Turkish Journal of Electrical Engineering and Computer Sciences

Volume 29 | Number 5 Article 2

1-1-2021

Analyzing students' experience in programming with computational thinkingthrough competitive, physical, and tactile games: the quadrilateral methodapproach

M AHSAN HABIB

RAJA JAMILAH RAJA YUSOF

SITI SALWAH SALIM

ASMIZA ABDUL SANI

HAZRINA SOFIAN

See next page for additional authors

Recommended Citation

HABIB, M AHSAN; YUSOF, RAJA JAMILAH RAJA; SALIM, SITI SALWAH; SANI, ASMIZA ABDUL; SOFIAN, HAZRINA; and BAKAR, AISHAH ABU (2021) "Analyzing students' experience in programming with computational thinkingthrough competitive, physical, and tactile games: the quadrilateral methodapproach," *Turkish Journal of Electrical Engineering and Computer Sciences*: Vol. 29: No. 5, Article 2. https://doi.org/10.3906/elk-2010-73

Available at: https://journals.tubitak.gov.tr/elektrik/vol29/iss5/2

Analyzing students' experience in programming with computational thinkingthrough competitive, physical, and tactile games: the quadrilateral methodapproach

Authors
M AHSAN HABIB, RAJA JAMILAH RAJA YUSOF, SITI SALWAH SALIM, ASMIZA ABDUL SANI, HAZRINA SOFIAN, and AISHAH ABU BAKAR

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2280 – 2297

doi:10.3906/elk-2010-73

Analyzing students' experience in programming with computational thinking through competitive, physical, and tactile games: the quadrilateral method approach

Mohammad Ahsan HABIB[1], Raja-Jamilah RAJA-YUSOF[1,*], Siti Salwah SALIM[1], Asmiza Abdul SANI[1], Hazrina SOFIAN[1], Aishah ABU BAKAR[2]

[1]Department of Software Engineering, Faculty of Computer Science, University of Malaya, Kuala Lumpur, Malaysia

[2]Faculty of Civil Engineering Technology, Universiti Malaysia Pahang, Gambang, Malaysia

Received: 16.10.2020 • **Accepted/Published Online:** 12.03.2021 • **Final Version:** 23.09.2021

Abstract: The lack of computational thinking (CT) skills can be one of the reasons why students find themselves having difficulties in writing a good program. Therefore, understanding how CT skills can be developed is essential. This research explores how CT skills can be developed for programming through competitive, physical, and tactile games. The CT elements in this research focus on four major programming concepts, which are decomposition, pattern recognition, abstraction, and algorithmic thinking. We have conducted game activities through several algorithms that include sorting, swapping, and graph algorithms and analyzed how the game affects the student experience (SX) in understanding the CT concept in those algorithms. We have applied the quadrilateral method approach to the data collection and analysis. The data was obtained through observation, interview/survey based on six SX criteria (attention, engagement, awareness, satisfaction, confidence, and performance), and performances of the conducted game activities were compared. The results of the quadrilation of the data collected show a positive impact on the SX, highlight the effectiveness of the competitive, physical, and tactile game approach proposed in this research towards programming and CT skills development.

Key words: Programming, computational thinking, competitive, physical, tactile games, quadrilateral method

1. Introduction

Computer programming courses play a significant role in producing computer programmers who can craft state-of-the-art softwares. Teaching programming courses has always been a challenge to many lecturers and instructors [1–3]. Several related problems have been identified, such as difficulty in applying basic programming knowledge [4], difficulties in translating problems into solutions in the form of data and codes [1], and understanding existing codes and pseudocodes [5]. These problems may be due to unstructured thinking and the lack of computational thinking (CT) elements.

1.1. Computational thinking and its relation to programming

According to Tang et al. [6], computational thinking can be divided into two major definitions, (1) relates to a way of drawing from programming and computing concepts, and (2) relates to the set of components that

*Correspondence: rjry@um.edu.my

contribute to the competencies needed in both domain-specific knowledge and general problem solving skills. The first definition focuses on a theoretical framework that includes computational concepts, practices, and perspectives [7] as well as conceptual frameworks that include data, modeling, computational problem solving, and system thinking practices [8]. The second definition, which motivates this research, is looking into developing the CT skill that can be applied in any domain especially in programming. For example, the operational elements of CT such as abstraction, decomposition, algorithmic thinking, evaluation and generalization [9].

Programming can also be viewed as an activity that promotes CT skill development. A few practitioners view CT as an algorithmic process involving the formulation of the thinking process to be executed by the computer, and others relate CT to elements such as decomposition, abstraction, automation, pattern recognition, sequential execution, recursion, and parallelism [10]. Therefore, problems associated with learning to program are essentially problems related to CT [11].

1.1.1. How CT is related to programming

Some studies implicitly explain that CT is related to programming [12]. However, a few studies in the literature explicitly relate CT elements to the programming syntax. The framework discusses sequence, loops, events, parallelism, conditionals, operator, and data. The observed CT elements adopted by the programmers as strategies are as follows: incremental and iterative procedures, testing and debugging, reusing and remixing, abstracting, and modularizing.

Decomposition is simply a method of breaking a complex problem down into smaller pieces so that it can be conceived and managed easily [7, 10]. In programming, each keyword is used to address a small problem. Likewise, calculating values, assigning values to variables, or calling functions are statements each of which can be utilized to resolve a small problem and contribute to the solution of a bigger problem. Repetitively calling a certain sequence of steps is also a concept of decomposition [13]. Hence, it can be said that usages of keywords such as assignments, functions, and loops have direct correlations with the idea of decomposition.

Pattern recognition is a shared characteristic that occurs in each problem [14]. Once we have decomposed a complex problem into smaller parts, the next step is to look at the similarities that they share[1]. These similarities can be detected by selecting statements through comparison [15].

"Abstraction", on the other hand, refers to the process of generalizing important information while ignoring the irrelevant details. In programming, the act of calling a function invokes the formulation of an abstraction procedure [19]. In a simpler explanation, the process of abstraction is an application of many-to-one mapping [16]. Classes defined in the object-oriented (OO) programming languages are a form of abstraction; the background details or pieces of explanatory information are hidden to simplify the programming codes and to encourage reusability [13]. However, to be regarded as part of the CT element, the program codes involved should include more than one class.

