Development and Evaluation of a Physical Computing Game-Design Project for Students' Computational Thinking

I-Ying Hsu[1] and Fu-Hsing Tsai[2*]

[1]The Affiliated Senior High School of National Kaohsiung Normal University, Taiwan // [2]Teacher Education Center, National Chiayi University, Taiwan // tosca@tea.nknush.kh.edu.tw // fhtsai@mail.ncyu.edu.tw

[*]Corresponding author

(Submitted July 26, 2022; Revised December 8, 2022; Accepted January 30, 2023)

ABSTRACT: This study developed a physical computing game-design project that incorporates block-based programming, physical computing, and computer game design for Taiwan's high school technology education curriculum to strengthen students' computational thinking. The project asked students to develop a somatosensory computer game using a block-based programming language and physical computing devices. This study also attempted to enhance students' attitudes toward programming, technology, and engineering, and to explore the effectiveness differences between students with different majors. The research findings indicate that the project may improve students' computational thinking concepts, but did not improve students' attitudes toward programming, technology, and engineering. While participating science major students' perceptions and attitudes toward technology and engineering were significantly higher than those of social science majors, this study also found that students' performance on their project product showed no significant difference between the different groups of majors. These results imply that the application of this project could be feasible and may be beneficial to deepen science majors' interest in technology and engineering.

Keywords: Physical computing, Game design, Computational thinking, Project-based learning

1. Introduction

Computational thinking refers to the skills, cognitive procedures, and concepts that computer scientists and engineers use to operate computers and solve problems (Anderson, 2016). Wing (2006) noted that computational thinking should not be a skill exclusive to computer engineers; rather, it should be a fundamental skill that everyone should learn at an early age because such skills can also be used in daily life. This argument has prompted rapid development in computational thinking capabilities in education systems worldwide and led to computational thinking being viewed as a crucial skill for the 21st century. Many countries, such as the United States, the United Kingdom, the Netherlands, Australia, Poland, and South Korea, have prioritized cultivating computational thinking skills through their national K–12 education systems (Hsu et al., 2018; Voogt et al., 2015). Taiwan also includes computational thinking as a vital educational goal for the technology education curriculum of the national 12-year basic education program (Ministry of Education, 2018). Computational thinking has rapidly become a skill to be cultivated in every student from an early age; accordingly, the proper methods to do so have become a focus of research on education.

As programming education is an effective method of strengthening students' computational thinking capabilities (Shute et al., 2017), many countries, such as the United Kingdom, Japan, and Finland, have begun to offer programming courses in primary schools (Seow et al., 2019). This has contributed to a gradual expansion of computer science education at the primary school level. However, programming is extremely difficult to learn for beginners, especially the complex traditional text-based programming syntax (Lu et al., 2020). To make programming easy to learn and interesting, visual/block-based programming languages are considered suitable for novice programmers (Price & Barnes, 2015). For example, Scratch, Makecode, and App Inventor have been widely used in computer programming courses in primary and secondary schools as they enable novice programmers to program by manipulating block-based languages without syntax. Taiwan's technology education curriculum also recommends that students begin learning block-based programming languages in the third and fourth grades (Ministry of Education, 2020). Many studies have noted that using block-based programming languages can strengthen students' computational thinking skills (Rodríguez-Martínez et al., 2020; Sáez-López et al., 2016).

Block-based programming languages facilitate both programming and computer game design activities; for example, popular programming software Scratch uses a block-based programming language that enables students to develop game projects (Maloney et al., 2010). As students in the era of digital natives enjoy playing video games, having them design games using block-based programming languages can encourage them to engage in

programming (Mladenović et al., 2018). In addition, game production helps students learn design rules, generate and test creative ideas, and collaborate with others (Ke, 2014). Thus, computer game design is a form of learning by design (Robertson & Howells, 2008). Many studies also indicated that game design can contribute to the development of computational thinking skills (Garneli et al., 2015; Troiano et al., 2019).

Using physical computing devices to help students learn programming has also become a common way to increase students' interest in learning programming. Physical computing devices are programmable devices for which users write programs to control and develop creative, practical, and interactive hardware. Physical computing enables students to understand how programs work in physical objects and to touch the final product, which increases students' interest, unlike traditional programming, where the results are displayed on a monitor (Hodges et al., 2013). In addition, physical programming can provide students with hands-on and building experience. Thus, physical programming is appropriate for technology and engineering education courses that specifically emphasize hands-on activities. Taiwan's new technology education curriculum also recommends exposing elementary students to physical computing during the fifth and sixth grades (Ministry of Education, 2020). With the rise of the maker movement, small and inexpensive physical computing devices, such as microcontrollers, have increasingly been used in programming and technology education; for example, BBC micro:bit is an inexpensive microcontroller with many built-in sensors. As it can be used with block-based programming software, such as MakeCode and Scratch, which are easy for novice programmers to use, the BBC micro:bit has been widely used in programming education in the United Kingdom since 2016 (Ball et al., 2016). Some studies have demonstrated that using BBC micro:bit for physical computing can strengthen students' computational thinking skills (Song et al., 2020; Wu & Su, 2021).

Therefore, to strengthen students' computational thinking skills, learning block-based programming languages and applying them to game design or physical computing are common and feasible teaching strategies that make programming easy to learn and interesting. However, few studies have examined the effects of combining block-based programming with physical computing and computer game design. The BBC micro:bit microcontroller has a built-in accelerometer sensor, which enables it to detect forces in three dimensions, as well as a radio antenna to communicate wirelessly with other micro:bits. These features enable programmers to turn a device into a handheld controller for somatosensory games by applying block-based programming. Thus, block-based programming languages, physical computing devices, and computer game design can be combined with a somatosensory game.

