TPACK leveraged: A redesigned online educational technology course for STEM preservice teachers

Duygu Umutlu
Bogazici University, Turkey

Integration of computational thinking and programming into science, technology, engineering, and math (STEM) classes is needed to promote students' learning of twenty-first century skills. Yet, teachers are not equipped to achieve this integration successfully as teacher education curricula do not generally align with this need. With the Covid-19 outbreak, curricula also need to be adapted for online environments. This qualitative study presents the redesign of an educational technology course that introduces programming and computational thinking to STEM preservice teachers for online settings, and explores learning experiences of preservice teachers, in terms of how they combine technological knowledge with pedagogy and content. Data were collected from course artifacts, such as reading responses, coding challenges, and lesson designs and implementations. The findings showed the online course design was helpful in enhancing preservice STEM teachers' pedagogical approaches of how to teach computational thinking and programming. Offering hands-on coding practices in the course allowed preservice teachers to improve their technological knowledge (programming), and they were able to integrate their technological pedagogical knowledge into their content area and design meaningful lessons. The study offers implications for design of online teacher education courses that promote preservice teachers' technological pedagogical content knowledge for computational thinking and programming.

Implications for practice or policy:

- The online course design implemented in this study can be adjusted into different contexts, considering that fully-online or blended teacher education courses will still be needed in the future.
- The design guidelines used in this study can be utilised to develop online teacher education modules for educational technology topics other than programming.
- The question prompts given to preservice teachers in the study can be refined to trigger deeper reflection on pedagogy of computing education.

Keywords: TPACK, online teacher education, programming, computing education, STEM preservice teachers

Introduction

With the Covid-19 outbreak, all educational institutions in the world underwent an abrupt transition from face-to-face courses to fully online courses. Most higher education institutions and faculty were not prepared for this rapid switch to fully online education, and they were forced to revise and adjust their course designs (Johnson et al., 2020). Design of teacher education courses also had to be adapted for implementation in online education settings, and this was rather demanding for many instructors. When it comes to educational technology courses for teacher candidates, this adaptation becomes even more challenging as they usually have a component that requires practice, that is, lab exercises and sessions.

This paper presents the redesign of an educational technology course for online settings. Because of the pandemic, this educational technology course had to be implemented fully online through videoconferencing tools. The course introduces STEM preservice teachers to block-based programming and how the use of block-based programming and computational thinking can be integrated into their classes. The instructional design model used for the course in this study was informed by the relevant literature of technological pedagogical and content knowledge (TPACK) framework (Mishra & Koehler, 2006), online learning communities, and computing education.

TPACK is an appropriate framework to be used in the design of courses for pre-service teachers' learning and integrating of computational thinking (Yadav, Stephenson, et al., 2017). The TPACK framework

(Mishra & Koehler, 2006) suggests that teachers learn to integrate technology effectively by combining their own content area and pedagogical knowledge. Subject area teachers' learning of computer science can also be facilitated through educational technology courses grounded in the TPACK framework (Margulieux & Yadav, 2020; Yadav, Gretter et al., 2017; Yadav et al. 2016). Yet, there is a scarcity of studies that examine how the framework should be adjusted for computer science education, including programming and computational thinking (Hubbard, 2018; Ioannou & Angeli, 2015; Mouza et al., 2017; Yadav, Stephenson et al., 2017). In their study, Mouza et al. (2017) examined preservice teachers' computational thinking when they were involved in block-based programming, and formed a new type of technological knowledge: technological knowledge-computational thinking. Yet, how teacher candidates' technological and pedagogical knowledge for computational thinking for content area classes can be developed through block-based programming still needs to be investigated (Grover et al., 2020; Mouza et al., 2017; Rich et al., 2020).

As this study was conducted in the context of online education due to the pandemic, the face-to-face format of the course was adapted by combining the TPACK framework with literature-derived guidelines for online learning communities and computing education. As preparing teachers for computational thinking integration is important to equip them with necessary skills for twenty-first century classrooms (Caskurlu et al., 2021), this study is timely since it reports how teacher education courses for computational thinking and computing education continue to be implemented in online environments. To implement the TPACK-based instructional design model effectively in online settings, pedagogical strategies for computational thinking (Kotsopoulos et al., 2017) were adjusted along with Yuan and Kim's (2014) guidelines for online learning communities. The purpose of the study was to examine the learning experiences of preservice teachers within the redesigned online course. The research questions that guided the study were:

1. What are the learning experiences of preservice teachers in the redesigned online educational technology course?
2. What are the TPACK indicators that emerged in these learning experiences?

Theoretical framework

TPACK

Examining the interplay among content, pedagogy, and technology, TPACK is a useful framework to guide the design of teacher education courses. It also informs teacher educators about what knowledge is required for effective technology integration. In the TPACK framework, technological knowledge refers to knowledge of how to use technologies in learning environments properly, pedagogical knowledge refers to the use of appropriate methods and techniques to teach a subject matter, and content knowledge is "the actual subject matter that is to be learned or taught" (Mishra & Koehler, 2006, p. 1026).

These three components of the framework should interact with each other to form teachers' knowledge required to create meaningful learning experiences in technology-integrated classes (Koehler et al., 2013; Niess, 2011). With all the intersections, the framework has seven components namely content knowledge (CK), pedagogical knowledge (PK), technological knowledge (TK), pedagogical content knowledge (PCK), technological content knowledge (TCK), technological pedagogical knowledge (TPK), and TPACK (Koehler & Mishra, 2009). The core intersection (TPACK) includes a synthesis of three knowledge types that supports effective use of technologies with appropriate pedagogical approaches in a specific content area (Mishra & Koehler, 2006).

Although it has been reported that the framework is useful in teacher preparation for technology integration, there is no agreed-upon roadmap for the development of preservice teachers' TPACK. Aligning with the seven interacting components of the TPACK framework, the main guiding principle of teacher education courses is providing tasks that encourage preservice and in-service teachers to synthesise technological, pedagogical, and content knowledge (Koehler et al., 2013). In their review, Chai et al. (2013) examined several interventions in teacher education and found that most of them started with either PCK, TCK, TPK, or TPACK. After careful examination of teacher education models, Koehler et al. (2014) suggested three primary approaches to teachers' TPACK development: (a) from PCK to TPACK, (b) from TPK to TPACK, and (c) developing PCK and TPACK simultaneously. According to Koehler et al. (2014), the first approach is prevalently adopted for in-service teacher training while the second one is regarded as the default approach for preservice teacher education. Developing PCK and TPACK simultaneously is usually implemented in methods courses in teacher education programs.

Considering the participants' background, the approach utilised in the design of the online educational technology course within the scope of this study, was the default approach for preservice teacher education, that is, from TPK to TPACK. The designed course was an elective opened to junior and senior students. Accordingly, 13 out of 15 participants in this study were 4th-year students who had already taken most of the methods and required content area courses in their teacher education programs. Therefore, it would be expected that they would have acquired the PCK for their subject areas before taking this course. However, they may not have had the appropriate TK and PK for programming and computational thinking. Chai et al.'s (2013) definitions of TPACK dimensions were adapted for this study: (a) TK refers to the knowledge of block-based programming, (b) PK includes knowledge of how to integrate block-based programming and computational thinking into their content area classes, and (c) CK is participants' knowledge of the subject matter they previously gained. Given participants' lack of knowledge in block-based programming, the course mainly focused on teaching block-based programming. While learning block-based programming during the classes, participants also explored several pedagogical approaches that can be adjusted for computer science education. In the last 3 weeks of the course when they peer-taught their lessons, participants had the opportunity of integrating their technological and adjusted pedagogical knowledge into their existing CK.

Online learning communities

Referring to Lave and Wegner's (1991) communities of practice, Yuan and Kim (2014) defined an online learning community as a group of learners who aim to achieve a shared goal in a virtual environment. In online learning, it is important to create communities to eliminate or mitigate any potential student disengagement from courses because of the identified dropout problem (Stiller & Bachmaier, 2017). This challenge has become more formidable with the rapid transition to fully online education at most educational institutions in the world since the Covid-19 pandemic started in March 2020.