Solving problems through step-by-step instructions that produce output correctly is algorithmic[2]. Table 1 shows a summary of the relation of CT elements to programming code. The programming attributes are categorized into three programming code elements, synthesized from those of Meyer, Hazzan & Kramer, Bishop [7, 13, 14, 16], and Franc[3], which are related to the CT element definitions used by Wing [17].

[1]Envato Pty Ltd (2021). The Basics of Computational Thinking [online]. Website https://webdesign.tutsplus.com/articles/the-basics-of-computational-thinking–cms-30172 [accessed 01 February 2021].
[2]Ibid.
[3]Ibid.

Table 1. Relation of CT elements to programming code.

Programming code attributes	Programming code elements categories	CT elements (Wing [17])	Author
Variable assignments Function defined Loop Sequential calls of attributes	Keywords (specific to a programming Language)	Decomposition	Meyer [13]), Brennan & Resnick's [7]
Function called Classes (in OO) Conditional statements (If- else, switch) Comparison	Syntax of codes	Abstraction Pattern recognition	Franc[4], Meyer [13], Hazzan & Kramer [16], Bishop [14]
Correctness	Program output	Algorithmic	Franc[5]

1.2. Programming and CT through competitive, physical, and tactile games

A search through the literature yielded eight interesting teaching approaches to programming that incorporate CT and games that are used as a mechanism to deliver knowledge and skills. Table 2 summarizes the results of the literature search. Nardelli & Ventre [18] incorporated multiple CT elements through many stages of problem-solving tasks in the form of video games and cartoon characters. Such an effort could be seen through the work of Dantas, Lopes & Amaral, Weintrop & Wilensky, Hyman et al., Esper et al., Brady et al., Brennan & Resnick [19–23].

Table 2. Literature on game-based programming learning tools with CT elements incorporated.

Researchers	Brief description	Incorporated CT elements	Learning Tools
Nardelli & Ventre [18]	Tutorials are based on video games and cartoon characters	To define solutions involving several CT components	Hour of Code from http://code.org
Brennan & Resnick [7]	Game-based programming through story and data simulation	Iteration, parallelism, designing, debugging, remixing	Scratch
Dantas, et al. [20]	Learning programming through serious games	Logical reasoning, algorithmic thinking in solving problems	Programming Life
Weintrop [23]	Design and implement codes for their robot to defeat opponents	Computationally expressing ideas, algorithmic thinking, and debugging	Robo games
Hyman, et al. [22]	Collaborative historical puzzles emphasizing computational thinking	Pattern recognition, decomposition, algorithmic thinking, and data representation	The Tessera
Brady et al.[21]	Open-hardware programs to explore technology that enables the Internet	Abstract and quantitative reasoning	Wearing the Web-in
Esper, et al. [19]	A text-based or syntax-based programming game environment	Decomposition, pattern recognition, debugging	CodeSpell
Zhao & Shute [24]	Web-based video games that use Blockly as the code editor.	Process of abstraction, reusable code blocks	Penguin Go

Extending a purely computer-based environment to the more tactile robotic, Arduino, Raspberry Pi and sensor-based programming sessions following the work of the visionary Seymour Papert [25] and those of the

more recent Kubitza & Schmidt [26] had positive effects on teaching programming. Corral et al. [27] as well as and Melcer & Isbister, [28] through a tangible user interface (TUI) such as Sifteo cubes, wooden blocks programming also found positive effects. The use of games found in all the literature implies that for students to be engaged in programming subjects, there is a need to create an environment that is competitive, and it seems that tangible items that can be physically touched appeal to students [29]. Games create an environment conducive to learning due to several reasons [30]. They provide a platform for multisensory active as well as problem-based learning experience [31]. At different levels, the players can evaluate their skills and gradually build up confidence within their environment as games involve social interactions.

From a psychological perspective, Csikszentmihalyi [32] explains that games create engagement and enjoyment through the concept of flow. Flow is an optimal state of mind in which a person becomes fully absorbed in an activity and it is conducive to productivity. The state of mind and body governed by rules, goals, and feedback promotes this optimal experience.

According to Csikszentmihalyi [32], it is not what humans do that makes it fun, but how they do it. The relation between flow and enjoyment depends on whether the flow-producing activity is complex, leads to new challenges, and contributes to individual/group growth. In the context of this study, computer programming is not itself enjoyable to some students. However, if the learning process is carried out in episodes of flow such as a competitive environment through games, learning becomes rewarding and worth doing. Competitiveness increases the adrenaline level, especially if it relates to the outcome of having a winner-loser situation. Any physical and tactile movement would amplify the adrenaline experience and increase awareness, which adds to the increase in performance, engagement, and enjoyment of the game in play.

1.3. Exploratory model in teaching programming

Figure 1 shows the model of our approach in teaching programming courses with computational thinking designed based on the literature mentioned in the previous section. It focuses on competitive, physical, and tactile games to promote CT as an alternative solution to the problem of teaching and learning programming-related subjects. It is an active learning approach that leads to the state of optimal flow to create a conducive learning experience.

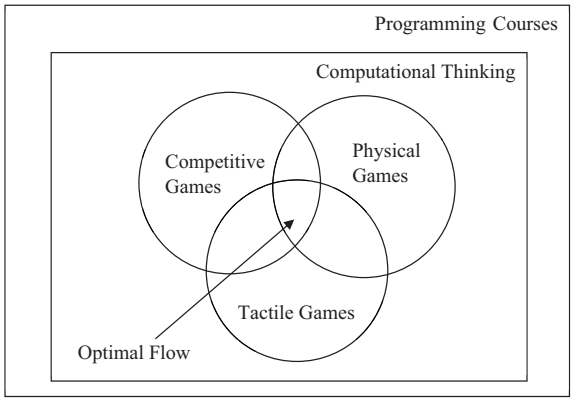

Figure 1. Competitive physical and tactile games for programming-related subjects.