In 2019, Taiwan began to implement a new technology education curriculum, which combines living technology education and information education that originally focused on hands-on practice and information technology, respectively. Thus, in addition to computational thinking, design thinking is a primary educational goal in the new curriculum (Ministry of Education, 2018). As the curriculum prioritizes computational thinking and hands-on practice, providing students with opportunities to engage in physical computing design projects is a good choice for Taiwan's new technology education curriculum. Therefore, this study proposed a physical computing game-design project that incorporates the block-based programming, physical computing, and computer game design for the high school technology education curriculum in Taiwan. That is, this project aimed to guide students to develop a somatosensory computer game using Scratch and BBC micro:bit as the programming language and the game controller, respectively. It also aimed to explore whether it can improve learning outcomes in terms of computational thinking concepts and attitudes toward computer programming, technology, and engineering.

Numerous studies have explored gender differences in game design (Hsu, 2013), programming, and computational thinking skills (Mouza et al., 2020; Wu & Su, 2021). However, few studies have explored the differences of learning effectiveness in computational thinking among students with different majors. High school students in Taiwan are grouped into science or social science majors in 11th grade based on aptitude. Science majors study in depth on mathematics, physics, and chemistry, while social science majors focus on social science courses, such as geography and history. However, as the technology education curriculum is compulsory for Taiwan's high school students, this study also explored the differences in the effects of the project among different majors. Specifically, the objectives of this study were, as follows.

- Explore the changes and differences of students with different majors regarding computational thinking concepts and attitudes toward computer programming, technology, and engineering after participating in the physical computing game-design project.
- Explore the differences of students with different majors regarding perceptions and project results after taking part in the physical computing game-design project.

2. Literature review

2.1. Visual / block-based programming languages

To date, many people advocate that providing programming courses is an important way to cultivate students' computational thinking (Shute et al., 2017; Voogt et al., 2015). To facilitate learning, visual and block-based programming languages are used as the primary tool in introductory programming courses for K-12 students (Portelance et al., 2016). Visual programming languages provide a shortcut for producing programming code by using icons and graphical objects to represent instructions (Myers, 1990). They are suitable for young students in the era of digital natives because they enable students to solve programming problems through trial and error (Mladenović et al., 2018).

Scratch, which is a visual programming software, was developed by MIT Media Lab to increase young students' interest in programming (Maloney et al., 2010). Students program by dragging and dropping visual blocks, similar to building blocks, which makes it a block-based programming language. The ability to stack blocks is perfect for novice programmers (Price & Barnes, 2015) as it eliminates the restrictions of traditional textual languages (João et al., 2019). According to Brennan and Resnick (2012), Scratch involves seven concepts of computational thinking: sequences, loops, events, parallelism, conditionals, operators, and data, and students can transfer these concepts to other programming or non-programming tasks. Empirical studies have noted that Scratch facilitates the development of students' computational thinking (Rodríguez-Martínez et al., 2020; Sáez-López, 2016).

Scratch can also shift the focus of programming activities from math problems to game design (Mladenović et al., 2018). As it simplifies game design, Scratch is widely used to design computer games to help students learn programming and develop computational thinking (Garneli et al., 2015). Kafai and Burke (2015) analyzed 55 studies on game design-based learning and discovered that game production helped the students learn computing concepts. Zur-Bargury et al. (2013) used Scratch in a middle school's computer science course and discovered that it helped the students learn the computational concept of loops. Mladenović et al. (2018) also discovered that using Scratch to design games was more effective than textual language-based software as it prevented misunderstanding of the computational concept of loops.

2.2. Physical computing

The term physical computing was derived from O'Sullivan and Igoe (2004) and refers to an interactive application that combines virtual and physical worlds through computer programming, sensors, microcontrollers, and tangible materials. It is also known as digital making or tangible programming (Kotsopoulos et al., 2017). Physical computing helps students learn about hardware, software, and design, and makes abstract programming concepts concrete (Kotsopoulos et al., 2017). It also encourages students to use their creativity to invent physical interactive devices (Przybylla & Romeike, 2014). As physical computing involves programming, embedded systems, and electronic engineering, physical computing education was previously only offered in university education; however, with the advances in physical devices and easily used visual programming, young students can now engage in physical programming (Jang et al., 2016). The maker movement has also facilitated the development of physical computing, as several making activities (e.g., creating robots) involve physical computing.

The physical computing device, BBC micro:bit, is an inexpensive and programmable microcontroller. It has a variety of on-board modules, including LEDs, alight sensor, compass, accelerometer, and programmable buttons. The United Kingdom has used this device for programming education since 2016, and students in more than 50 countries have used this device to learn programming and create physical products (Austin et al., 2020). In some studies, students have used the micro:bit to create paper-cutting lamps (Lu et al., 2021), headbands, stuffed animals (Klimová, 2020), and wireless remote-control car (Austin et al., 2020). Most students indicated that the micro:bit is easy to use and increased their motivation to learn programming (Gibson & Bradley, 2017). Some studies have also indicated that applying the micro:bit to physical computing activities improves students' computational thinking skills (Song et al., 2020; Wu & Su, 2021).

3. Methods

3.1. Physical computing game-design project

This physical computing game-design project was implemented in an 11th-grade technology education course (two 50-min classes per week for 20 weeks). The first half of the project lasted 9 weeks. The teacher taught the students how to use Scratch and bDesigner, which provide more block-based instructions for the micro:bit, to control micro:bit with different sensing modules (buttons, accelerometer, buzzers, and LEDs). The students also learned to wirelessly transmit information between micro:bits to create a simple somatosensory game and used the IFTTT web service to upload the game scores to the cloud services.

