like to print it out like a tree.
T2.6	☐	We have a list of students who attended the class, and we want to count the number of those students.

Figure 2: Theoretical evaluation questions.

Practical Evaluation

The authors used practical evaluation for the three unplugged activities. The questions tested their students' ability to trace recursive code. Hence, for each question, a block of recursive code was shown together with an illustrating figure. Since the authors split the students into three groups, students in each group worked only on the practical questions related to the activity taught to that group. The first group of students (S1-S5) worked on the robot cleaner practical question, whereas the second group (S6-S12) worked on the employee network, and the third group (S13-S19) worked on the chocolate distribution question.

Robot Cleaner

The students are presented with the scenario given in Figure 3. The robot is placed at cell 0,0. The student is supposed to write down the steps the robot takes to clean the maze. For instance, in the present figure, the robot first marks cell 0,0 as *cleaning*, then moves on to cell 0,1 and marks it as *cleaning*, etc. In grading the answers to these questions, the accuracy and the order of the steps regarding each grading are considered, that is, students are only given a point if they identify a step, as well as its correct preceding step.

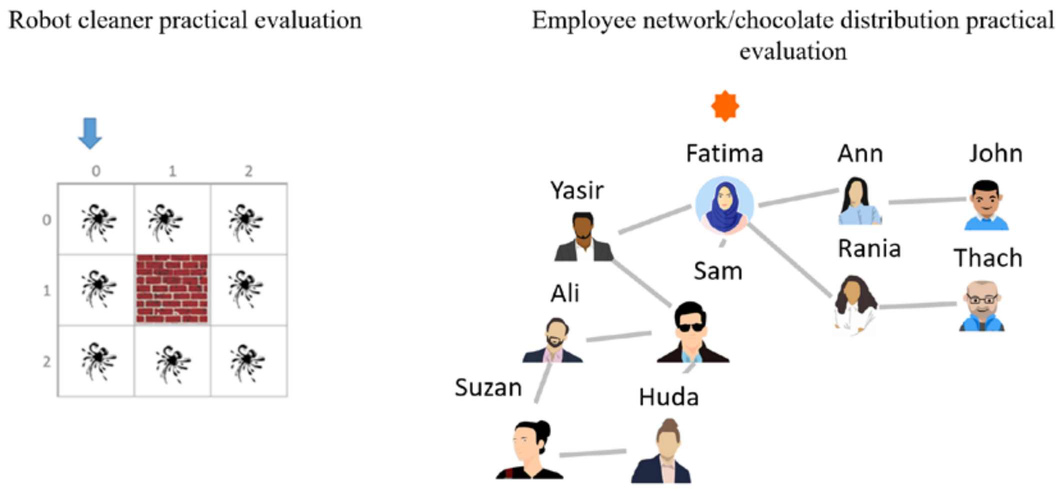

Robot cleaner practical evaluation
0
1
2
0
1
2
Employee network/chocolate distribution practical evaluation
Fatima
Ann
John
Yasir
Rania
Thach
Sam
Ali
Suzan
Huda

Figure 3: Practical evaluation questions.

Employee Network

The students are presented with the scenario given in Figure 3, where Fatima hears some news, and the students are required to write the steps of what happens next so that the news spreads in the network. Since there are 6 steps, the problem's credit is six points (one point for each step). Further, the accuracy and the order of the steps are considered. Students are only given a point if they identify a step, as well as its correct preceding step.

Chocolate Distribution

The students are presented with the scenario given in Figure 3, where Fatima receives the chocolate, and the students are required to write the steps of what happens next, so that the chocolate is distributed in the network. The answer consists of 19 steps. Each correct step is worth one point. Like the previous questions, the identification of the correct steps, as well as the order of the steps are considered in grading.

Questionnaire

The students were required to fill out a questionnaire as the last step in the study. The questionnaire asked the students if the nature of the on-line environment made the activity challenging. Further, the students were asked whether the unplugged activities were enjoyable, easy to learn, and if they wanted to see such activities implemented in classes. Additionally, the questionnaire open-ended questions asked students what they enjoyed and/or disliked, and whether they had suggestions for improvement.