Our exploratory study analyzes students' experience (SX) in programming with CT through competitive, physical, and tactile games. We attempted a quadrilateral analytical research approach in our data gathering

and analysis based on methods adopted to teach and learn engineering-related subjects. The following sections cover the research methodology, results, discussion, and conclusion.

2. Research methodology: the quadrilateral approach

Our quadrilateral research approach is a mixture of qualitative and quantitative methods as shown in Figure 2. It is based on research studies done in the context of teaching and learning in engineering-related subjects. First, it involves the gathering of data through observation of SX in class as done by Kennedy&Kraemer [33], interviewing students as done by Peterson et al. [34]), comparing students' academic performance from the previous session (as done by Berenguel et al. [35]) and Guzelis [36], and comparing students' academic performance within this study session (as done by Akaslan and Law [37]). Based on the problems and motivation, our research questions focus on student experience and performance as follows:
RQ1: What are the students' experiences in relation to the competitive physical and tactile game?

1. How did the competitive physical games catch the students' attention?
2. Were the students engaged in the activity?
3. Were the students aware that competitive physical games help them understand computational thinking concepts more effectively?
4. Did the students find satisfaction after playing the games?
5. Do the students show confidence in handling CT concepts?

RQ2: How are the performances of programming in relation to the competitive physical and tactile game?

1. Does the event enhance their academic performance?
2. Is there any difference in the students' academic performance before and after the competitive physical and tactile game session?
3. Is there any difference in the students' academic performance comparing previous and current sessions?

As described above, our study also captures the SX (to answer RQ1&2) through observation, interviews, and surveys as outlined in Section 2.1. A quantitative study was conducted to answer RQ2, i.e. to find out if there are any significant differences in the performance of the two groups of students in answering the programming questions: students who went through the competitive and tactile games, and those who did not. This is outlined in Section 2.2.

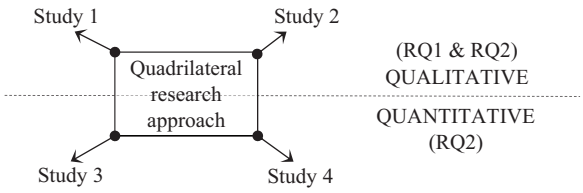

Figure 2. The quadrilateral research approach.

2.1. Study setting and participants: interviews and surveys (study 1), observations (study 2)

In study 1, SX was captured through six parameters: attention, engagement, awareness, satisfaction, confidence, and performance. Through the qualitative study, the authors observed behaviors of the students who participated in the aforementioned games in class. Written consents from the students were obtained for the study conducted in the chosen courses, the procedures of which include observations, video recordings, and interviews. The students were given the understanding that the data obtained would be kept confidential, they can withdraw anytime, and their identities remain anonymous. total of 193 students consisting of pre-university students and first-to-fourth year undergraduate students. However, in study 2, the open-ended survey questions were asked to 18 randomly chosen and willing participants, with whom the semi-structured interview sessions were conducted. Their answers and opinions to the questions regarding their experiences during the lecture sessions were recorded. Our study was part of the learning process conducted during the lecture time; the topics were part of the course learning outcomes. The study entailed observing the students during and after the teaching of specific topics.

The observations, interview, and surveys were conducted among the students taking the courses of Analysis Algorithms, Data Structures, Programming I and Programming workshop. The details are as in Table 3. As our studies are explorative ones, we adopted a naturalistic observational research design even to advance topics of programming subjects. The reason for exploring these studies for both fundamental and advanced programming courses is due to our teaching experience observing students' difficulties in fundamental as well as advanced classes. Table 3 also shows a total of 11 videos recorded, with a total duration of 6 min and 5 s during the algorithm, programming, and data structures classes and the lessons that were related to the sorting and graph topics. Many photographs were also taken during the sessions. The interviews were conducted based on the research questions on RQ1 and RQ2a (i.e. the six SX parameters). Table 4 shows the RQ1 questions posed to the participants.

Table 3. Subjects and participants involved in the studies.

Subjects	Semester Year	Student Year	Students involved	Number of videos recorded	Duration of videos (seconds)	Topic of focus
Programming I	1, 2016/2017	1^{st}	30	5	8, 8, 17, 21, 22	Sorting
Data structure	1, 2016/2017	2^{nd}	21	3	22, 23, 30	Sorting
Analysis of Algorithms	2, 2016/2017 1, 2016/2017	2^{nd} 4^{th}	44 13	3 -	45, 65, 95 -	Sorting, graph Sorting
Programming workshop	2018&2019	1^{st} year & pre-university	85	-	-	Sorting
TOTAL			193	11	6 min 5 s	

2.1.1. Research materials and procedures

Games and materials

The authors were inspired by the unplugged computer science activities [38] and CT video materials from Code.org. Eight CT game activities were conducted, namely, swapping, bubble sort, quick sort, merge sort, selection sort, insertion sort, radix sort, and the graph algorithm. Four major components of the CT elements

Table 4. The relation between research questions, themes, and subthemes.

Relation to research Question	Theme	Subthemes
1. How did the competitive physical games caught students' attention?	Attention	Helpful knowledge, clear confusion, deliver the intended notion and high academic value
2. Were the students engaged in the activity?	Engagement	Competitiveness and the needed lessons
3. Were the students aware that the competitive physical games help them to understand computational thinking concepts more effectively?	Awareness	Important takeaways and learning new things
4. Did the students obtain satisfaction after playing the games?	Satisfaction	Interest in games and good academic content.
5. Did the students show confidence in handling CT concepts?	Confidence	Confidence to do programming
6. Did the event enhance their academic performance?	Performance	Academically relevant and helpful, academic performance and improvement.

(i.e. abstraction, decomposition, pattern recognition, and algorithm) were involved in the games. See Table 5 for the detailed description of the games.

There were two types of games where materials used were slightly different. Type-I games required a single player and were played using small cards. Type-II games referred to those games which require students to form groups consisting of 5–15 members. The materials used were A4 size coloured cards and alphabet mats (A–Z). At the end of the game, winners and all students received sweets and/or chocolates as prizes.