The second half of the project lasted nine weeks. The students worked in groups of three to five, depending on their preferences, and used Scratch and micro:bits to design a somatosensory game with any theme. Each group was provided with two mciro:bits. The first micro:bit was used to detect the player's movement for the game controller. Each group was required to use various materials, such as cardboard, to create a game controller according to their game theme, and to affix the first micro:bit in the self-developed game controller. The second micro:bit was connected with a computer as the receiver and transmitter for receiving the movement information from the first micro:bit and transmitting the messages to the self-developed Scratch game. Finally, each group was asked to present their final products in the last week.

3.2. Research procedure

This experiment was conducted in a high school's technology education course for 20 weeks. To explore the learning outcomes, during the first and last weeks, the students took pre-tests and post-tests on computational thinking concepts and completed attitude scales regarding computer programming, technology, and engineering. The participants were also required to indicate their perceptions of the project on a scale during the last week. Besides, students participated in the learning sessions from week 2 to week 10, and their project designs from week 11 to week 19.

3.3. Research participants

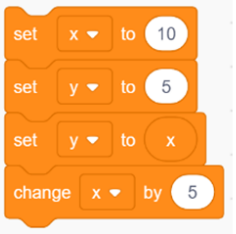
This study recruited 11th-grade students from a high school in southern Taiwan as these students accepted the new compulsory technology education curriculum. To explore the learning effectiveness differences between majors, one class of science majors (45 students) and two classes of social science majors (54 students) were selected. As four science majors did not participate in the entire experiment, 41 science majors and 54 social science majors remained, and they were divided into 10 and 14 groups, respectively, during the project. Ethical approval for this study was waived by the Taiwan Centers for Disease Control Policy # 1010265075, as this study was conducted in a general teaching environment for educational purposes, and all participants were provided the same teaching procedure and activities regardless of participants' major. This study collected no data that could identify specific individuals.

3.4. Research instruments

3.4.1. Computational thinking test

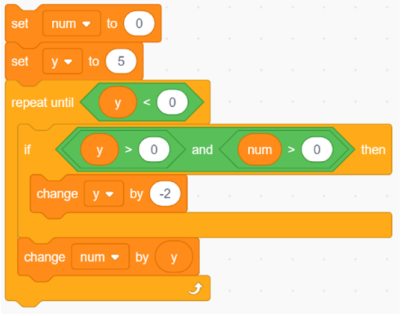
This study developed a computational thinking test including 10 items related to Scratch-based visual programming problems, which encompassed the seven computational thinking concepts of sequences, loops, events, parallelism, conditionals, operators, and data, as proposed by Brennan and Resnick (2012). Items 1–7 tested a single concept each, and Items 8–10 tested multiple concepts simultaneously, and the highest total score was 100 points. Figure 1 presents an example item testing the computational thinking concept of the data. Figure 2 presents an example item testing multiple concepts, namely data, operators, conditionals, and loops. The test was reviewed by a technology education teacher to ensure its appropriateness. According to a pilot study with 134 high school students, the Kuder-Richardson reliability was .73.

Figure 1. Computational thinking test on the concept of data



What are the final values of x and y after this program is executed?
(a)x=5, y=x
(b)x=5, y=5x
(c)x=15, y=x
(d)x=15, y=10

Figure 2. Computational thinking test on the concept of data, operators, conditionals, and loops



What are the final values of x and y after this program is executed?
(a)x=5, y=x
(b)x=5, y=5x
(c)x=15, y=x
(d)x=15, y=10

3.4.2. Computer programming attitude scale

This study referenced Korkmaz and Altun (2013) to create a computer programming attitude scale comprising three dimensions: confidence, preference, and usefulness. Each dimension contained four items, and each of which was answered on a 5-point scale. Items in the confidence and preference dimensions included "I am confident that I can learn computer programming" and "I like computer programming", respectively. The item in the usefulness dimensions included "I don't think programming will be useful in my life." According to a pilot study with 134 high school students, the Cronbach's α of this scale was .92.

3.4.3. Technology and engineering attitude scale

This study used the 9-item subscale of technology and engineering attitude in the STEM attitude scale, as developed by Faber et al. (2013), to evaluate the students' attitude changes toward technology and engineering after the project. A 5-point scale was used for scoring the items, such as "I like to imagine creating new products," "I am good at building and fixing things," and "If I learn engineering, then I can improve things that people use every day." According to a pilot study with 153 high school students, the Cronbach's α of this scale was .91.

3.4.4. The rubric of project assessment

This study referenced the Creativity Product Analysis Matrix of Besemer (1998) to develop a project assessment rubric (Table 1). The students' projects were evaluated in terms of two dimensions. The first was resolution, which evaluated the game's logic and comprehensibility, playability, and innovativeness; the highest score was 40 points. The second dimension was elaboration, which evaluated the game's basic operation and the quality of the game's software and hardware; the highest score was 60 points. Table 1 presents the scoring standards for each indicator. The evaluation standards were provided to the students as a reference before the project. All the final project products, as developed by the science and social science majors, were scored by two scorers using the abovementioned rubric, and the average scores were their final product scores. The scorer reliability was .94.

3.4.5. Participation perception scale

This study developed a 5-point scale with 11 items to understand the participants' perceptions after the project. The questions assessed the participants' feelings towards the project, as well as their knowledge acquisition, such as "I thought this project was interesting," "I acquired knowledge and skills about micro:bit through this

project," and "I felt a sense of achievement when we finished our somatosensory game." An open-ended question was included at the end of this scale, which enabled students to further provide their perceptions and suggestions. The Cronbach's α of this scale was .92.