Results

Figure 4 shows a bar chart showing the performance improvement in questions 1 and 2. The improvement has been calculated by subtracting the students' performance in the pre-test questions from their performance in the post-test. Concerning question 1, some students, such as S2, received a significantly higher grade in the post-test results, whereas others, such as student S1, received a lower grade in the post-test. In general, students did well on sub-questions the instructor explicitly communicated during the teaching session, such as T1.1 and T1.6 (both average improvement of 0.6 points), but their performance regressed on sub-questions that the instructor implicitly covered, such as T1.4 (average regression -0.9 point). Nevertheless, overall, for a 95% level of confidence, there is a statistical difference before and after the intervention (*p* value = 0.008), confirming the positive effect of this unplugged activity. Concerning question 2, students improved in all sub-questions, except for students S2 and S14. In fact, after performing the paired *t*-test, for a 95% level of confidence, there is again, a statistical difference before and after the intervention (*p* = 0.028).

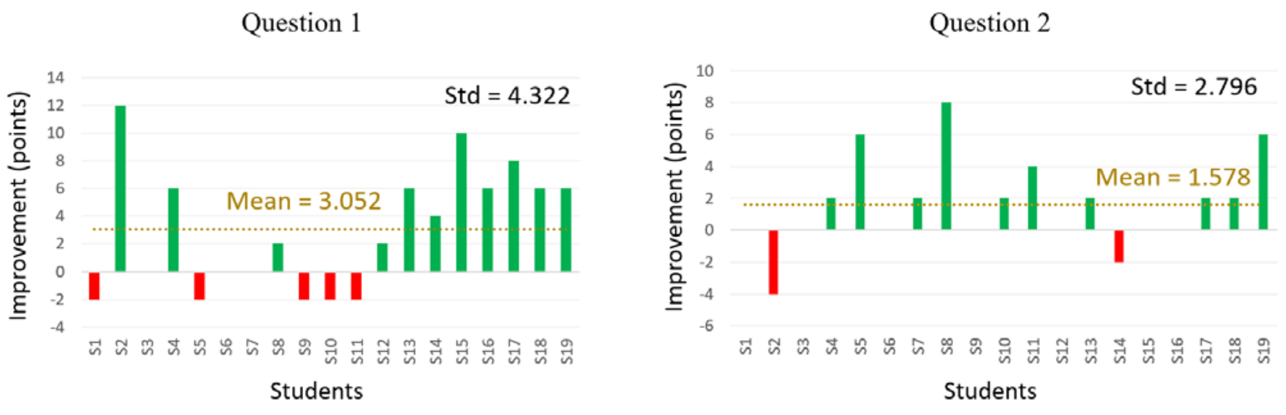


Figure 4: Performance improvement for questions 1 and 2.

Figure 5 shows charts showing students' performance improvement in the unplugged activities. Concerning the robot cleaner activity, all students (except S2) had an improved grade with a remarkable average improvement of 8.2 points. A paired *t*-test revealed that for a 94% level of confidence, there is an improvement in terms of students' performance (*p* = 0.056). Concerning the employee network activity, students' performance improved with an average improvement of 2.14 points apart from S11. A paired *t*-test revealed that for an 89% level of confidence, there is an improvement in terms of students' performance (*p* = 0.101). Concerning the chocolate distribution activity, students' performance improved with a remarkable average improvement of 10.14. A paired *t*-test showed that for a 99% level of confidence, there is a positive progress in terms of students' performance (*p* = 0.01).

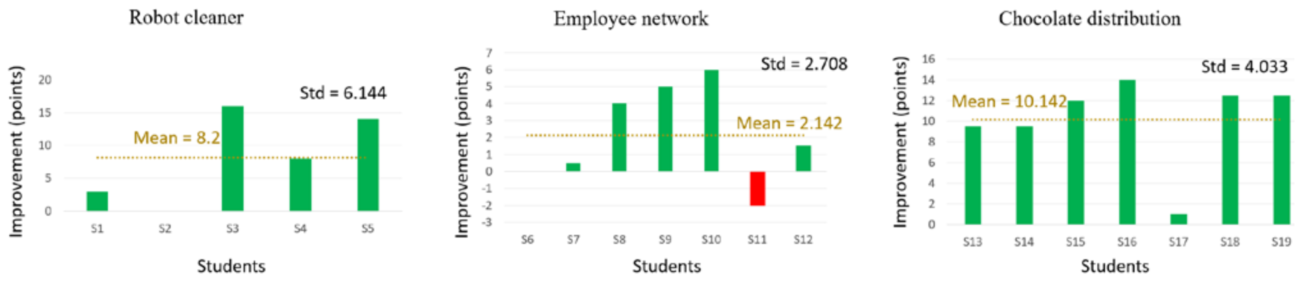


Figure 5: Performance improvement for the unplugged activities.