Game procedure

Figure 3 shows the descriptions of the activities that were carried out. The game sessions were conducted in four parts (A, B, C, and D). The first part-A is the verbal explanation, the second part-B is a short demonstration of the competitive, physical, and tactile games to draw the students' attention, the third part-C is the beginning of the actual game session among the students to capture students' engagement with the games. The last part-D is to identify the winners of the games based on the students' performance in the games. The winners were announced, and prizes were given to them.

2.1.2. Verification and validation

A set of criteria was used to verify and validate the qualitative research conducted. Verification was achieved through credibility and conformability, while validation was achieved through transferability. The research credibility fully supported the data collected from research studies 1 and 2. This method has the advantage of deriving results from participants' reflections and researchers' observations. The dual perspective served as a check and balance on the interpretation of the data. The aspects of conformability were compiled through developing the research questions guided by the motivation and related work from the studies in the literature that covered the subjects of teaching programming and computational thinking. The steps of the games created listed in Table 5 were confirmed for its correctness by a lecturer that teaches programming.

The transferability aspects were achieved in two methods because the feedback records were in two types; the recorded audio and the written survey forms. The recorded audios were transcribed into written English

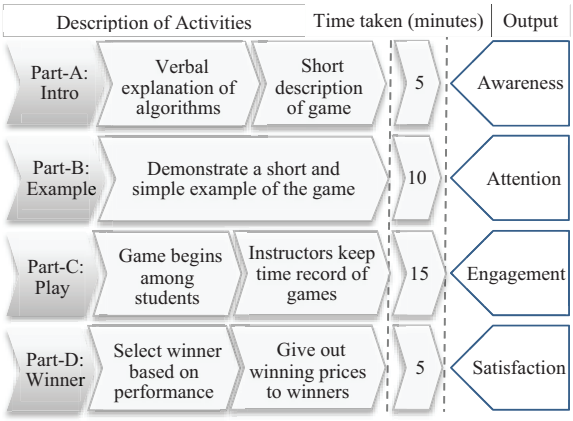

Figure 3. The activities carried out based on the game procedure.

language and transferred to an excel sheet. The responses in the survey form were transferred in the same excel sheet. Later on, the information in the excel sheet was validated by the participants.

2.1.3. Qualitative data analysis

We used the thematic approach to analyze the data. It was conducted as soon as all the interview recordings were transcribed. The theme was based on the six SX parameters; attention, awareness, engagement, confidence, satisfaction, and performance. Later, the responses were grouped into subtheme categories aligned with the main themes. See Table 4 for the relation between the research questions, themes, and subthemes. The participants' responses based on each subtheme were counted, and the percentages of the subtheme responses were tabulated.

2.2. Grade performance

There were two ways to observe performance. Study 3 and study 4 were our approaches to explore the performance of both fundamental and more advanced programming courses after conducting the competitive, physical, and tactile games in answering RQ2b and RQ2c. Study 3 explores the performance of more advanced programming classes while study 4 explores the performance of fundamental classes.

2.2.1. Study 3

The grading of performances was conducted by comparing the examination results of two groups of students. The 1^{st} group consists of 8 students from the Analysis of Algorithm class in 2016; the second group is comprised of 12 students from the Analysis of Algorithm class in 2017. The year-2017 group was exposed to competitive physical games, while the year-2016 group was not. Special permission was obtained to analyze the answer scripts of the participating students. The topics considered in the study were sorting and graphs. The students' marks remained anonymous so that the results could be presented without knowing the identities. The two sets of examination questions from the two different years were set by the same lecturer, and the difficulty levels were the same. The students' marks (maximum of points) were identified and recorded. A total of 20 students' marks were analyzed for the consecutive exams.

2.2.2. Study 4

The session consists of three parts. The first is the learning session on "Programming Basics" in which a pretest question was distributed. The second part was the game session, and the third part is the answering posttest question session. The session was arranged in which 33 first-year students (admission of the year 2018) and 52 pre-university students participated (2019 session). Most of them had little experience in programming. The students were divided into two groups: one group played the game (GG- 13 students from the first year, and 31 students from the pre-university) and the other group only had to listen to the programming lecture (LG- 20 students from the first year, and 21 students from the pre-university). LG was asked to skip part two of the session, and the members proceeded to part three. In part two of the session, participants from GG had partaken in three gaming activities– bubble sort, radix sort, and selection sort. The information of all the abovementioned gaming activities can be found in Table 5.

Table 5. Short description of games.

No	Game	Computational thinking element involved
1	Swap	Algorithmic: requires three sequential steps
2	Quick sort	Decomposition: divide the problem into subproblems (half the input to be processed) Pattern recognition: compare numbers bigger/smaller Algorithmic: repeat comparison steps Abstraction: if a procedure works for the smallest input problem, the procedure should work for the whole set of input
3	Merge sort	Decomposition: divide the problem into subproblems (half the input to be processed) Pattern recognition: compare numbers bigger/smaller Algorithmic: repeat division and comparison steps Abstraction: if a procedure works for the smallest input problem, the procedure should work for the whole set of input
4	Selection sort	Decomposition: repeat unit steps Pattern recognition: Find the minimum Algorithmic: repeat the step of finding the minimum number
5	Insertion sort	Decomposition: repeat unit steps Pattern recognition: Find the minimum Algorithmic: Repeat the insertion step
6	Radix sort	Decomposition: repeat unit steps (1^{st}, 2^{nd}, 3^{rd} place values) Pattern recognition: find matching number positions Algorithmic: repeat sequential actions
7	Graph Theory	Algorithmic: the names that are chosen one after another should be within a short distance Abstraction: if a procedure works for the smallest input problem, the procedure should work for the whole set of input
8	Bubble sort	Decomposition: repeat unit steps Pattern recognition: Find the minimum/maximum Algorithmic: repeat the comparison steps

3. Results

The results are divided into two parts: Section 3.1 presents the observation results together with the interview and survey results and Section 3.2 presents the performance academic performance.