Table 1. Project assessment indicators

Dimension	Indicators	Scoring standards	Scores
1. Resolution	1.1. Logical and understandable	• Is the game understandable and logical? • Does the game have a name, goal, and rules? • Is the game based on somatosensory principles?	0–20
	1.2. Playable	• Is the game style interesting? • Can the game attract players to play constantly?	0–10
	1.3. Innovative	• Is the game original and creative as a somatosensory game?	0–10
2. Elaboration	2.1. Basic operation	• Does the game work on Scratch? • Can the game be controlled using a wireless game controller? • Does the game run smoothly?	0–20
	2.2. Quality of software	• Are the game's graphics and art refined? • Is the game's software optimized? Does it have bugs?	0–20
	2.3. Quality of hardware	• Is the game controller well produced? • During the game, can the micro:bit be fixed firmly in the game controller without affecting the game performance?	0–20

4. Results

4.1. Analysis of students' computational thinking concepts

To determine how the students' understanding of concepts in computational thinking was changed by participating in the project, their pre-test and post-test scores in the computational thinking test were analyzed through paired sample *t*-test, and Table 2 presents the results. The science major group's post-test scores (M = 77.56, SD = 17.72) after the project were significantly higher than their pre-test scores (M = 68.29, SD = 18.01), $t(40) = 3.53$, $p < .05$. The social science major group's post-test scores (M = 67.96, SD = 22.1) were also significantly higher than their pre-test scores (M = 54.07, SD = 19.18), $t(53) = 4.69$, $p < .05$. The results indicate that the proposed project may improve students' computational thinking concepts, no matter the major.

Students' scores on different questions regarding computational thinking concepts were also explored, including seven single-concept questions (10 points per concept) and three multiple-concept questions (30 points). All students performed well in the computational thinking concepts of sequences, parallelism, and conditionals; however, the pre-test scores were low for the concepts of data, operators, loops, and multiple concepts, as shown in Table 2. After the project, the science major group's post-test scores for the concepts of data ($t(40) = 3.13$, $p < .05$) and loops ($t(40) = 2.5$, $p < .05$) were significantly higher than their pre-test scores. The social science major group's post-test scores for the concepts of data ($t(53) = 2.7$, $p < .05$), loops ($t(53) = 4.01$, $p < .05$), events ($t(53) = 3.23$, $p < .05$), and multiple-concepts ($t(53) = 2.72$, $p < .05$) were also significantly higher than their pre-test scores. In other words, the proposed project could help most students improve their computational thinking concepts regarding data and loops, and additionally improve social science major students' computational thinking concepts regarding events and multiple concepts.

To identify differences in the effectiveness of the project between students with different majors, this study used the total scores in the computational thinking pre-test and post-test scores as the covariable and dependent variable, respectively, to conduct a one-way analysis of covariance (ANCOVA). Before ANCOVA, the assumption of homogeneity of regression was conducted ($F(1, 91) = 0.02$, $p > .05$) and was not violated. The results of the ANCOVA were $F(1, 92) = 0.27$, $p > .05$, which indicated that after the effects of the covariates were excluded, and the factor of the student's major did not significantly affect the post-test scores. The post hoc comparison using the least significant difference (LSD) also revealed no significant difference in post-test scores between the science (M = 73.26) and social science majors (M = 71.23). Moreover, this study used the pre-test and post-test scores of each concept and multiple-concept scores in computational thinking as the covariable and dependent variable, respectively, to conduct one-way ANCOVA, and the results indicated that the post-test

scores did not differ significantly between majors. In other words, although the science majors' pre-test scores were significantly higher than those of the social science majors scores in the computational thinking test ($t(93) = 3.67$, $p < .05$), after the effects of prerequisite abilities were excluded, the learning effectiveness for students with different majors to acquire computational thinking concepts had no significant difference. Therefore, the proposed project could help students improve their understanding of computational thinking concepts, and the improvement efficiency had no significant difference between students with different majors.

Table 2. Results of Computational Thinking Test

	Science major group			Social science major group			Science and social science major groups
	Pre-test	Post-test	*t*	Pre-test	Post-test	*t*	ANCOVA
Total	68.29	77.56	3.53*	54.07	67.96	4.69*	.27
Sequences	10.00	10.00	0.00	9.44	9.81	1.43	.00
Events	8.04	8.75	1.00	5.93	8.33	3.23*	.01
Parallelism	10.00	9.76	-1.00	8.89	9.44	1.14	.16
Conditionals	8.78	9.27	0.81	8.15	8.15	0.00	1.98
Data	3.90	6.59	3.13*	2.04	4.26	2.70*	3.30
Operators	6.83	6.83	0.00	6.11	6.48	0.42	.04
Loops	3.90	6.34	2.50*	1.85	5.00	4.01*	.74
Multiple	16.83	20.00	1.92	11.67	16.48	2.72*	.88

Note. $^{*}p < .05$.

4.2. Analysis of students' computer programming attitudes

To determine how the project changed the students' attitudes toward computer programming, this study analyzed the students' responses to the computer programming attitude scale before and after the experiment. To facilitate statistical analysis, the answers were converted into points, with strongly agree, agree, neither agree nor disagree, disagree, and strongly disagree corresponding to the scores of 5–1 points, respectively, and Table 3 presents the statistical results. The scores indicate that the attitudes of the science majors both before and after the project were overall positive. However, the paired sample *t*-test results indicate that the mean scores before ($M = 3.51$, $SD = 0.84$) and after the project ($M = 3.46$, $SD = 0.99$) did not differ significantly ($t(40) = -0.88$, $p > .05$). The social science majors' attitudes were less positive before the project ($M = 2.44$, $SD = 0.76$), and the total mean score after the project still indicated a negative attitude ($M = 2.49$, $SD = 0.80$) without significant improvement ($t(53) = 0.69$, $p > .05$). These results suggest that before the project, the science majors had significantly more positive attitudes than the social science majors, and further analysis revealed a significant difference between the groups' attitudes before the project ($t(93) = 6.50$, $p < .05$). However, after the project, students' attitudes toward computer programming had no significant change, no matter the major. This result indicated that the proposed project may not improve students' overall computer programming attitudes.