With the goal of enhancing online education, aspects of effective online learning communities have been investigated widely. For instance, Garrison et al. (2000) asserted that there are three types of presences that need to be developed to form communities in online learning environments: *teaching presence*, *social presence*, and *cognitive presence*. Teaching presence refers to instructors' design and facilitation of the online learning processes. Social presence involves learners' communication and interaction with their peers and the instructor during online learning. Cognitive presence refers to learners' construction of knowledge in online communities. Aligning with these three types of presences, Fiock (2020) proposed several meaningful activities and strategies to build up effective online learning communities. For instance, designing a wide variety of instructional activities and encouraging both student-student and student-teacher interactions, are two of the offered strategies for teaching presence (Fiock, 2020). Encouraging students to share their own experiences and implementing group work in classes enhance social presence of students. Cognitive presence can be achieved by allowing students to examine course topics through multiple perspectives and supporting active learning.

Following the onset of the pandemic in 2020, Wang (2021) also offered an instructional design model for creating content, activities, facilitation, and evaluation (CAFE) in online learning environments. Wang (2021) suggested that instructors should develop content for their courses systematically and offer various instructional activities in online settings. Communications between teacher-learner, learner-learner, and learner-content should also be facilitated by instructors (Wang, 2021). As for evaluation, Wang (2021) suggested that learners' performances in online environments should be assessed holistically. In particular, the facilitation component of the CAFE model may offer some guidance on how to form learning communities as it relates to interaction in online learning environments.

Similarly, Yuan and Kim (2014) proposed four guidelines for creating online communities. In these guidelines, the who, when, where, and how of forming learning communities were discussed. Yuan and Kim (2014) proposed that both instructors and learners should engage in community building, and this process should continue throughout the semester. Both asynchronous and synchronous activities that involve different instructional strategies should also be designed for effective learning communities in online environments (Yuan & Kim, 2014). Focusing on teacher education programs in the emerging post-pandemic era, Thomas (2020) and Jin (2022) argued that teacher education programs need to adapt to online settings, and classroom communities of teacher candidates should be formed to enhance their social presence.

Computing education

The US National Centre for Computing Education (NCCE) (2020) published 12 computing education pedagogy principles. These principles suggest that computing education should occur in a project-based learning environment blended with modeling, hands-on experiments, and peer collaboration. Also, instructors should provide students with opportunities to explore and get better conceptual understanding of computing through concrete examples. Designing structural lessons that include a wide variety of activity types is another important part of computing education.

Referring to constructionism (Papert & Harel, 1991), Kotsopoulos et al. (2017) suggested a computational thinking pedagogical framework. Their framework includes four pedagogical activity types which align with the pedagogy guidelines (National Centre for Computing Education [NCCE], 2020): (a) *unplugged* activities for modeling and concrete examples, (b) *making* for computing projects and peer collaboration, (c) *tinkering* for exploration, and (d) *remixing* for hands-on activities for programming. Unplugged activities include computational thinking tasks that can be implemented without computers, such board games and modelling. Making refers to hands-on projects that can be designed for individual or group work. Tinkering means learners' exploration during programming. In remixing activities, some parts of a program are given to learners, and they are asked to change those codes to create a new program. How these relevant principles and guidelines were used to form the instructional design model in this study is discussed in the following section.

Methods

Research design

This study employed a case study design (Stake, 1995). The online educational technology class where the instructional design model was implemented was treated as the case of the study. The goal behind doing so was to get a better understanding of the online learning experiences of preservice STEM teachers throughout the course and how the instructional design model shaped these experiences (Hyett et al., 2014; Thomas, 2011).

Participants and setting

Participants of this study were 15 preservice teachers (4 males, 11 females) from early childhood education, math education, and science education programs at a large public university in Turkey (Table 1). The online course designed was an elective course about block-based programming. Participants were purposefully selected to be mostly senior undergraduate students in the college of education (Patton, 1990). That is, they were close to graduation and starting to teach in real classrooms. None of them had prior knowledge of block-based programming. The course where data were collected lasted 14 weeks in a fully online format. The class met synchronously via a videoconferencing tool for 3 hours every week. The course assignments and projects participants completed over 14 weeks were included in the dataset of this study.

At the beginning of the semester, ethics approval was given by the institutional review board of the university where the study was conducted. Participants were informed about the study and asked to sign the consent form. All the students in the course agreed to participate in the study.

Table 1
Demographic information about participants

Participant ID	Program	Year in the program
Participant 1	Science education	Senior
Participant 2	Early childhood education	Senior
Participant 3	Early childhood education	Senior
Participant 4	Early childhood education	Senior
Participant 5	Mathematics education	Senior
Participant 6	Mathematics education	Senior
Participant 7	Mathematics education	Senior
Participant 8	Science education	Junior
Participant 9	Science education	Senior
Participant 10	Science education	Junior
Participant 11	Science education	Senior
Participant 12	Mathematics education	Senior
Participant 13	Early childhood education	Senior
Participant 14	Science education	Senior
Participant 15	Mathematics education	Senior

Course prototype

Grounded in the TPACK framework, the instructional design model of the course prototype used literature-derived guidelines about online learning communities and computing education. The course aimed to introduce both block-based programming and instructional strategies to integrate programming and computational thinking into teaching to preservice STEM teachers. Following the pathway from TPK to TPACK described in the *Theoretical framework* section, the course started with Scratch challenges with the goal of improving participants' TK. Almost simultaneously, reading responses were included in the course to encourage participants to delve into pedagogical approaches in computer science education. After participants started to synthesise their existing PK with the newly-learned TK, that is, block-based programming, they were asked to put these knowledge types into practice within their subject area. Table 2 summarises the key aspects of the instructional design model developed, based on Yuan and Kim's (2014) guidelines for online learning communities, underpinned by the TPACK framework.

Table 2
Online course activities based on TPACK

Technological knowledge (TK)	
Instructional activity: Scratch challenges	
When	• Biweekly throughout the semester (Weeks 3 to 9)
Where	• Asynchronous: Individual coding and written summary
	• Synchronous: Small-group debugging
How	• Individual and small-group work
Pedagogical knowledge (PK)	
Instructional activity: Reading responses (to 4 assigned articles about pedagogical approaches in computer science education)	
When	• Biweekly throughout the semester (Weeks 4 to 10)
Where	• Asynchronous: Written reading responses (individual task)
	• Synchronous: Whole-class discussions during live classes
How	• Individual work and whole-class discussions
TPACK (TK + PK → TPK + CK → TPACK)	
Instructional activity: Final lesson plan projects & online peer-teaching	
When	• Last three weeks of the semester (Weeks 11 to 14)
Where	• Asynchronous: Group work for lesson design preparation
	• Synchronous: Peer-teaching during live classes
How	• Group work and individual reflection

In the TK phase, STEM preservice teachers in the course completed four Scratch coding challenges and wrote summaries of the computational thinking phases they went through during coding. After they coded the challenges individually, they attended small group meetings with their classmates to discuss the challenging parts they got stuck in, and revised their coding based on the feedback they received. In the PK phase, they read an article and wrote a reading response based on a given prompt before the synchronous class time biweekly (Table 3). Although they wrote these reading responses individually, they discussed the articles in small groups and as a whole-class during synchronous class sessions. These two phases were interrelated and completed almost simultaneously (Table 2). While gaining TK through the coding challenges, participants had a chance to read about how similar coding activities were implemented in K-12 classrooms (PK) in the articles.

Table 3
Assigned articles for PK

Article #	Content summary
Article 1	Coding and robotics through debugging in elementary classrooms
Article 2	Instructional approaches in computer science education
Article 3	Use of robots and block-based programming in kindergarten classes
Article 4	How teachers should bring computer science into K-12 classes

For the TPACK phase, a final project was designed for the course. The course final project was to prepare a lesson plan and peer-teach it during online live class sessions. Within the scope of the final project, STEM preservice teachers in the course were asked to select a content area topic and integrate block-based programming and computational thinking into it for lesson planning. As a result, these preservice teachers had the opportunity of integrating TPK they gained through coding challenges and reading responses earlier in the semester into CK in the final project. In peer-teaching sessions, preservice teachers in the course became teachers in turns and taught the subject area topic they selected to their classmates who role-played to be their students.

Data collection

The data set of the study includes participants' artifacts: (a) 60 weekly Scratch challenge summaries, (b) 60 reading responses, (c) 5 final lesson design projects, and (d) the researcher's observation notes during participants' online peer-teaching. Participants coded four Scratch challenges biweekly from the Week 3 to 9 over the semester. After they coded those challenges, they wrote short summaries of difficulties they encountered and which computational thinking phases they experienced while coding. They also individually wrote four short reading responses (about 400 words each) about the computer science education articles (Table 3) read biweekly from the Week 4 to 10 over the semester. In the last 3 weeks of the semester, participants in groups of three developed lesson plans that integrated programming and computational thinking into their subject areas and had their online peer-teaching sessions during live classes. The researcher took observation notes during these peer-teaching sessions.