3.1. Results of study 1 (observation) and 2 (interviews and survey)

3.1.1. Study 1: observations

Based on the observations of the activities conducted and other information gathered, the students appeared to be excited when they came to know that they would be participating in an educational game. The observations are reported in four parts; awareness, attention, engagement, and satisfaction of the year-2 and year-4 students when they were learning the subjects of Programming, Data Structure, and Algorithms. On the whole, the observations are consistently the same in all of the classes. Regarding attention, while playing the game, they looked very attentive and exhibited a genuine interest in accomplishing the goals to the best of their ability. Regarding engagement, for individual games, the participating students were usually silent and seen to focus on the steps to complete the games. Sometimes, there were short dialogues between participants who were discussing game strategies or giving a few game tips. Regarding awareness, the students were aware of the topic introduced. Sometimes, they asked questions regarding the algorithm and brief answers were given by the lecturers/instructors conducting the session. Regarding confidence, they were serious to complete the game and put in great effort to score points. They tried their best to execute the algorithm steps together in the fastest time possible so that they could win the game. Finally, regarding satisfaction, they seemed to enjoy themselves thoroughly in the games. Everyone seemed happy and satisfied when they found out that everybody was given sweets as gifts for participating in the game.

3.1.2. Study 2: interview and survey results

Table 6 shows the overall results of the interviews, and they were reported according to the sequence of the research questions, themes, and subthemes. The results in this section are referred to from participant number 1 to number 18 which are represented as P#1 to P#18.

Attention

The event captured the students' attention, and they benefited in these aspects,

The first aspect is that it clears confusion. 53.33% of the participants stated that the gaming session cleared their confusion. P#17 wrote, "It cleared my programming confusion". P#6 said, "The game helped me gain a better understanding of the algorithms. However, better understanding does not necessarily mean that one can code easily because some programming tricks might be required for some implementations. This needs to be addressed."

The second aspect is that it delivers the intended notion. 82.50% of the students said that they could easily understand the academic material through the games and that the sessions were enjoyable. P#17 wrote, "I found that the game illustrates how the program works. Moreover, we are having fun."

The last aspect is its high academic value. 69.38% of the participants opined that games had the potential to be used in the academic field. P#13 commented, "I was able to learn more in this session than I would by studying slides."

Engagement

The engagement was generated through competitiveness and lesson needs. In terms of competitiveness, the game activities were designed to be intense and interesting to lit up the students' enthusiasm, engage them deeply, and encourage healthy competition. 83.13% of the students said that the games were really competitive.

P#4 mentioned that, "I'm very competitive"; while P#17 said, "My level of competitiveness was quite high because it appeared that everyone wanted to win the game."

As for lesson needs, students were not just playing; they were giving us suggestions when requested. About 86.88% of the students gave positive suggestions to increase the reach of this type of game sessions. For instance, P#18 stated that: "1) the progress of the game can be improved by increasing its difficulty and complexity and 2) good effort." P#2 observed that the physical games had the potential to be more interesting and enjoyable. In addition, P#2 said that such games should be created "for topics in which students are often weak...". P#13 stated that "more lessons should be conducted this way" and P#17 stated that this activity should be continued to other students in other years for different subjects such as data structures so that students can learn and improve their understanding through the games.

Awareness

The awareness was interpreted through important takeaways and learning new things. In terms of important takeaways, 61.11% of the participants said they learned something and had a few takeaways from the gaming sessions. P#2 stated that "The event was short and simple, yet it was quite effective...". P#6 wrote, "Clearer understanding of how the algorithm works." Similarly, P#3 said that "sorting" games involved developing a step-by-step solution," P#7 feel that "the merge sort helped in creating steps to solve each part of a problem." P#1 commented, "I do notice that step-by-step solutions are developed from the selection sort" and participant#6 "...we noticed that we focus on the important information.. ignoring the irrelevant details ..while playing the game."

Regarding learning new things, 81.88% of the students reported that they learned something new compared to the conventional classes before. P#15 stated that, "I learned new programming skills and algorithms." P#4 wrote that "I think everything I learnt is new".

Satisfaction

The satisfaction was found through interest in games and good academic content. In terms of interest in games, 71.25% of the students reported that the games were "really interesting and fun, especially, it is cool for a class to conduct this type of game," said P#17. P#4 said that she "enjoyed it very much".

Regarding good academic content, all the games were designed based on academic content. About 67.50% of the students stated that they were satisfied with the content. For example, P#5 said that he was satisfied with the session and its content. Students also gave suggestions when requested. About 86.88% gave positive suggestions to increase the reach of the game. For instance, P#18 stated: "1) the progress of the game can be improved by increasing its difficulty and complexity; 2) a good effort since it was observed that the physical games had the potential" to be more interesting and enjoyable for students...", quoted by P#2. In addition, they believed that such games should be created "for topics in which students are often weak..."(P#2) They stated that more lessons should be conducted this way, and "continued to other students in other years for different subjects. So that students who could not understand can learn and improve their understanding through the games...", stated by P#17.

Confidence

Having the confidence to solve a problem in programming is among the biggest challenges faced by students. 56.15% of the participants thought that the games boosted their confidence in handling programming assignments. P#2 and P#1 wrote "A bit clearer" and "Better" confidence.

Table 6. Thematic analysis with students' feedback.

Relation to research question	Theme	Subthemes	%
1. How did the competitive physical games catch your attention?	Attention	Helpful knowledge	62.50%
		Clear confusion	53.33%
		Deliver the intended notion	82.50%
		High academic value	69.38%
2. Were the students engaged in the activity?	Engagement	Competitiveness	83.13%
		The lessons need	86.88%
3. Were the students aware that the competitive physical games help them understand computational thinking concepts more effectively?	Awareness	Important takeaways	61.11%
		Learning new things	81.88%
4. Did the student find satisfaction after playing the games?	Satisfaction	Interest in games	71.25%
		Good academic content	67.50%
5. Did the students show confidence in handling CT concepts?	Confidence	Confidence to do programming	56.15%
6. Did the event enhance their academic performance?	Performance	Academically relevant and helpful	63.33%
		Academic performance	61.88%
		Improves thinking	66.80%

Performance

The performance was interpreted through the following aspects. The first aspect is academic relevance and helpfulness. All games were designed with learning and CT in mind. About 63.33% of the students mentioned that it was "easy to understand the concepts and to write codes", P#10 quoted.