This study also explored students' scores in the subscales. The responses to the subscales indicated that despite a low score in confidence, the science majors had a positive attitude toward programming before the project. No significant change in any of the three attitude categories was observed after the project, as the paired sample *t*-test results for the three sub-items indicated no significant difference (Table 3). The confidence and preference of social science major students toward programming before the experiment were low, and further analysis revealed that their initial confidence ($t(93) = 6.53$, $p < .05$) and preference scores ($t(93) = 6.95$, $p < .05$) differed significantly from those of the science majors. This is the primary reason that the social science majors had lower scores for overall attitudes toward computer programming before the project. However, the subscale results indicated that they found programming useful. After the project, the social science majors did not exhibit significant changes in any of the three attitude categories, and their confidence and preference toward computer programming still did not improve significantly, indicating a negative attitude. In other words, the project did not help most students improve their programming attitude regarding confidence, preference, and usefulness, regardless of major. The findings also indicated that the social science majors initially had a negative attitude toward programming in terms of their confidence and preference, which differed significantly from those of the science majors.

To determine whether the major factor affected the improvement of students' attitudes toward programming, the total mean pre-test scores of the programming attitude were used as a covariate, and the total mean post-test scores were used as a dependent variable, to conduct a one-way ANCOVA. Before ANCOVA, the assumption of homogeneity of regression was conducted ($F(1, 91) = 1.08$, $p > .05$) and was not violated. The results of the ANCOVA were $F(1, 92) = 0.20$, $p > .05$, indicating that after the effects of the covariates were excluded, the

factor of the student's major did not significantly affect the post-test scores. The LSD post hoc comparison also revealed no significant difference in mean post-test scores between the science (M = 2.95) and social science majors (M = 2.88). Moreover, this study used the scores of each subscale in the programming attitude pre-test and post-test as the covariable and dependent variable, respectively, to conduct a one-way ANCOVA. The results also indicated that the post-test scores did not differ significantly between majors (Table 3). Therefore, the project may not have significantly improved the students' attitudes toward programming in the short term. No difference in improvement efficiency was observed between the groups.

Table 3. Results of computer programming attitude

	Science major group			Social science major group			Science and social science major groups
	Pre-test	Post-test	*t*	Pre-test	Post-test	*t*	ANCOVA
Mean	3.51	3.46	-.88	2.44	2.49	.69	.20
Confidence	3.17	3.16	-.05	1.94	2.05	1.07	1.43
Preference	3.41	3.40	-.06	2.00	2.12	1.17	.33
Usefulness	3.96	3.82	-1.00	3.37	3.30	-.62	.70

4.3. Analysis of students' technology and engineering attitudes

To determine whether the project changed the students' attitudes toward technology and engineering, this study analyzed the students' responses to the technology and engineering attitude scale, which ranged from strongly agree to strongly disagree, corresponding to scores of 5–1 points, respectively. The mean score of the science majors before the project was higher than 3 points (M = 3.67, SD = 0.76), and their attitude toward technology and engineering was positive. After the project, despite a slight improvement in their mean score (M = 3.76, SD = 0.62), no significant difference was observed by paired sample t-test ($t(40) = 1.11$, $p > .05$). The mean score of the social science majors before the project was lower than 3 points (M = 2.51, SD = 0.64), indicating that their attitude toward engineering and technology was negative. Their mean score after the project was also similar to that before the project (M = 2.52, SD = 0.67), indicating no significant change by paired sample t-test ($t(53) = 0.04$, $p > .05$). Further analysis revealed a significant difference in mean scores between the groups before the project ($t(93) = 10.94$, $p < .05$). These results indicated that the project might not have significantly improved the students' attitudes toward technology and engineering. The science majors maintained a positive attitude toward technology and engineering, whereas the social science majors still had a negative attitude.

This study also performed a one-way ANCOVA using the mean pre-test and post-test scores as the covariate and dependent variable, respectively, to determine whether the factor of student's major significantly affected their improvement in attitudes toward technology and engineering. Before ANCOVA, the assumption of homogeneity of regression was conducted ($F(1, 91) = 0.06$, $p > .05$) and was not violated. The results of the ANCOVA were $F(1, 92) = 16.95$, $p < .05$, $\eta^2 = .16$, indicating that after the effects of the covariates were excluded, the factor of student's major significantly affected the post-test scores. The LSD post hoc comparison also revealed a significant difference in the adjusted mean post-test scores between the science (M = 3.37) and social science majors (M = 2.81), which means that the science majors exhibited a significantly larger change in technology and engineering attitudes than did the social science majors. Therefore, although the proposed project could not help all students significantly improve their attitudes toward technology and engineering, the improvement efficiency had a significant difference between students with different majors, thus, the project may improve the science majors' attitudes more than those of the social science majors.

4.4. Analysis of students' project products

Regardless of the students' majors, all project teams accomplished different somatosensory games according to the project task. Figure 3 presents examples of the final products. One group of social science majors transformed the micro:bit into a dumbbell that controlled a jumping game when lifted. Another group of science majors combined the micro:bit with a door handle and created a somatosensory game involving unlocking doors.