Data analysis

Qualitative content analysis (Mayring, 2000) through concept mapping was employed to analyse the data. *Quirkos* was used as the qualitative data analysis software tool to visually examine the maps. Using TPACK as a framework for analysis (Mishra & Koehler, 2006), *inductive categories* were generated through coding of the raw data. As suggested by Tracy (2020), the researcher and a research-fellow who was well-versed in computing education for preservice teachers, first coded the data independently. Each researcher coded one set of data which includes one reading response, one Scratch challenge code and summary, and one lesson plan. Following this individual coding, they discussed the emerging codes and created a codebook that included initial codes. Based on the initial codebook, they each individually coded another set of data. After several cycles of coding and revisions, they reached a consensus, and the researcher coded the rest of the dataset. Reviewing the whole dataset coded, the researcher crafted categories, patterns, and themes. An experienced qualitative researcher who is a teacher educator actively using TPACK both in teaching and research reviewed the codebook, codes, and themes, and the researcher revised the themes through peer debriefing.

The final version of the codebook included three main categories: technological, pedagogical, and content knowledge. Under each main category, there were two sub-categories that included 53 codes in total. As shown with the arrows in Figure 1, the categories were not mutually exclusive. *Coding knowledge* under

TK referred to participants’ knowledge that they gained through block-based programming experiences, such as Scratch challenges throughout the semester. The second sub-category, *computational thinking phases*, involved the stages participants went through while developing their block-based programs. Under PK, *pedagogical perspectives* were defined as participants’ viewpoints about how and why to integrate coding and computational thinking into their lessons, such as using coding to improve their students’ problem solving. *Instructional approaches* involved strategies specific to computer science education, such as remixing, making (hands-on experience), and pair programming. When it came to CK, the two sub-categories also complemented each other. *Adjustment of the content area topic* included participants’ selection of appropriate topics and tailoring them into coding-inserted lessons. To develop effective lesson plans, it was also important to integrate learning objectives of both content area and programming.

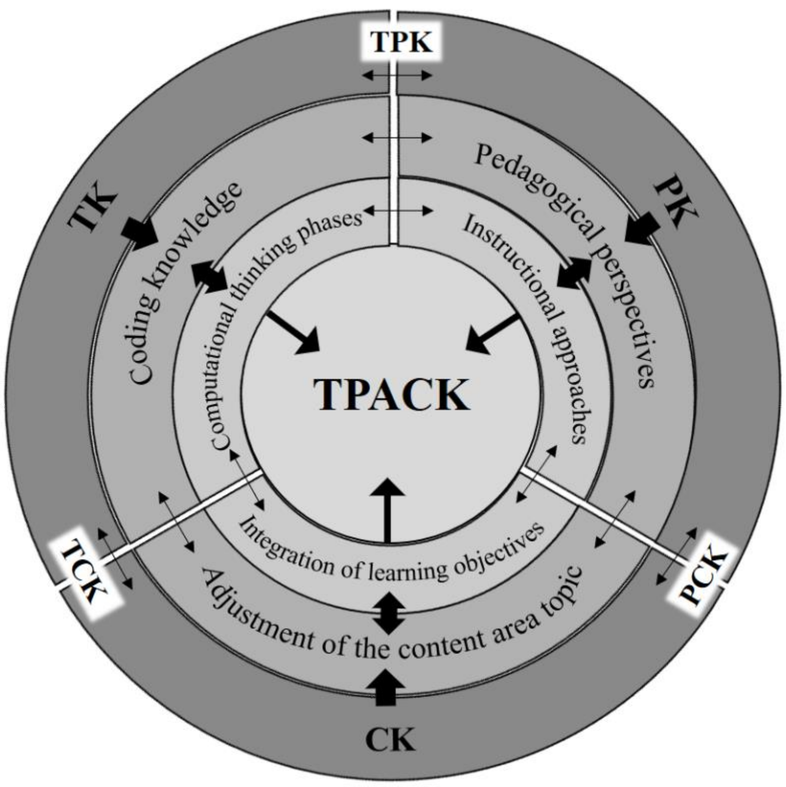

TPK
TK
PK
Coding knowledge
Pedagogical perspectives
Computational thinking phases
Instructional approaches
TPACK
Integration of learning objectives
Adjustment of the content area topic
TCK
PCK
CK

Figure 1. Main and sub-categories in the codebook

Findings

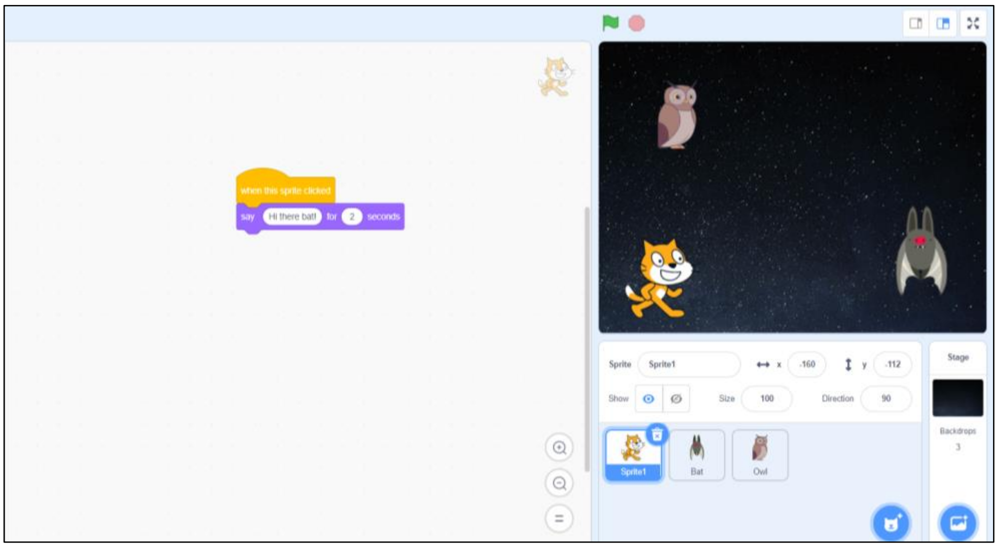
The findings revealed that STEM preservice teachers who plan to integrate block-based programming into their teaching in the future had an opportunity to examine and plan their probable teaching practice from different aspects in the designed educational technology course. It can also be seen that the proposed instructional design model was effective in enhancing STEM preservice teachers’ TPACK for block-based programming and computational thinking. Based on the TPACK development path selected, the findings are presented below through the interrelated elements: TK, TPK, and TPACK.

Starting with TK

Biweekly Scratch challenges and written summaries completed by participants were examined to identify any indicators of their TK. TK refers to STEM preservice teachers’ use of block-based programming on Scratch in this study. The levels of the challenges was incremental throughout the semester. That is, while participants were asked to code just a function by using one or two code categories in the first challenge, they coded a full game or story in the last one.

Aligning with the incremental level of the coding challenges, participants' coding skills developed from simple functions and codes to complex animations that include chunks of code sequences. That is, while they just used *motion*, *looks*, and simple *events* blocks on Scratch in the first two challenges, they adeptly used complex *control* blocks, such as *repeat* and *if-then* and advanced event blocks such as *broadcast* in the last two challenges. This shows participants' TK, that is, knowledge of how to program animations using functions of block codes and computational thinking phases developed over the semester.