The second aspect is academic performance. 61.88% of the participants opined that their academic performance of the relevant topics improved. P#7 said, "It helped me understand... in an easier way."

The last aspect is improving thinking. 66.88% of the participant agreed that, as a result of playing the games, their perspectives on the subject changed. Statistically, their results were better as well. P#11 wrote, "It helped me understand the theory even more easily." P#6 stated, "It does help me visualize problems in a much clearer form rather than just seeing them as words and numbers."

3.2. Result of study 3 and 4 (academic performance)

Apart from gauging the students' perception of their academic performance, it is also seen through the analysis of the marks scored in the pretest conducted before and the posttest conducted after the game activities to the gaming group (GG). A normal lecture was conducted on the control group (LG).

3.2.1. Study 3

Table 7 shows the standard deviations (SD) of the marks of both groups, which are, 1.72 and 1.71, but the means and medians of the year-2017 group are higher than those of the year-2016 group. In Figure 4, the results of our analysis of the students' academic records show that the average mark in group B for graph topics is about 60%, whereas the average mark for group A is about 80%, representing an increase of 20% in the average mark. For the sorting codes, the average mark is 40% for group B, and 80% for group A, representing an

increase of almost 40%.

Table 7. Comparative observation of SD, mean, and median marks of two groups.

	Group 2016	Group 2017
Standard deviation	1.72	1.71
Mean mark	5.55	8.13
Median mark	5.67	8.00

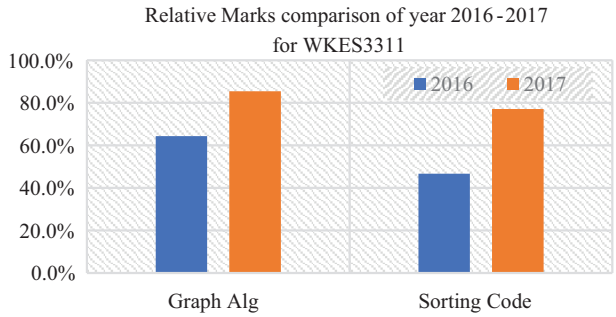

Figure 4. Average mark in the final Exams for the 2016 and 2017 student groups (study 1).

3.2.2. Study 4

Study 4 employs -*diff*- package [39] to estimate difference in differences (DiD) using STATA software to analyze the data, shown in Table 8, of the control group (i.e. LG) with 41 participants and the treated group (i.e. GG) with 44 participants. Total number of observations is 170 for 85 students in pretest and posttest, where $R^2 = 0.15$. The difference of mean score for pretest between the LG control group and GG treated group is –1.109 and the DiD estimation with p-value 0.017 ($p < 0.05$) indicates a significant difference at 5% level. And the difference of mean score for posttest between the LG control group and GG treated group is 2.223 and the DiD estimation with p-value = 0.000 ($p < 0.01$) indicates a significant difference at 1% level.

The last row, DiD treatment-effects estimand, is implying an increase in the score of students by 3.332. The p-value 0.000 (p<0.01), standard error 0.650, indicates the DiD estimand is significant at the 1% level. Figure 5 illustrates the treatment-effect in using lines. The effect of gaming workshops on GG is that the mean marks of GG increased by 2.780 from 1.500 (projected) to 4.280 (actual).

4. Discussion and conclusion

Observation results show that the students enjoyed competitive physical and tactile games which could be explained with the optimal flow theory by Csikszentmihalyi [32] and consistent with other previous findings [7, 18–23, 32]. They were aware, attentive, engaged, and satisfied with the game sessions. The class was lively and students were full of expressions during the game sessions. Learning in an experiential setting (translation of cognition to psychomotor) [40] has shown positive effects; this is the principle undergirding the present study. Attention was observed through the student's commitment to complete the competitive physical and tactile activities similar to many other studies on active learning in programming [41]. Engagement was detected through their views to have similar activities that cover different topics. In terms of CT skills, students

Table 8. Difference in differences estimation results.

Outcome var.	Marks	S. Err.	|t|	P> |t|
Before				
Control	4.098			
Treated	2.989			
Diff (T-C)	–1.109	0.460	–2.41	0.017**
After				
Control	2.057			
Treated	4.280			
Diff (T-C)	2.223	0.460	4.83	0.000***
Diff-in-Diff	3.332	0.650	5.12	0.000***

R-square: 0.24

* Means and Standard Errors are estimated by linear regression

**Inference: *** p<0.01; ** p<0.05; * p<0.1

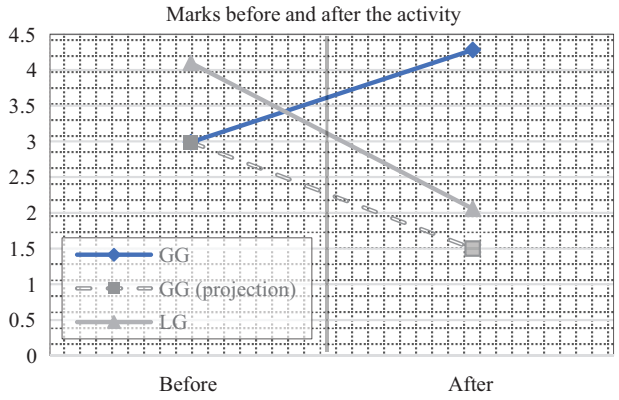

Figure 5. Mean marks of participants before and after the gaming workshop.

were aware of the decomposition, abstraction, pattern recognition, and algorithmic process that they have to go through in solving the problems posed to them. However, it could be more effective if the lecturers or instructors relate CT elements to the program codes such as indicated in the work of [7, 13, 14, 16].

Students were delighted and motivated when they were all rewarded for their participation and commitment. Similar observations regarding reward were made by other researchers who had employed competitive games as learning tools in the classrooms [42]. Confidence was also observed and extracted from the student opinion as they thought that the games boosted their CT skills.