Students' project products were scored according to the rubric shown in Table 1. To explore the differences in project performances between different majors, independent sample t-test was conducted on the students' project scores, and Table 4 presents the analysis results of students' project products. Although the social science majors had slightly higher total scores (M = 77.21, SD = 7.71) than the science majors (M =71.90, SD =11.95), the difference was nonsignificant ($t(22) = -1.33$, $p > .05$), indicating that their performance did not differ significantly. Regarding the scores on the dimension of resolution, the science (M = 29.90, SD = 4.31) and social

science majors (M = 30.21, SD = 2.91) had similar scores that did not differ significantly ($t(22) = -0.21$, $p > .05$). No significant differences in the scores for the three indicators in this dimension were observed between groups (Table 4). Thus, regardless of their major, most students produced logical games, and their games' playability and innovativeness performance did not differ significantly between groups. Regarding the scores of the dimension of elaboration, although the total quality scores of the social science majors (M = 47, SD = 5.04) were slightly higher than those of the science majors (M = 42.00, SD = 7.96), the difference was nonsignificant. In addition, the scores of social science majors in each of the three indicators of the elaboration dimension were slightly higher than those of the science majors; however, no significant difference was observed. This suggests that no significant difference in outcome was observed between the groups. Therefore, although the social science majors exhibited poor performance in terms of their understanding of concepts in computational thinking, and they had a negative attitude toward computer programming before the project, their performance did not differ significantly from that of the science majors, and they all accomplished the project task.

Figure 3. Students' somatosensory games

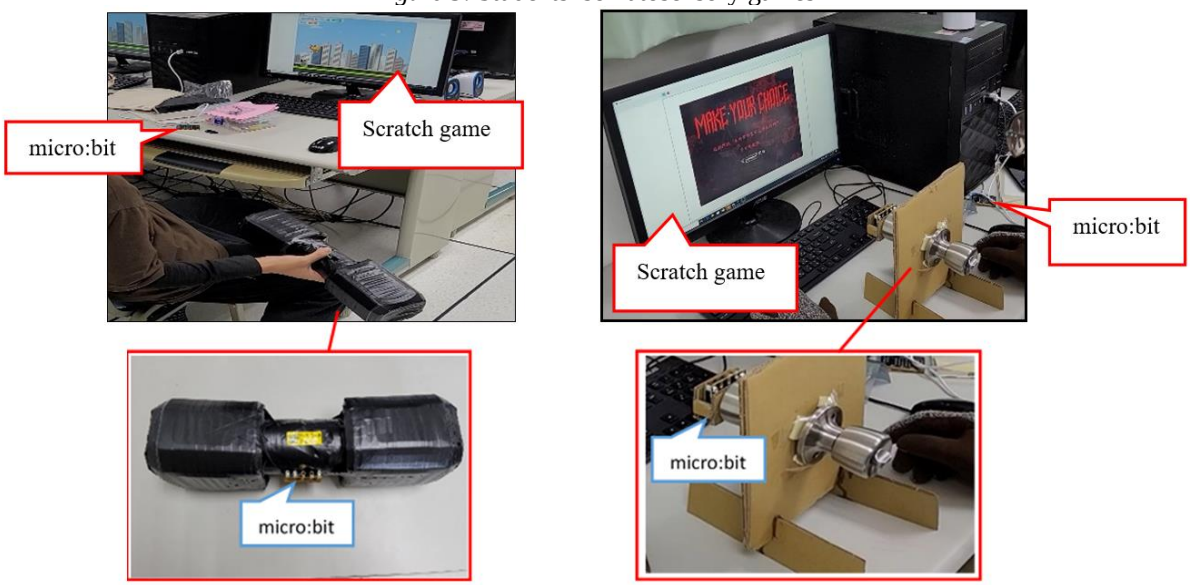


Table 4. Analysis of final products

	Science major group (n = 10)	Social Science major group (n = 14)	t
Total score	71.90	77.21	-1.33
1. Resolution	29.90	30.21	-.21
1.1: Logical and understandable	15.90	16.64	-1.15
1.2: Playable	7.10	7.07	.05
1.3: Innovative	6.90	6.50	.56
2. Elaboration	42.00	47.00	-1.89
2.1: Basic operation	14.70	16.50	-2.06
2.2: Quality of software	14.70	15.43	-.86
2.3: Quality of hardware	12.60	15.07	-1.72

4.5. Analysis of students' participating perception

To identify the differences in students' participating perceptions toward the project between the different majors, independent sample t-test was performed on the students' responses to the perception scale. To facilitate statistical analysis, responses were converted to the scores of 1 (*strongly disagree*) to 5 points (*strongly agree*). While the participating perceptions of both the science (M = 3.84) and social science majors (M = 3.55) were positive, the t-test results were $t(93) = 2.13$, $p < .05$, indicating that the mean score of the science majors was significantly higher than that of the social science majors. This suggests that the science majors' impressions of the project were more positive than those of the social science majors in terms of satisfaction and gain.

Analysis of the students' responses to the open-ended questions on the scale indicated that most students were satisfied with the project. Social science majors stated, "Designing a somatosensory game on my own was a unique experience," "Now I know how to apply programming for a range of tasks," and "I learned how to create

new games with my team members." Science majors stated, "The project helped me understand Scratch and micro:bit more," "This was my first experience using a microcontroller to execute a program I wrote, and it was really exciting and might help me in the future," and "I learned a lot about micro:bit."

5. Discussion

This study found that both science and social science major groups' post-test scores on computational thinking tests were significantly higher than those of their respective pre-test scores after the project, and the results of ANCOVA showed no significant difference between the groups. In other words, students significantly improved their scores in computational thinking, and the improvement efficiency did not differ from their initial concepts of computational thinking after participating in the project. Thus, this project helped all students strengthen their computational thinking concepts. The findings demonstrate that the combination of block-based programming, physical computing, and computer game design may be beneficial to students' computational thinking. The findings also support those of other studies regarding the use of Scratch, physical computing, and game design to improve students' computational thinking skills (Mladenović et al., 2018; Rodríguez-Martínez et al., 2020; Sáez-López, 2016; Wu & Su, 2021; Zur-Bargury et al., 2013).