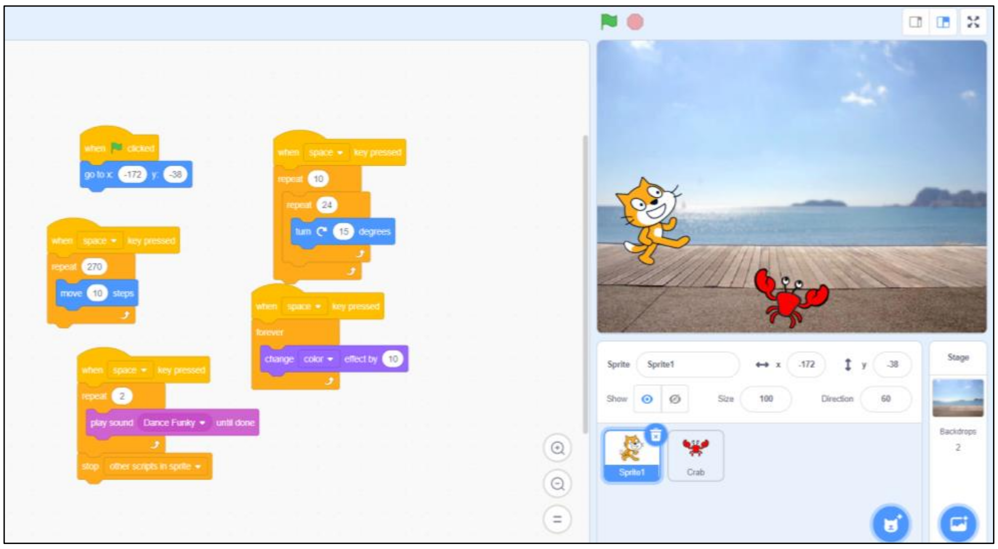
How Participant 14's coding skills developed is shown in Figures 2 to 5. In the first challenge, she used just two basic blocks: "when this sprite clicked" and "say hello for 2 seconds". While the sprites (characters) execute the commands through the codes one by one in the first challenge, Participant 14 was able to code two sprites (the cat and crab) to dance at the same time through implementing *parallelism* of codes in the second challenge. In the third challenge, she designed a number-guessing game by using "if-then" and "repeat" codes under the control category. The final animation she coded was a dashing game in which players had to avoid hitting the obstacles to win. In addition to "repeat" and "if-then" codes, she used the "broadcast-receive" and "show-hide" codes in this game. These findings show that her TK, which was block-based programming in this study, developed from a novice level to a skillful coder who can design complex animations.



```
when this sprite clicked
say (Hi there bat!) for (2) seconds
```

Sprite: Sprite1 x: -160 y: -112 Show Size: 100 Direction: 90

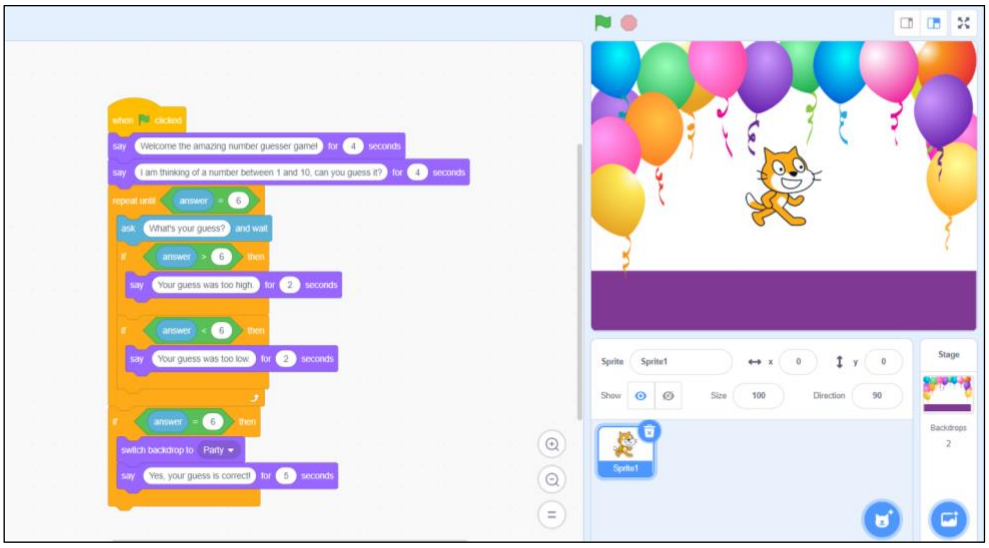
Figure 2. Participant 14's codes for Scratch challenge 1



```
when [flag] clicked
go to x: (-172) y: (-38)

when [space] key pressed
repeat (270)
    move (10) steps

when [space] key pressed
repeat (2)
    play sound [Dance Funky] until done
stop [other scripts in sprite]

when [space] key pressed
repeat (10)
    repeat (24)
        turn ↻ (15) degrees

when [space] key pressed
forever
    change [color] effect by (10)
```

Sprite: Sprite1 x: -172 y: -38 Show Size: 100 Direction: 60

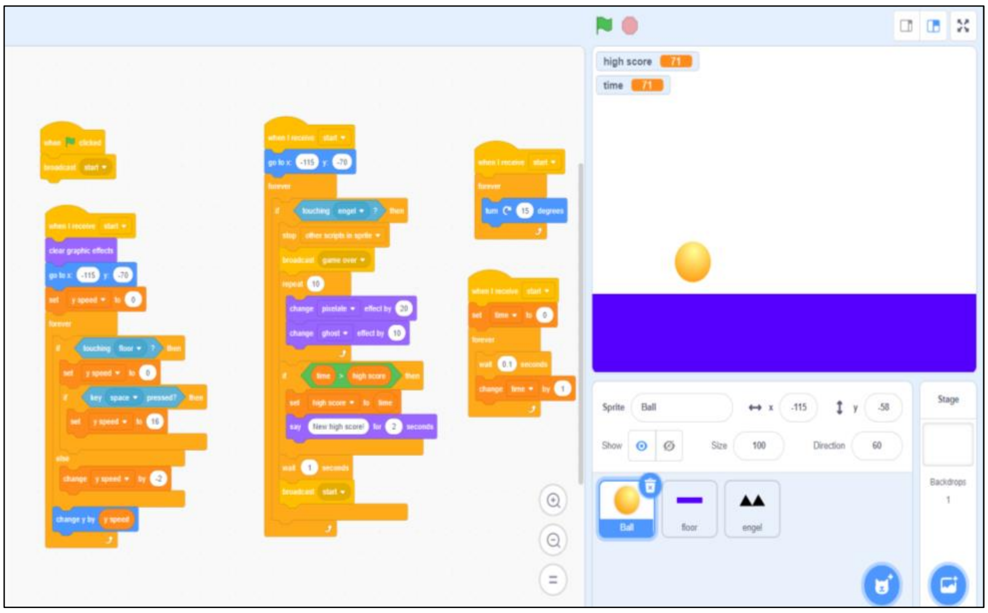
Figure 3. Participant 14's codes for Scratch challenge 2



```
when [flag] clicked
say (Welcome the amazing number guesser game!) for (4) seconds
say (I am thinking of a number between 1 and 10, can you guess it?) for (4) seconds
repeat until <(answer) = (6)>
    ask (What's your guess?) and wait
    if <(answer) > (6)> then
        say (Your guess was too high.) for (2) seconds
    if <(answer) < (6)> then
        say (Your guess was too low.) for (2) seconds
if <(answer) = (6)> then
    switch backdrop to (Party)
    say (Yes, your guess is correct!) for (5) seconds
```

Sprite: Sprite1 x: 0 y: 0
Show Size: 100 Direction: 90
Stage Backdrops 2

Figure 4. Participant 14’s codes for Scratch challenge 3



```
when [flag] clicked
broadcast (start)

when I receive (start)
clear graphic effects
go to x: (-115) y: (-70)
set (y speed) to (0)
forever
    if <touching (floor) ?> then
        set (y speed) to (0)
        if <key (space) pressed?> then
            set (y speed) to (16)
    else
        change (y speed) by (-2)
    change y by (y speed)

when I receive (start)
go to x: (-115) y: (-70)
forever
    if <touching (engel) ?> then
        stop (other scripts in sprite)
        broadcast (game over)
        repeat (10)
            change (pixelate) effect by (20)
            change (ghost) effect by (10)
        if <(time) > (high score)> then
            set (high score) to (time)
            say (New high score!) for (2) seconds
        wait (1) seconds
        broadcast (start)

when I receive (start)
forever
    turn ↻ (15) degrees

when I receive (start)
set (time) to (0)
forever
    wait (0.1) seconds
    change (time) by (1)
```

high score 71
time 71
Sprite: Ball x: -115 y: -58
Show Size: 100 Direction: 60
Ball floor engel
Stage Backdrops 1

Figure 5. Participant 14’s codes for Scratch challenge 4

In addition to coding the challenges on Scratch, participants wrote summaries of how they had coded these and which computational thinking stages they had gone through while coding. Salient observations from the summaries were that participants mostly resorted to decomposition and pattern recognition. As the Scratch challenges became gradually more and more difficult in the course, participants often used the computational thinking strategy of decomposition before they started coding the challenge. That is, they coded the challenges part by part. For instance, Participant 4 described how he divided Challenge 4 (a snake game for him) into small steps:

> After understanding the logic of the Snake game in general, I created a little part of the game. In this part I created, I managed to move my snake and grow by eating the baits. Then, I duplicated baits and modified them to follow each other. In the next phase, I added backgrounds and adjusted the background changes with broadcast. After that, I added music.