In terms of their academic performance, study 3 and study 4 imply that the students that participated in the game had better academic performance than those who did not. In study 3, the mean scores of students who participated in the game in 2017 were higher than those in the year 2016. While in study 4, the DiD results show a significant difference in the scores for the GG group with the mean difference of 3.33. It was seen that the actual scores are higher than the projected marks. These results imply that when students go through competitive, physical, and tactile gaming activities, they scored higher. It may be due to the fact that they were presented with tangible items that can be physically touched, activating their multisensory system

which enhances learning experiences. These are consistent with other studies in this area such as in robotics [25, 26, 28] as well as digital game-based learning [27, 29, 30].

However, this is in contrast to the findings of Hanus & Fox [43] and the reason may be the difference in the approach. Hanus & Fox gamified the process of learning such as getting reward for good participation in class, turning in the assignments early and others. However, it was not clear if the video game played in Hanus & Fox's study was related to the topics student learned. Therefore, gamifying the learning process may have to be in relation to the topics learned and not merely for other mundane tasks related to learning.

Most students admitted that the game helped them understand the concepts and cleared their mind about the problems. Our findings can be used to support efforts related to the use of active learning activities through competitive, physical, and tactile approach in the classroom to increase the students' academic performance. It is significant to recognize that computer programming curriculum must be transformed from the traditional delivery mode to active learning mode if challenges related to its delivery due to unstructured thinking and the lack of CT elements are to be overcome. Previous studies [25–30] showed that there are these pieces of evidences of effectiveness in teaching and learning when conducted in form of active learning through physical and tangible tools such as robots, raspberry pi and Arduino as well as in nontangible game-based approach. As our studies showed, a less costly approach using cheaper materials could also outperform the traditional lecture approach. Therefore, in countries and places where the luxury of having costly material in class could be replaced with cheaper materials to conduct active learning activities even in the higher educational settings.

Our quadrilateral method approach analyzing data from four angles gives richer datasets for the quadrilation of the results. This approach could be used in the context of reporting teaching and learning experiences of students in computer science and engineering degrees. Our contribution in defining and relating computational thinking elements to programming codes could be useful in translating the theoretical to practical applications in terms of designing games and teaching and training modules. The importance of the study is to address almost unsolved and recurring problems of teaching and learning programming. Programming skills are essential for the future generation because of technological advancement in all fields involving interaction with a computerized environment. Nontechnical users within the scientific and nonscientific communities are starting to learn how to program. For technological advancement to grow faster especially in developing countries, alternative and effective methods should be available in teaching programming.

5. Limitation and future work

Our work was reported based on the available dataset obtained from the class we conducted the game session on topics related to sorting and graphs algorithm. We could not include results related to other topics such as arrays, binary trees, and amortized algorithms. In these classes, similar observations of SX were seen. Data was not sufficient to report positive or negative outcomes.

Further work could be done in the areas such as modeling a problem-based approach in solving algorithmic problems related to programming. Teaching modules could be created based on a problem-solving approach instead of syntax-based ones since the fundamentals of programming teaching modules should address the issue of how to go about solving problems instead of only teaching syntax of the programming language.

Acknowledgments

We would like to thank the University of Malaya for providing the UM-LiTeR grant (RU002Y-2016). Also, this project has been funded by the European Commission under the Erasmus plus (submission no: 586297-EPP-1-

2017-1- EL-EPPKA2-CBHE-JP). This publication reflects the views of only the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein. This project is also partnered with the Department of Software Engineering, Faculty of Computer Science and Information Technology, the University of Malaya with the grant no. IF024-2018. We would also like to express our appreciation to Siti Sarah Azmi for helping out with the transcription of the interviews.

References

[1] Ortiz OO, Franco JAP, Garau PMA, Martin RH. Innovative mobile robot method: improving the learning of programming languages in engineering degrees. IEEE Transactions on Education 2017. doi: 10.1109/TE.2016.2608779

[2] Kunkle WM, Allen RB. The impact of different teaching approaches and languages on student learning of introductory programming concepts. ACM Transactions on computing Education 2016; 16 (1): 1-26. doi: 10.1145/2785807

[3] Hegazi MO, Alhawarat M. The challenges and the opportunities of teaching the introductory computer programming course: case study. In: 2015 Fifth International Conference on e-Learning (econf) 2015; pp. 324-330. doi: 10.1109/ECONF.2015.61

[4] Ab Hamid SH. Solutions to teaching object-oriented programming. WSEAS Transactions on Communications 2004; 3 (1): 99-104.

[5] Marcolino AS, Barbosa E. A survey on problems related to the teaching of programming in Brazilian educational institutions. In: 2017 IEEE Frontiers in Education Conference (FIE) 2017; pp. 1-9. doi: 10.1109/FIE.2017.8190495

[6] Tang X, Yin Y, Lin Q, Hadad R, Zhai X. Assessing computational thinking: a systematic review of empirical studies. Computers & Education 2020; 148: 103798.

[7] Brennan K, Resnick M. New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 annual meeting of the American educational research association; Vancouver, Canada 2012; 1:25. doi: 10.1.1.296.6602

[8] Weintrop D, Beheshti E, Horn M, Orton K, Jona K et al. Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology 2016; 25 (1): 127-47. doi: 10.1007/s10956-015-9581-5

[9] Selby C, Woollard J. Computational thinking: the developing definition. In: ITiCSE Conference, University of Kent, Canterbury, England; 2013.

[10] Barr V, Stephenson C. Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? Acm Inroads 2011; 2 (1): 48-54. doi: 10.1145/1929887.1929905

[11] Li Y. Teaching programming based on computational thinking. In: 2016 IEEE Frontiers in Education Conference (FIE) 2016; pp. 1-7. doi: 10.1109/FIE.2016.7757408

[12] Looi CK, How ML, Longkai W, Seow P, Liu L. Analysis of linkages between an unplugged activity and the development of computational thinking. Computer Science Education 2018; 28 (3): 255-79. doi: 10.1080/08993408.2018.1533297

[13] Meyer B. Object-oriented software construction. Prentice hall Englewood Cliffs, 1997; 2: 331-410.