Moreover, the analysis results of the students' computational thinking test also indicated that the changes in their scores were mainly due to the significant increase in all students' computational thinking concepts regarding data and loops, and social science majors' events concept and multiple concepts. However, the students' computational thinking concepts of data, operators, loops, and multiple concepts can still be improved. These results are consistent with those of other studies, in which novice programmers exhibited poor performance in terms of their understanding of data, operators, and loops concepts (Grover & Basu, 2017).

Regarding students' attitudes toward computer programming, technology, and engineering, this study found that students with different majors differed significantly in their attitudes before the project; the social science majors had negative attitudes, lacked confidence and preference for programming, and were uninterested in technology and engineering, whereas the science majors exhibited positive attitudes. After the project, the students' attitudes did not differ significantly from those before the project. The social science majors' confidence and preference for programming did not increase significantly, and their attitudes toward technology and engineering remained similar, as did those of the science majors. This suggests that the project did not significantly improve the students' attitudes toward computer programming, technology, or engineering, which may be because the project was too short to result in immediate changes in attitude or because the students had already established career goals related to their majors. Nonetheless, the project could still be effective to influence science major students' attitudes toward technology and engineering. The ANCOVA analysis results of the technology and engineering attitude scale suggests that the project improved the science majors' technology and engineering attitudes more than the social science majors' attitudes.

The science and social science majors did not significantly differ in terms of the scores for their project products, which indicates that although science majors initially exhibited a deeper understanding of programming concepts and more positive attitudes toward programming and technology creation than the social science majors, their somatosensory games were not significantly superior to those of social science majors. In addition, although the social science majors exhibited poor performance in terms of their initial understanding of the concepts of computational thinking and lacked interest in programming and technological creation, they completed the project successfully after a semester of study and activities. In other words, the difficulty level of the proposed project was appropriate, and the approach may be appropriate for all students. Therefore, this project could continue to be implemented in Taiwan's technology education curriculum.

Regarding the students' perceptions toward the project, this study found that most students were satisfied and indicated that it helped them acquire knowledge. This supports the above discussion, indicates that the difficulty of the project was appropriate, and explains the finding that social science majors can produce the same quality products as the science majors and improve their computational thinking. The analysis results also revealed that science majors' scores for participating perception were significantly higher than those of the social science majors, indicating that the science majors enjoyed the project more than the social science majors; however, this result might be related to the students' differences in attitudes toward programming and technology between majors.

6. Conclusion

This study combined block-based programming, physical computing, and game design in a project for Taiwan's high school technology education curriculum to increase interest in computer programming and technology and strengthen computational thinking. This study also explored differences in the project's effects between majors, and the results indicate that the project may improve students' computational thinking concepts; however, the project may not improve students' attitudes toward programming, technology, and engineering. Regarding the comparisons of different majors, this study found science majors exhibited better performance in their computational thinking test, and they had significantly more positive attitudes toward programming and technology than the social science majors before the project. Nevertheless, no significant differences were observed in the computational thinking concepts, programming and technology attitudes, and the final product scores after the project between the groups. These findings imply that the proposed project could be beneficial to all students and feasible for a compulsory technology education curriculum. Moreover, this study also found that science majors' scores regarding their participation perception and technology and engineering attitude were significantly higher than those of the social science majors, which implies that this project may be beneficial to deepen science majors' interest in technology and engineering.

Therefore, this proposed project can continue to be promoted in Taiwan's technology education curriculum; however, future research should increase the number of participants to further verify the effectiveness between different majors. In addition, more rigorous methods can be conducted in the future to assess students' projects.

Acknowledgment

This research was financially supported by the Ministry of Science and Technology, Taiwan, under Grant no. MOST 109-2511-H-415-008-MY3.

References

Anderson, N. D. (2016). A Call for computational thinking in undergraduate psychology. *Psychology Learning & Teaching, 15*(3), 226–234. https://doi.org/10.1177/1475725716659252

Austin, J., Baker, H., Ball, T., Devine, J., Finney, J., De Halleux, P., Hodges, S., Moskal, M., & Stockdale, G. (2020). The BBC micro: bit: from the UK to the world. *Communications of the ACM, 63*(3), 62-69. https://doi.org/10.1145/3368856

Ball, T., Protzenko, J., Bishop, J., Moskal, M., De Halleux, J., Braun, M., Hodges, S., & Riley, C. (2016). Microsoft touch develop and the BBC Micro: bit. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)* (pp. 637-640). IEEE.

Besemer, S. P. (1998). Creative product analysis matrix: Testing the modelstructure and a comparison among products--Three novel chairs. *Creativity Research Journal*, *11*(4), 333–346. https://doi.org/10.1207/s15326934crj1104_7

Brennan, K., & Resnick, M. (2012). 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* (Vol. 1, pp. 25).

Faber, M., Unfried, A., Wiebe, E. N., Corn, J., Townsend, L. W., & Collins, T. L. (2013). Student attitudes toward STEM: The development of upper elementary school and middle/high school student surveys. In *2013 ASEE Annual Conference & Exposition* (pp. 23-1094). https://doi.org/10.18260/1-2--22479

Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015). Computing education in K-12 schools: A Review of the literature. In *2015 IEEE Global Engineering Education Conference (EDUCON)* (pp. 543-551). IEEE.

Gibson, S., & Bradley, P. (2017). A Study of Northern Ireland key stage 2 pupils' perceptions of using the BBC Micro: bit in STEM education. *The STeP Journal, 4*(1), 15-41.