The other prevalent computational thinking phase that appeared in participants’ summaries was pattern recognition. Most of the participants referred to how they used the codes in previous challenges for the new ones when the problems in those coding challenges were similar. They were able to identify similarities across the coding challenges and transfer what they learnt to code the following challenges. Some of them also used pattern recognition within a challenge. For instance, Participant 13 was involved in pattern recognition to draw different shapes on Scratch for Challenge 3:

> After drawing the square, I duplicated the code group of the square for the rectangle by using the pattern recognition step because they should have a similar pattern. Then I changed two codes' step numbers with 110 instead of 80 to form the rectangle's long sides. At the end of the code group, I used triangle broadcast to connect the rectangle with the drawing of an equilateral triangle.

The increasing complexity of the animations participants produced and the computational thinking strategies they employed, hint that they became adept at block-based programming and using computational thinking during coding over the semester. It can be inferred that the course module intended for developing participants' TK, that is, Scratch challenges helped these preservice teachers to be experienced coders.

TPK: Synthesising TK with existing PK

PK that participants gained regarding teaching and integrating coding into their future classes was revealed in the reading responses they completed. The articles they read were about bringing coding and computational thinking into K-12 classrooms (Table 3). Example prompts for the reading responses were:

- What do you think about the argument (Robotics and coding teach problem-solving through debugging) in this article (Article 1)?
- Please reflect on the argument briefly.
- Write about what you learnt about pedagogy of computing education from the reading (Article 2).
- In your response, discuss the ways to bring computational thinking and coding into K-12 classrooms by referring to the article you read from a perspective of a future teacher. (Article 3).
- Read the article (Article 4) and evaluate your own learning in this course from the perspective of TPACK. Describe your plans about integrating computer science into your classes. Please refer to the article in your response.

As a response to Article 1, which was about teaching problem-solving through coding and debugging in elementary classrooms, Participant 13 highlighted how collaboration should be encouraged during coding and debugging:

> Debugging allows them to gain the experience of a real-world computer scientist and thus their coding and problem-solving skills are further enhanced. In addition, it is extremely important that children try to solve problems that they cannot solve with their friends. While students help with problems that their friends cannot solve, they both develop their own problem-solving and coding skills and the other student learns different problem-solving ways. Thus, they both strengthen their communication with each other and learn to look at problems from different angles. Thus, students learn different problem-solving ways and methods from each other.

Similarly, Participant 12, who was a math teacher candidate, indicated that creativity and analytical thinking go hand in hand during coding and debugging, so these should be encouraged in elementary classrooms:

> In the example given in the article, students need to understand a comprehensive story and complete the missing parts in that story, find conflicting situations, or solve a problem asked. As far as I understand from the article, the two disciplines (mathematics and coding) feed each other in this aspect. Thus, I know from my department, a child's effort to create a whole story develops creativity and analytical thinking.

For Article 2 which was about activity types for computational thinking, how Participant 10 viewed teaching programming was indicated in her response below. She suggested that inquiry-based and collaborative learning processes be promoted either through unplugged (activities done without using computers) or computer-based coding activities:

> Combining inquiry-based learning and computational thinking emphasizes the student's role in the learning process. Coding is one of the ways to develop numerical thinking skills. However, technology is not always necessary to develop numerical thinking skills. Unplugged activities can be implemented without the use of computers. It is collaborative most of the time and this may inspire a students' perceptions about the nature of computer science. It can be taught in any location hence it becomes interesting for everyone. Teachers should also allow pupils for experimentation, discovery, using ungraded problems and they have to make it clear that making mistakes is part of learning process especially in programming.

Notions that group work during coding should be encouraged and that creativity is needed for programming were also revealed in Participant 3's response to Article 3 which was about use of Bee-bots in kindergarten classes:

> Forming groups and starting from the basics are important for children to understand what they are doing. For exploration, Bee-bot can be on stage and children in groups can figure out the system. Guiding children and peer relations support their learning process. To conclude, sometimes I scare and I cannot believe myself to implement in my classrooms; but, this is a new way for me, and I will gain experience and also learn with children by doing as well as through trial-and-error. These computational activities give children freedom to think and chance to create something. I think this fits our philosophy about early childhood education and as a preschool teacher I just need to learn how to implement this to kindergarten classes. I will be a preschool teacher, so I will generally prefer unplugged activities, then I can use making and tinkering. For example, blocks and roads can be useful to apply computational skills; thus, they can understand the basics of computational skills such as decomposition, algorithms, pattern recognition, and abstraction. Also, peer interaction is important to support them. In every experience, we should offer peer interaction, and small groups to work in are important. As a teacher, supporting and guiding them should be required, it should not be teacher-directed.

The most salient perspective expressed in the responses to Article 4 was to start computer science education at early grades as it supports problem solving. For instance, Participant 2 wrote:

> If students are encouraged to think through algorithms, they can easily determine the steps to solve any problem on their own in daily and school life. Especially in the early years of life, the growth of the brain is so fast and the changings are mostly permanent in the brain. In this regard, especially preschoolers should be exposed to computational thinking practices much more than everyone.

Participant 9 added: "In today's changing world, I think computational thinking skills can bring convenience to our lives and problem-solving skills can be acquired more easily if computer science education starts at an early age." To highlight the viability of offering computer science education at early grades, Participant 15 also pointed out that "teachers can teach computational concepts in K-12 classrooms without using computers, and computational thinking can be applied to all disciplines".

Participants listed the following pedagogical guidelines to be followed for teaching computer science in their future classes: (1) the promotion of collaborative work and creativity during coding activities, (2) the need for problem/ inquiry-based learning contexts for programming to promote analytical and critical thinking, and (3) the implementation of both unplugged and computer-based activities in any discipline to teach computational thinking. How these pedagogical guidelines were implemented in the peer-teaching sessions is discussed in the following section.

TPACK: Incorporating TPK into CK

By the time of the final project, participants had completed the Scratch challenges and reading responses (Table 2). That is, they had TPK for block-based programming and computational thinking when they started the final project. For the final project, participants designed an online lesson in which they integrated coding into their own subject area. They also peer-taught their lessons on a video-conferencing platform during live class sessions. The lesson plans indicated how their TPK interacted with their CK. Effective use of block-based coding to prepare subject-area materials for their STEM lessons, appropriate implementation of computer science pedagogy principles in their lessons, and addressing learning objectives of both subject area and computer science topics in an aligned way were salient themes that emerged from the lesson plans.

Effective use of block-based coding to prepare subject-area materials

Participants 5 and 12 worked together to design their lesson about patterns in math for 5th graders. They created their materials to present the topic interactively as a warm-up activity on Scratch (Figure 6). They used the "question-answer" function on Scratch to encourage interaction and participation during their peer-teaching session. The artifact in Figure 3 shows how TPACK emerged in their lesson plan of Participants 5 and 12. Appropriate use of the "question-answer" function on Scratch (TK) to design interactive and inquiry-based (PK) tasks to teach patterns in math (CK), illustrates the participants' TPACK.

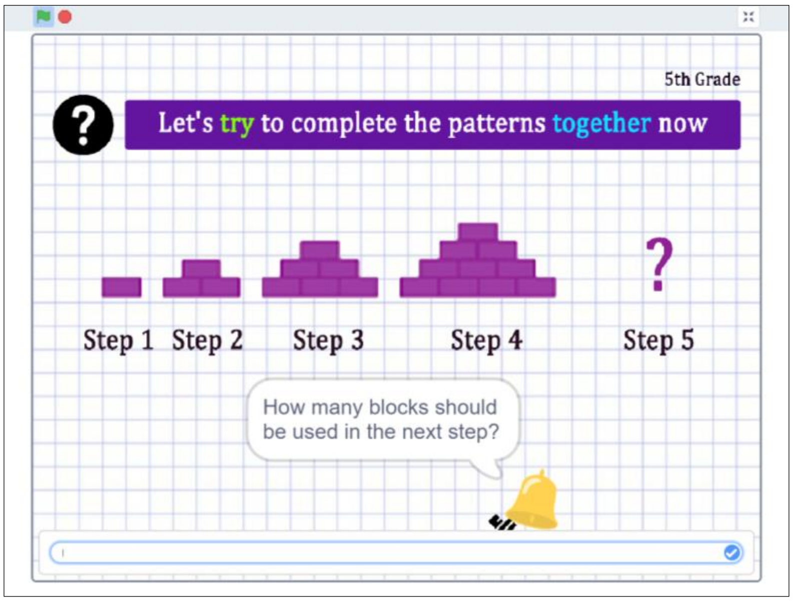

5th Grade
Let's try to complete the patterns together now
Step 1
Step 2
Step 3
Step 4
?
Step 5
How many blocks should be used in the next step?