The results of the questionnaire are overwhelmingly encouraging, despite the teaching sessions being totally on-line. Seventeen out of 19 students did not think that the remote mode made the teaching activity challenging. Eighteen out of

19 students were engaged during the teaching activity, while 17 of them believed it was easy for them to learn the content. Eighteen of the students would like to see this teaching method being used to teach programming, and they think programming can be fun.

On the open-ended questions, students mentioned that they enjoyed the unplugged activities. For instance, one of the students mentioned, *I enjoyed the fact I can relate and associate a code to real life situation, which helps* [me] *to understand the code better and be aware of how it can be implemented in different situations*. Other students cited, *I enjoyed the way of interacting* and [1] *enjoyed learning by applying*.

When asked about what they did not enjoy about the study, the majority did not have any criticism, but one student stated, *The only issue was that we do not have the hands-on material available physically*. Unfortunately, providing physical hands-on materials was not possible in this study. In terms of suggestions for improvement, a common theme amongst many students was the need for more examples.

SURVEY

Prior to conducting this study, the authors carried out a survey to identify students' familiarity with recursion. The survey tested students' basic understanding of recursion, the applications of recursion, as well as reading and tracing recursive code. Twenty-six non-computer science undergraduate students were surveyed. Six of the students were second-year students, 17 were third-year, and the remaining three were fourth-year students. The surveyed students were a different sample from the students who were recruited for the study, though all of them were a similar population - non-computer science students majoring in technology-related fields such as multimedia and Web design.

The survey consisted of five questions. The first two questions were the two questions used in the theoretical evaluation in the study, the third question was a qualitative question asking students to define recursion, whereas the fourth and fifth questions were the same practical evaluation questions related to the robot cleaner and employee network activities. The authors used the same grading criteria specified earlier in the evaluation of the study. The students were told they could spend up to 40 minutes working on it. Nevertheless, most of the students filled it out in less than five minutes indicating that they were not sure how to tackle the questions (particularly the practical questions).

Figure 6 shows the histograms of the results of the survey quantitative questions. Regarding the first question, 50% of the students had a score below 2 (where the minimum points = -10, and the maximum points = 10), whilst 69.2% of the students had a score below 3 in the second question (minimum points = -6, maximum points = 6). Concerning the robot cleaner question, 57% of the students had a score below 2 (minimum points = 0, maximum points = 16) and 96% of the students had a score below 2 (minimum points = 0, maximum points = 6) in the employee network question. Concerning the qualitative question 3, only eight students (30.8%) were able to define recursion, while the remaining 18 (69.2%) had no clue what recursion was.

To conclude, about slightly less than a third of the students seemed to have an idea about recursion. However, most of the students had poor performance in both practical questions on recursion that required reading and tracing recursive code.

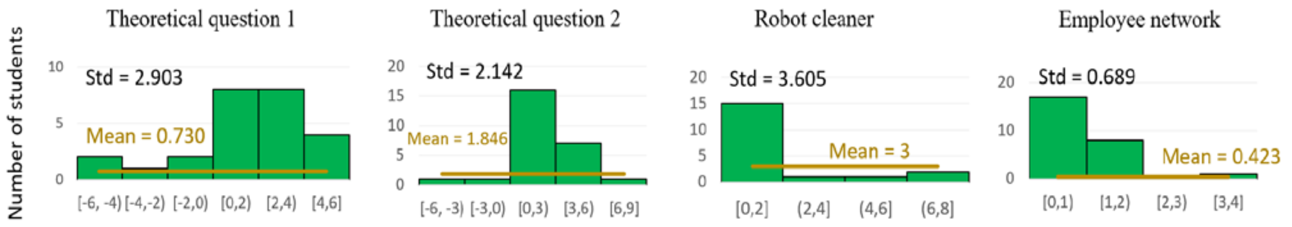


Figure 6: Histograms of the results of the survey questions.