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 267-272). https://doi.org/10.1145/3017680.3017723

Hodges, S., Scott, J., Sentance, S., Miller, C., Villar, N., Schwiderski-Grosche, S., Hammil, K., & Johnston, S. (2013). . NET Gadgeteer: A New platform for K-12 computer science education. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 391-396). https://doi.org/10.1145/2445196.2445315

Hsu, H. M. J. (2013). Gender differences in elementary school students' game design preferences. *International Journal of Information and Education Technology, 3*(2), 172-176.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296-310. https://doi.org/10.1016/j.compedu.2018.07.004

Jang, Y., Lee, W., & Kim, J. (2016). The Changes of middle school students' perception and achievement based on the teaching method in physical computing education. *Indian Journal of Science and Technology*, *9*(24), 1-8.

João, P., Nuno, D., Fábio, S. F., & Ana, P. (2019). A Cross-analysis of block-based and visual programming apps with computer science student-teachers. *Education Sciences, 9*(3), 181. https://doi.org/10.3390/educsci9030181

Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. *Educational psychologist*, *50*(4), 313-334. https://doi.org/10.1080/00461520.2015.1124022

Ke, F. (2014). An Implementation of design-based learning through creating educational computer games: A Case study on mathematics learning during design and computing. *Computers & Education*, *73*, 26-39. https://doi.org/10.1016/j.compedu.2013.12.010

Klimová, N. (2020). Wearables: Educational projects made with the BBC micro: bit. In *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)* (pp. 323-329). IEEE.

Korkmaz, Ö., & Altun, H. (2013). Engineering and ceit student's attitude towards learning computer programming. *International Journal of Social Science, 6*(2), 1169–1185.

Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J., & Yiu, C. (2017). A Pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education*, *3*, 154-171.

Lu, C. C., Hong, J. C., Chen, F. F., & Ma, S. Y. (2020). Elementary school students learn Arduino programming to assemble sensory-controlled works. *Int. J. Inf. Educ. Technol, 10*(4), 265-270.

Lu, S. Y., Lo, C. C., & Syu, J. Y. (2021). Project-based learning oriented STEAM: The Case of micro–bit paper-cutting lamp. *International Journal of Technology and Design Education*, *32*, 2553–2575. https://doi.org/10.1007/s10798-021-09714-1

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 1-15. https://doi.org/10.1145/1868358.1868363

Ministry of Education (2018). *Curriculum guidelines of 12-year basic education for Technology domain*. https://www.k12ea.gov.tw/files/class_schema/課綱/13-科技/13-1/十二年國民基本教育課程綱要國民中學暨普通型高級中等學校—科技領域.pdf

Ministry of Education (2020). *The Reference instructions of curriculum development for the primary living technology and information technology education*. https://www.naer.edu.tw/upload/1/9/doc/280/1090611_國民小學科技教育及資訊教育課程發展參考說明.pdf

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies, 23*, 1483-1500. https://doi.org/10.1007/s10639-017-9673-3

Mouza, C., Pan, Y. C., Yang, H., & Pollock, L. (2020). A Multiyear investigation of student computational thinking concepts, practices, and perspectives in an after-school computing program. *Journal of Educational Computing Research, 58*(5), 1029-1056. https://doi.org/10.1177/07356331209056

Myers, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, *1*(1), 97-123. https://doi.org/10.1016/S1045-926X(05)80036-9

O'Sullivan, D., & Igoe, T. (2004). *Physical computing: Sensing and controlling the physical world with computers*. Course Technology Press.

Portelance, D. J., Strawhacker, A. L., & Bers, M. U. (2016). Constructing the ScratchJr programming language in the early childhood classroom. *International Journal of Technology and Design Education, 26*, 489-504. https://doi.org/10.1007/s10798-015-9325-0

Price, T. W., & Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the eleventh annual international conference on international computing education research* (pp. 91-99). https://doi.org/10.1145/2787622.2787712

Przybylla, M., & Romeike, R. (2014). Physical computing and its scope--Towards a constructionist computer science curriculum with physical computing. *Informatics in Education, 13*(2), 241-254.

Robertson, J., & Howells, C. (2008). Computer game design: Opportunities for successful learning. *Computers & Education, 50*(2), 559-578. https://doi.org/10.1016/j.compedu.2007.09.020

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments, 28*(3), 316-327. https://doi.org/10.1080/10494820.2019.1612448

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A Two year case study using “Scratch” in five schools. *Computers & Education, 97*, 129-141. https://doi.org/10.1016/j.compedu.2016.03.003

Seow, P., Looi, C. K., How, M. L., Wadhwa, B., & Wu, L. K. (2019). Educational policy and implementation of computational thinking and programming: Case study of Singapore. In *Computational thinking education* (pp. 345-361). Springer. https://doi.org/10.1007/978-981-13-6528-7

Shute, V., Sun, C., Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Song, O. J., Park, E. K., & Bae, J. M. (2020). The Effect of software education using Micro: bit on computational thinking of elementary school students. *Journal of Knowledge Information Technology and Systems (JKITS), 15*(1), 37-46.

Troiano, G. M., Snodgrass, S., Argımak, E., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G., & Harteveld, C. (2019). Is my game OK Dr. Scratch? Exploring programming and computational thinking development via metrics in student-designed serious games for STEM. In *Proceedings of the 18th ACM international conference on interaction design and children* (pp. 208-219). https://doi.org/10.1145/3311927.3323152

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies, 20*, 715–728. https://doi.org/10.1007/s10639-015-9412-6

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wu, S. Y., & Su, Y. S. (2021). Visual programming environments and computational thinking performance of fifth-and sixth-grade students. *Journal of Educational Computing Research*, *59*(6), 1075-1092. https://doi.org/10.1177/07356331209888

Zur-Bargury, I., Pârv, B., & Lanzberg, D. (2013). A Nationwide exam as a tool for improving a new curriculum. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 267-272). ACM Press. https://doi.org/10.1145/2462476.2462479