[14] Bishop J. Language features meet design patterns: raising the abstraction bar. In: Proceedings of the 2nd International Workshop on the Role of Abstraction in Software Engineering 2008; pp. 1-7. doi: 10.1145/1370164.1370166

[15] Riley DD, Hunt KA. Computational thinking for the modern problem solver. CRC press 2014. doi: 10.1201/b16688

[16] Hazzan O, Kramer J. The role of abstraction in software engineering. In: Companion of the 30th International Conference on Software Engineering 2008; pp. 1045-1046. doi: 10.1145/1370175.1370239

[17] Wing JM. Computational thinking. Communications of the ACM 2006; 49 (3): 33-5. doi: 0001-0782/06/0300

[18] Nardelli E, Ventre G. Introducing computational thinking in Italian schools: a first report on "programma il futuro" project. In: 9th International Technology, Education and Development Conference 2015; pp. 7414-7421.

[19] Esper S, Foster SR, Griswold WG, Herrera C, Snyder W. CodeSpells: bridging educational language features with industry-standard languages. In: Proceedings of the 14th Koli Calling International Conference on Computing Education Research 2014; pp. 05-14.

[20] Dantas TF, Lopes PP, do Amaral EM. Programming life: gamification applied to the teaching of algorithms and programming through a serious game. In: Handbook of Research on Immersive Digital Games in Educational Environments. Hershey, PA, USA: IGI Global, 2019, pp. 486-523.

[21] Brady C, Orton K, Weintrop D, Anton G, Rodriguez S et al. All roads lead to computing: Making, participatory simulations, and social computing as pathways to computer science. In: IEEE Transactions on Education 2016; 60(1): 59-66. doi: 10.1109/TE.2016.2622680

[22] Hyman C, Shrout A, Kaczmarek-Frew K, Green H, Frazier N et al. Decolonizing methodologies: recovery and access amidst the ruins. In: DH 2017.

[23] Weintrop D, Wilensky U. Robobuilder: a computational thinking game. In: SIGCSE 2013; p. 736. doi: 10.1145/2445196.2445430

[24] Zhao W, Shute VJ. Can playing a video game foster computational thinking skills? Computers & Education 2019; 141:103633.

[25] Papert S. A critique of technocentrism in thinking about the school of the future. In: Children in the Information Age 1988; Pergamon. pp. 3-18. doi: 10.1016/b978-0-08-036464-3.50006-5

[26] Kubitza T, Schmidt A. meSchup: a platform for programming interconnected smart things. Computer 2017; 50 (11): 38-49. doi: 10.1109/MC.2017.4041350

[27] Corral JM, Balcells AC, Estévez AM, Moreno GJ, Ramos MJ. A game-based approach to the teaching of object-oriented programming languages. Computers & Education 2014; 73: 83-92. doi: 10.1016/j.compedu.2013.12.013

[28] Melcer EF, Isbister K. Bots & (Main) frames: exploring the impact of tangible blocks and collaborative play in an educational programming game. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems 2018; pp. 1-14. doi: 10.1145/3173574.3173840

[29] Papastergiou M. Digital game-based learning in high school computer science education: impact on educational effectiveness and student motivation. Computers & Education 2009; 52 (1): 1-2. doi: 10.1016/j.compedu.2008.06.004

[30] Oblinger D. The next generation of educational engagement. Journal of Interactive Media in Education 2004; 2004 (1). doi: 10.5334/2004-8-oblinger

[31] Klement M. How do my students study? An analysis of students' of educational disciplines favorite learning styles according to VARK classification. Procedia-Social and Behavioral Sciences 2014; 132: 384-90. doi: 10.1016/j.sbspro.2014.04.326

[32] Csikszentmihalyi M. Flow and the psychology of discovery and invention. NY, USA: Harper Perennial, 1997; 39.

[33] Kennedy C, Kraemer ET. Qualitative observations of student reasoning: Coding in the wild. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education 2019; pp. 224-230.

[34] Peterson CH, Peterson NA, Powell KG. Cognitive interviewing for item development: Validity evidence based on content and response processes. In: Measurement and Evaluation in Counseling and Development 2017; 50 (4): 217-23. doi: 10.1080/07481756.2017.1339564

[35] Berenguel M, Rodríguez F, Moreno JC, Guzmán JL, González R. Tools and methodologies for teaching robotics in computer science & engineering studies. Computer Applications in Engineering Education 2016; 24 (2): 202-14.

[36] Güzeliş C. An experience on problem based learning in an engineering faculty. Turkish Journal of Electrical Engineering & Computer Sciences 2006; 14 (1): 67-76.

[37] Akaslan D, Law EL. A model for flipping electrical engineering with e-learning using a multidimensional approach. Turkish Journal of Electrical Engineering & Computer Sciences 2016; 24 (5): 3419-3431. doi: 10.3906/elk-1411-144

[38] Bell T, Witten TH, Fellows M, Adams R, McKenzie J et al. CS Unplugged: An enrichment and extension programme for primary-aged students. Creative Commons 2015.

[39] Villa JM. diff: Simplifying the estimation of difference-in-differences treatment effects. The Stata Journal 2016; 16 (1): 52-71. doi: 10.1177/1536867X1601600108

[40] Bransford J, Brown A, Cocking R. How people learn: Mind, brain, experience, and school (Exp. ed.). Washington, DC: National Academy Press, 2000, p. 374.

[41] Ibanez MB, Di-Serio A, Delgado-Kloos C. Gamification for engaging computer science students in learning activities: A case study. IEEE Transactions on learning technologies 2014; 7 (3): 291-301. doi: 10.1109/TLT.2014.2329293

[42] Cagiltay NE, Ozcelik E, Ozcelik NS. The effect of competition on learning in games. Computers & Education 2015; 87: 35-41. doi: 10.1016/j.compedu.2015.04.001

[43] Hanus MD, Fox J. Assessing the effects of gamification in the classroom: a longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. Computers & Education 2015; 80: 152-161.