Figure 6. Warm-up activities prepared by Participants 5 and 12

In another lesson plan, Participants 2, 13, and 15 together created a story by modifying Little Red Riding Hood to introduce shapes to 2nd graders. They coded their version of the story on Scratch and used the story as a hook in the beginning of their peer-teaching session. They added the shapes in shown in Figure 7 as obstacles that Little Red Riding Hood encountered while going to her grandma's house.

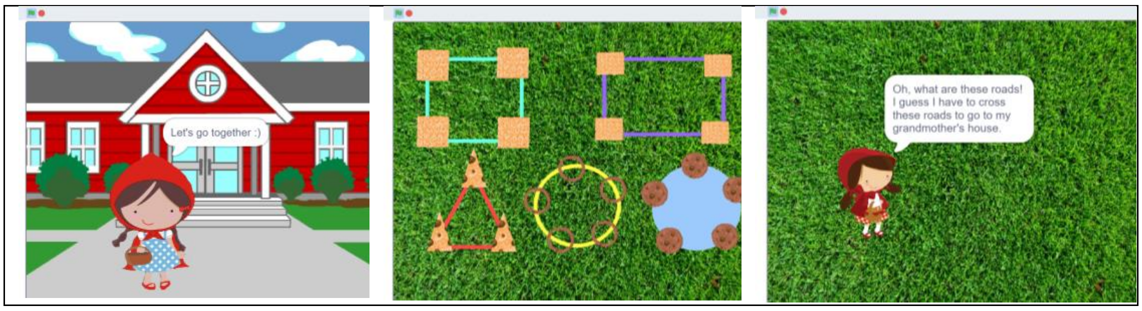

Let's go together :)
Oh, what are these roads! I guess I have to cross these roads to go to my grandmother's house.

Figure 7. Story presentations prepared by Participants 2, 13, and 15

How these participants' TPACK was revealed in the animation they coded can be explained through the three main knowledge types in the framework. In terms of TK, they coded the whole story and added some interactive elements into it. As for PK, they gave their students a problem to solve within a story-based learning environment, that is, the girl had to cross the obstacles to reach the house. Also, they assigned each shape to a group to encourage collaboration during programming. When it came to CK, they used the story as a context to introduce shapes in math to 2nd graders.

Appropriate implementation of computer science pedagogy principles
In all of the lesson plans, remixing activities and group work were appropriately designed and implemented during the peer-teaching sessions. Participants in this study got familiar with such pedagogical strategies (PK for computer science) when they completed the reading responses earlier in the course. Remixing refers to when the participants, during their peer-teaching sessions, provided initial codes and then asked their students to modify the codes to program other commands. For instance, Participants 1, 3, and 14 prepared a science lesson about recycling for 4th graders. The recycling categories were plastic, glass, paper, and metal (Figure 8). When learners put the material to the right recycling bin, the animation showed that their attempt was correct. To illustrate, when they put a glass bottle to the recycling bin for glass, the animation showed a green tick (Figure 8).

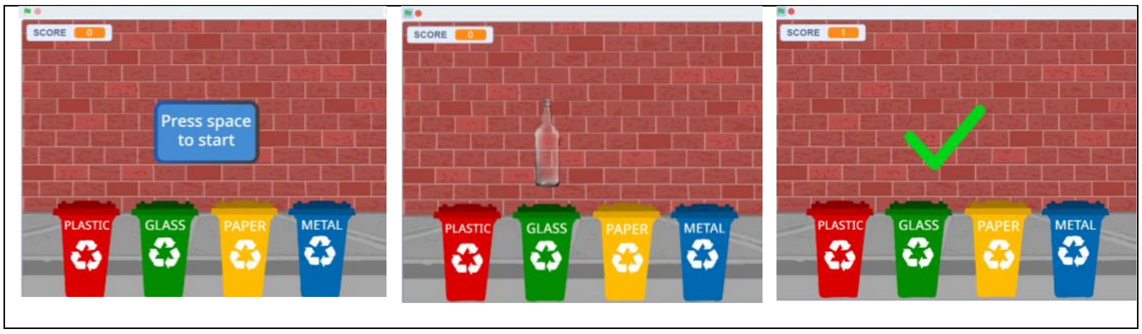


Figure 8. Animations coded by Participants 1, 3, and 14

Participants 1, 3, and 14 gave the code for the glass bottle to the students during their peer-teaching. Yet, they did not code all the materials that they included in the animation, such as a plastic bottle (Figure 9). And, they asked their students to code those materials by remixing the codes they had prepared. Use of group work was also extensive in all of the lesson plans. For instance, Participants 1, 3, and 14 asked their students to complete the coding tasks in Figure 10 in groups. The lesson they designed included TPACK indicators. They coded an animation on Scratch (TK) for their lesson which was about recycling (CK). They also asked their students to complete remixing tasks in groups (PK) in the peer-teaching session.

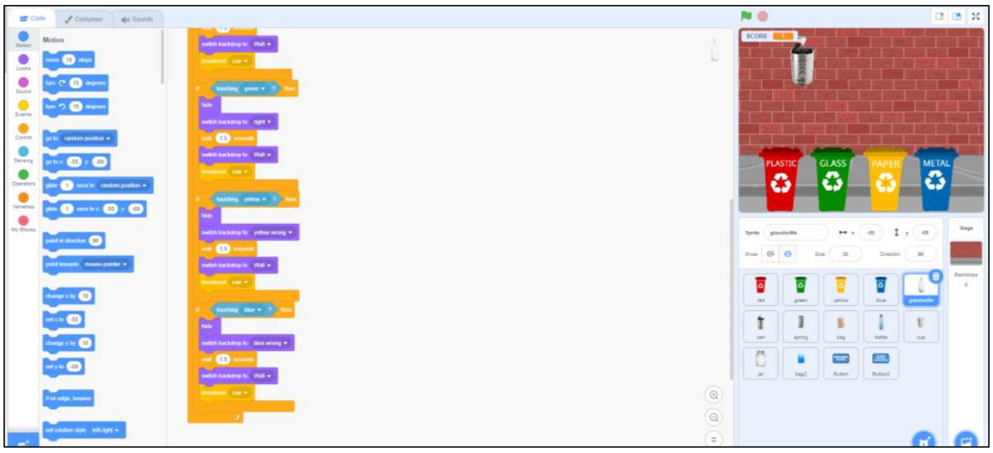
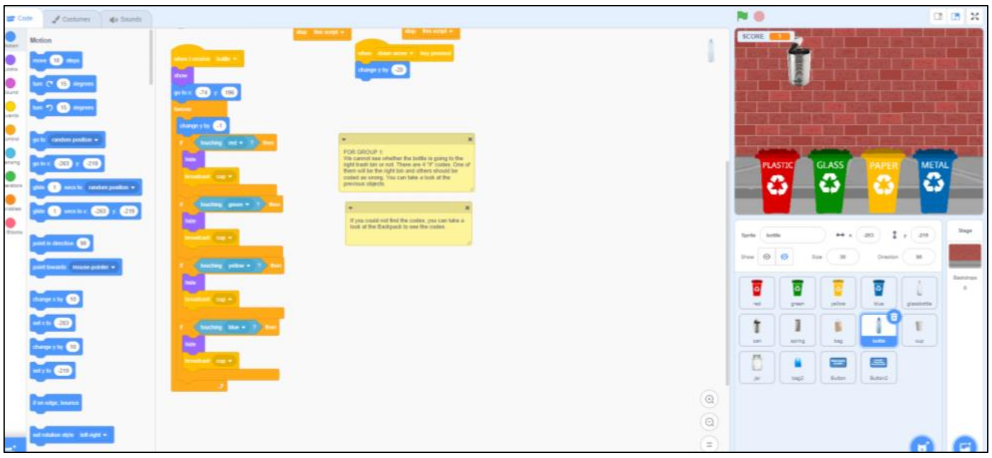

```
    switch backdrop to Wall
    broadcast can
if touching green ? then
    hide
    switch backdrop to right
    wait 1.5 seconds
    switch backdrop to Wall
    broadcast can
if touching yellow ? then
    hide
    switch backdrop to yellow wrong
    wait 1.5 seconds
    switch backdrop to Wall
    broadcast can
if touching blue ? then
    hide
    switch backdrop to blue wrong
    wait 1.5 seconds
    switch backdrop to Wall
    broadcast can
```

Figure 9. Codes for the recycling activity of Participants 1, 3, and 14


FOR GROUP 1:
We cannot see whether the bottle is going to the right trash bin or not. There are 4 "if" codes. One of them will be the right bin and others should be coded as wrong. You can take a look at the previous objects.