CONCLUSIONS AND LIMITATIONS

In this article, the authors presented an innovative approach to teach recursive programming using unplugged activities in the form of engaging visual unplugged activities. Despite implementing this method in an on-line setting, the evaluation shows that students were engaged during the activities. Further, the students' performance on theoretical and practical questions improved after being engaged in the given three unplugged activities. As such, the authors invite other educators to conduct similar studies to evaluate the effect of unplugged activities on learning recursion in a higher education setting.

Despite the promising results, there is certainly room for improvement. For instance, more examples of unplugged activities need to be tested with a higher number of students. Further, it would be valuable to test the effect of remote teaching versus in-class teaching on the students' understanding of the same materials. The next step would be testing students' ability to write recursive code as opposed to reading and tracing it.

REFERENCES

1. Gal-Ezer, J. and Harel, D., What (else) should CS educators know? *Commun. ACM*, 41, **9**, 77 (1998).
2. Tong, J., Application of microlectures in teaching visual basic programming. *World Trans. on Engng. and Technol. Educ.*, 12, **3**, 554-558 (2014).
3. Kuang, T. and Zhu, S., A 3C3R teaching model applied to a C programming language course. *World Trans. on Engng. and Technol. Educ.*, 12, **4**, 610-613 (2014).
4. Pirolli, P.L. and Anderson, J.R., The role of learning from examples in the acquisition of recursive programming skills. *Canadian J. of Psychology/Revue Canadienne de Psychologie*, 39, **2**, 240-272 (1985).
5. Tessler, J., Beth, B. and Lin, C., Using cargo-bot to provide contextualized learning of recursion. *Proc. ICER '13*. Association for Computing Machinery, New York, NY, USA, 161-168 (2013).
6. Bau, D., Gray, J., Kelleher, C., Sheldon, J. and Turbak, F., Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (June 2017), 72-80 (2017)
7. Bell, T. and Vahrenhold, J., *CS unplugged - how is it used, and does it work*? In: Adventures between Lower Bounds and Higher Altitudes. Springer, Cham, 497-521 (2018).
8. Rodriguez, B., Rader, C., and Camp, T., Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment. *Proc. ITiCSE '16*. Association for Computing Machinery, New York, NY, USA, 95-100 (2016)
9. Gunion, K., Milford, T. and Stege, U., Curing recursion aversion. *SIGCSE Bull*. 41, **3**, 124-128 (2009).
10. Pollard S. and Forbes J., Hands-on labs without computers. *SIGCSE Bull*. 35, **1**, 296-300 (2003).
11. Feaster, Y., Segars, L., Wahba, S.K. and Hallstrom, J.O., Teaching CS unplugged in the high school (with limited success). *Proc. ITiCSE '11*. Association for Computing Machinery, New York, NY, USA, 248-252 (2011).
12. Thies, R. and Vahrenhold, J., Reflections on outreach programs in CS classes: learning objectives for *unplugged* activities. *Proc. SIGCSE '12*. Association for Computing Machinery, New York, NY, USA, 487-492 (2012).
13. Korhonen, A. and Vivitsou, M., Digital storytelling and group work. *ITiCSE 2019 - Proc. 2019 ACM Conf. on Innov. and Technol. in Computer Science Educ.* (2019).
14. Adobe Connect (2020), 10 August 2020, www.adobe.com/products/adobeconnect.html
15. Tufte, E.R. and Moeller, E.W., Visual explanations: Images and quantities, evidence and narrative. *Graphics Press Cheshire*, 36 (1997).
16. Wu, C., Dale, N.B. and Bethel, L.J., Conceptual models and cognitive learning styles in teaching recursion. *Proc. SIGCSE '98*. Association for Computing Machinery, New York, NY, USA, 292-296 (1998).
17. Coffman-Wolph, S., Innovative computer science activities for the classroom and outreach events. *Proc. ASEE's 123rd Annual*, New Orleans, LA, USA (2016).
18. Koffman, E.B. and Wolfgang, P.A.T., *Objects, Abstraction, Data Structures and Design Using C++*. John Wiley & Sons (2006).
19. Unplugged Activities Documentation (2021), 5 March 2021, https://www.dropbox.com/sh/c68504nskjm47e8/AADWgm1meVLMUuVEeGELvvkYa?dl=0
20. Wilson, D.B., Generating random spanning trees more quickly than the cover time. *Proc. Twenty-eighth Annual ACM Symp. on Theory of Computing*, Philadelphia: ACM, 296-303 (1996).