If you could not find the codes, you can take a look at the Backpack to see the codes.

Figure 10. The recycling task prepared by Participants 1, 3, and 14

Addressing learning objectives of both subject area topics and programming

When the learning objectives and activities in the lesson plans were examined, it was clear that the STEM teacher candidates aimed to address learning objectives of both subject area topics and block-based programming. For instance, for their lesson which was about geometric shapes, Participants 2, 13, and 15 wrote four learning objectives: Students will be able to (1) classify geometric shapes (triangle, square, rectangle, ring, and circle) according to the number of edges and corners, (2) compare similarities or differences of triangle, square, rectangle, ring and circle, (3) create structures by using shape models and draw the structures they create on Scratch, and (4) use "broadcasting" and "touching" codes in Scratch blocks. Participants 7, 9, and 10 prepared a lesson about the periodic table for a chemistry class for 7th graders. They integrated a variety of Web 2.0 tools into their lesson to implement online quizzes and games to introduce and review the subject area topic. They used web 2.0 tools to explore or assess the subject area topic in depth. They also used Scratch to achieve the learning objectives they identified for block-based programming. These examples indicate that participants successfully incorporated TPK into CK in their lesson plans.

Discussion and conclusion

Teacher preparation for computer science education particularly in subject area classes has been rising in prominence in the context of twenty-first century classrooms where technology is ubiquitous. Although the TPACK framework has extensively guided STEM teacher education research (e.g. Deng et al., 2017; Kontkanen et al., 2016; Mourlam et al., 2021; Oner, 2020), there is little research on how TPACK could be used in the settings where computer science education is to be incorporated into subject area classes (Mouza et al., 2017; Umutlu, 2021). With the Covid-19 pandemic, how preservice teachers' TPACK can be enhanced in fully-online educational technology courses has also come into play in the field of teacher education. In this study, an online educational technology course design was presented, and how preservice STEM teachers' learning experiences of block-based programming and computational thinking unfolded within the course was examined based on the TPACK framework.

The instructional design model of the course was conceptualised by using the TPACK framework and literature of online learning communities and computing education. All the sessions and activities were designed following the guidelines derived from the literature (e.g. Fiock, 2020; Wang, 2021; Yuan & Kim, 2014). When it came to how the course presented in this paper was implemented online to create learning communities, it is important to note that keeping synchronous and asynchronous tasks and individual and group activities in balance was useful to develop preservice teachers' TPACK for computer science education (Ioannou & Angeli, 2015; Mouza et al., 2017). Within the course, preservice teachers completed several individual and group tasks. The individual tasks were usually completed asynchronously whereas the groups ones were executed synchronously. For instance, each preservice teacher wrote their own reading responses individually before class time and then discussed their viewpoints with their peers in groups during live class time. Scratch challenges were completed in a similar manner. In online learning environments, communication and interaction may be limited outside of class time as students are separated by distance and time (Moore, 1989). Thus, it is important to let students discuss their ideas with their classmates when they come together for a synchronous session. In this way, students become more socially present and develop their sense of belonging in the online community (Fiock, 2020; Garrison, 2000; Yuan & Kim, 2014). Within the instructional design model of the course, in particular the when and how of online tasks (Yuan & Kim, 2014) was kept in balance to build up an effective learning community.

Another aspect of the preservice teachers' learning experiences was that they were able to develop and integrate their TPACK successfully, aligning with the identified path (from TPK to TPACK) (Koehler et al., 2014). Findings of this study revealed several TPACK indicators in preservice teachers' artifacts. For example, as biweekly coding challenges were assigned to preservice teachers, they had a chance to practice programming. They had hands-on authentic experiences (Dunlap & Lowenthal, 2018; Fiock, 2020; Morrison, 2017; Stephens & Roberts, 2017) of programming on Scratch with the challenges that incrementally became more difficult to code (Savenye & Hong, 2017). Through these coding challenges, preservice teachers developed their programming skills from a novice-level to an expert-level. It can be concluded from the findings that providing hands-on authentic coding activities can help preservice teachers who are inexperienced coders to improve their TK, the block-based programming within the scope of this study.

With regard to PK, it can be suggested that preservice teachers' awareness of pedagogical approaches and practices in computing education was supported through the assigned articles and the related prompts given. Reading the articles and writing reflective responses about them seem to have scaffolded preservice teachers' learning how to teach programming as they had an opportunity to examine real classroom implementations of programming or computational thinking activities in the articles (Mouza et al., 2017). As the three main components of the TPACK framework are not isolated entities (Koehler et al., 2013; Niess, 2011), it is important to create contexts where preservice teachers can put their TPK into practice in their content area classes. The final project of the course presented in this paper was lesson planning and peer-teaching it during synchronous classes. Findings from the analysis of lessons plans demonstrated that preservice teachers skillfully used the programming skills they gained in the Scratch challenges over the semester to code programs for their lessons. Probably drawing pedagogical ideas from the articles they read, they mostly implemented remixing activities as group work in their content area lessons during peer teaching (Kotsopoulos et al., 2017; NCCE, 2020). They were also able to address learning objectives of both content area and computer science teaching, which indicates appropriate incorporation of TPK into CK. Lesson planning and peer-teaching seem to have enabled preservice teachers to integrate their CK with their PK about computer science education, and TK about block-based programming (Ioannou & Angeli, 2015; Mouza et al., 2017).

Although the online course design implemented in this study was just a prototype, findings from the study may have important implications given that effects of the pandemic still continue. Fully-online or blended learning environments will be needed for teacher education in a future which will require post-pandemic teaching approaches (Murphy, 2020). The instructional design model of the online educational technology course may provide guidance about how to design computer science courses in teacher education curricula. Teacher educators can also adapt it to face-to-face classes (Umutlu, 2021). The online course design proposed here can serve as a foundation of future interventions on how to improve preservice teachers' TPACK for computer science education in STEM classes. Based on the TPACK indicators identified in this study, it is suggested that designing hands-on practice tasks for programming, balancing individual and group tasks while teaching programming, introducing pedagogies that might be specific to computer science education, and making connections with teacher candidates' content areas are effective approaches to be followed by teacher educators.

The study should be examined within its limitations. In this study, there was no data regarding how the coding challenges were completed asynchronously, although the researcher could observe small discussions about the challenges and take notes during synchronous classes. As higher education institutions in the world have adopted different means to design online education environments after the Covid-19 outbreak, the instructional design model used in the study may not be implemented as is in other contexts. Adjustments may be needed for different settings. Considering these limitations, it is recommended that future research focus on exploring a wider variety of ID models for online computer science education for preservice teachers from different subject areas. Additionally, participants could be asked to record their computer screens while completing individual asynchronous coding activities. In this way, videos of how they code would be an available data source to examine their TK more holistically.

References

Caskurlu, S., Yadav, A., & Santo, R. (2021). Preparing teachers for computational thinking integration in K-12: A meta-aggregation. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 1317-1317. https://doi.org/10.1145/3408877.3439639

Chai, C-S., Koh, J. H-L., & Tsai, C-C. (2013). A review of technological pedagogical content knowledge. *Educational Technology & Society*, *16*(2), 31–51.

Deng, F., Chai, C. S., So, H.-J., Qian, Y., & Chen, L. (2017). Examining the validity of the technological pedagogical content knowledge (TPACK) framework for preservice chemistry teachers. *Australasian Journal of Educational Technology*, *33*(3), 1-14. https://doi.org/10.14742/ajet.3508

Dunlap, J. C., & Lowenthal, P. R. (2018). Online educators' recommendations for teaching online: Crowdsourcing in action. *Open Praxis*, *10*(1), 79-89. https://openpraxis.org/index.php/OpenPraxis/article/view/721/421

Fiock, H. (2020). Designing a community of inquiry in online courses. *The International Review of Research in Open and Distributed Learning*, *21*(1), 135-153. https://doi.org/10.19173/irrodl.v20i5.3985

Garrison, D. R., Anderson, T., & Archer, W. (2000). Critical ınquiry in a text-based environment: Computer conferencing in higher education. *The Internet and Higher Education*, *2*(2-3), 87-105. https://doi.org/10.1016/s1096-7516(00)00016-6

Grover, S., Fisler, K., Lee, I., & Yadav, A. (2020). Integrating computing and computational thinking into K-12 STEM learning. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, Portland, OR, 481-482. https://doi.org/10.1145/3328778.3366970

Hubbard, A. (2018). Pedagogical content knowledge in computing education: A review of the research literature. *Computer Science Education*, *28*(2), 117-135. https://doi.org/10.1080/08993408.2018.1509580

Hyett, N., Kenny, A., & Dickson-Swift, V. (2014). Methodology or method? A critical review of qualitative case study reports. *International Journal of Qualitative Studies on Health and Well-Being*, *9*(1), 23606. https://doi.org/10.3402/qhw.v9.23606

Ioannou, I., & Angeli, C. (2015). Technological pedagogical content knowledge as a framework for integrating educational technology in the teaching of computer science. In C. Angeli, & N. Valanides (Eds.), *Technological pedagogical content knowledge: Exploring, developing and assessing TPCK* (pp. 225-237). Springer. https://doi.org/10.1007/978-1-4899-8080-9_11

Jin, M. (2022). Preservice teachers' online teaching experiences during COVID-19. *Early Childhood Education Journal*. https://doi.org/10.1007/s10643-022-01316-3

Johnson, N., Veletsianos, G., & Seaman, J. (2020). US faculty and administrators' experiences and approaches in the early weeks of the COVID-19 pandemic. *Online Learning*, *24*(2), 6-21. https://doi.org/10.24059/olj.v24i2.2285

Koehler, M. J., & Mishra, P. (2009). What is technological pedagogical content knowledge? *Contemporary Issues in Technology and Teacher Education*, *9*(1), 60-70. https://citejournal.org/volume-9/issue-1-09/general/what-is-technological-pedagogicalcontent-knowledge

Koehler, M. J., Mishra, P., Akcaoglu, M., & Rosenberg, J. M. (2013). Technological pedagogical content knowledge for teachers and teacher educators. In N. Bharati, & S. Mishra (Eds.), *ICT integrated teacher education: A resource book* (pp. 1-8). Commonwealth Educational Media Center for Asia.

Koehler, M. J., Mishra, P., Kereluik, K., Shin, T. S., & Graham, C. R. (2014). The technological pedagogical content knowledge framework. In J. M. Spector, M. D. Merrill, J. Elen, & M. J. Bishop (Eds.), *Handbook of research on educational communications and technology* (4th ed., pp. 101–111). Springer.

Kontkanen, S., Dillon, P., Valtonen, T., Renkola, S., Vesisenaho, M., & Väisänen, P. (2016). Pre-service teachers' experiences of ICT in daily life and in educational contexts and their proto-technological pedagogical knowledge. *Education and Information technologies*, *21*(4), 919-943. https://doi.org/10.1007/s10639-014-9361-5

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*(2), 154-171. https://doi.org/10.1007/s40751-017-0031-2

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press. https://doi.org/10.1017/CBO9780511815355

Margulieux, L., & Yadav, A. (2020). Middle science computing integration with preservice teachers. *Society for Information Technology & Teacher Education International Conference*, 63-72. Association for the Advancement of Computing in Education (AACE). https://www.learntechlib.org/primary/p/215731/

Mayring, P. (2000). Qualitative content analysis. *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, *1*(2), Art. 20. https://doi.org/10.17169/fqs-1.2.1089

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, *108*(6), 1017–1054. https://www.tcrecord.org/Content.asp?ContentId=12516

Moore, M. G. (1989). Editorial: Three types of interaction. *American Journal of Distance Education*, *3*(2), 1-7. https://doi.org/10.1080/08923648909526659

Morrison, G. R. (2017). Activities. In A. A. Piña (Ed.), *Instructional design standards for distance learning* (pp. 47-53). Association for Educational Communications and Technology.

Mourlam, D., Chesnut, S., & Bleecker, H. (2021). Exploring preservice teacher self-reported and enacted TPACK after participating in a learning activity types short course. *Australasian Journal of Educational Technology*, *37*(3), 152-169. https://doi.org/10.14742/ajet.6310

Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, *33*(3), 61-76. https://doi.org/10.14742/ajet.3521

Murphy, M. P. A. (2020). COVID-19 and emergency e-learning: Consequences of the securitization of higher education for post-pandemic pedagogy. *Contemporary Security Policy*, *41*(3), 492-505. https://doi.org/10.1080/13523260.2020.1761749

National Centre for Computing Education (2020). How we teach computing: 12 pedagogy principles. https://teachcomputing.org/pedagogy/

Niess, M. (2011). Investigating TPACK: Knowledge growth in teaching with technology. *Journal of Educational Computing Research*, *44*(3), 299–317. https://doi.org/10.2190/ec.44.3.c

Oner, D. (2020). A virtual internship for developing technological pedagogical content knowledge. *Australasian Journal of Educational Technology*, *36*(2), 27-42. https://doi.org/10.14742/ajet.5192

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*(2), 1-11.

Patton, M. Q. (1990). *Qualitative evaluation and research methods* (2nd ed.). Sage Publications.

Rich, K. M., Yadav, A., & Larimore, R. A. (2020). Teacher implementation profiles for integrating computational thinking into elementary mathematics and science instruction. *Education and Information Technologies*, *25*(4), 3161-3188. https://doi.org/10.1007/s10639-020-10115-5

Savenye, W. C., & Hong, Y. (2017). Sequence. In A. A. Piña (Ed.), *Instructional design standards for distance learning* (pp. 33-46). Association for Educational Communications and Technology.

Stake, R. E. (1995). *The art of case study research*. Sage.

Stephens, G. E., & Roberts, K. L. (2017). Facilitating collaboration in online groups. *Journal of Educators Online*, *14*(1), 1-16. https://eric.ed.gov/?id=EJ1133614

Stiller, K. D., & Bachmaier, R. (2017). Dropout in an online training for trainee teachers. *European Journal of Open, Distance and E-learning*, *20*(1), 80-94. https://doi.org/10.1515/eurodl-2017-0005

Thomas, G. (2011). A typology for the case study in social science following a review of definition, discourse, and structure. *Qualitative Inquiry*, *17*(6), 511–521. https://doi.org/10.1177/1077800411409884

Thomas, M. (2020). Virtual teaching in the time of COVID-19: Rethinking our WEIRD pedagogical commitments to teacher education. *Frontiers in Education*. https://doi.org/10.3389/feduc.2020.595574

Tracy, S. J. (2020). *Qualitative research methods: Collecting evidence, crafting analysis, communicating impact* (2nd ed.). John Wiley & Sons.

Umutlu, D. (2021). An exploratory study of pre-service teachers' computational thinking and programming skills. *Journal of Research on Technology in Education*. https://doi.org/10.1080/15391523.2021.1922105

Wang, C. X. (2021). CAFE: An instructional design model to assist K-12 teachers to teach remotely during and beyond the Covid-19 pandemic. *TechTrends 65*, 8-16. https://doi.org/10.1007/s11528-020-00555-8

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017).Computational thinking in teacher education. In P. Rich, & C. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205-220). Springer. https://doi.org/10.1007/978-3-319-52691-1_13

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, *60*(6), 565-568. https://doi.org/10.1007/s11528-016-0087-7

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55-62. https://doi.org/10.1145/2994591

Yuan, J., & Kim, C. (2014). Guidelines for facilitating the development of learning communities in online courses. *Journal of Computer Assisted Learning*, *30*(3), 220-232. https://doi.org/10.1111/jcal.12042

Corresponding author: Duygu Umutlu, duygu.umutlu@boun.edu.tr

Please cite as: Umutlu, D. (2022). TPACK leveraged: A redesigned online educational technology course for STEM preservice teachers. *Australasian Journal of Educational Technology, 38*(3), 104-121. https://doi.org/10.14742/ajet.4773