Cocode: Providing Social Presence with Co-learner Screen Sharing in Online Programming Classes

JEONGMIN BYUN, JUNGKOOK PARK, and ALICE OH, KAIST, South Korea

Social presence is known to be important for distance education, and a common approach in online classes is to provide chat boxes and forums to provide the social presence. In such a class, however, learners must explicitly act beyond their normal learning activities, so often there is no social presence in the class even when there are several learners working on the same course material. In this paper, we develop an approach where learners can share the social presence without any explicit action; their normal learning activities would be used to provide visual cues for social presence. We present Cocode, a system designed for an online programming class that shows other learners' code editors and running output in the programming environment with minimum privacy issues. For evaluation, we ran two user studies with groups of participants who took an offline class and an online programming class from the university; results from the studies showed that learners felt less social presence in Cocode than in offline classes, but they felt significantly more social presence in Cocode than in online classes with live video lectures, forums, and chat sessions.

CCS Concepts: • **Social and professional topics** → **Computing education**; **Computer science education**.

Additional Key Words and Phrases: education; programming education; distance learning; social presence

ACM Reference Format:
Jeongmin Byun, Jungkook Park, and Alice Oh. 2021. Cocode: Providing Social Presence with Co-learner Screen Sharing in Online Programming Classes. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 300 (October 2021), 28 pages. https://doi.org/10.1145/3476041

1 INTRODUCTION

Learners taking online programming classes often work on learning materials by themselves from home, but previous research has found that the social presence of the instructor and other learners is essential in remote learning [33, 40]. Existing solutions for providing a social presence for online learners focus on either using rich communication mediums such as video chat [10] or introducing traditional web applications for communication like forums and comments [1, 51, 54].

In offline classes, even when learners do not intentionally communicate with other learners, they give each other social presence cues by visually showing themselves working on the learning material. Using a rich medium like video chats or screen sharing can simulate social presence in an online environment. However, video chats or screen sharing require heavy network bandwidth and large server systems [4], and the participation rate is low for reasons such as privacy issues [16]. On the other hand, traditional text-based solutions like forums and comments require learners to additionally learn how to use them to communicate with others effectively, and explicitly invest

Authors' address: Jeongmin Byun, jmbyun@kaist.ac.kr; Jungkook Park, pjknkda@kaist.ac.kr; Alice Oh, alice.oh@kaist.edu, KAIST, Daehak-ro 291, Daejeon, South Korea.

2573-0142/2021/10-ART300 $15.00
https://doi.org/10.1145/3476041

their time to deliver their presence to others [28]. As a result, it is hard to feel a social presence from other learners in online classes.

However, in online *programming* classes, learner activities captured in the programming environment can be used as visual cues for a social presence; this is similar to how screen-sharing can be used for social presence. When learners use a web-based programming environment to work on their exercises, raw code contents and running output data are available to the system. Therefore, we can easily record, process, and anonymize learners' activity logs to store on the server, and visualize this information to the other learners to provide a social presence in their working environment. This reduces potential privacy issues and requires less network and system resources, compared to the video-based medium used in online classes for social presence.

We built a prototype system based on this idea that provides social presence to learners by sharing their co-learners' activities in online programming classes. The system, Cocode, collects and stores the learning activities in learners' code editors and running environments; and then, Cocode displays these collected activities in the programming environments of new learners. Learners can easily see what other learners have done in their programming environment, and compare the behavior and progress of co-learners with their own behavior and progress. We expected this feature to asynchronously help learners to feel that they are learning with their co-learners whenever they work on the online programming exercises.

To evaluate Cocode, we conducted two user studies to see if the participants used the co-learner screens, and also answer the following research questions:

- RQ1) Does Cocode with co-learner screens provide more social presence than an offline programming course in physical classrooms?
- RQ2) Does Cocode with co-learner screens provide more social presence than an online programming course with existing social features?

The user studies were conducted before and after the start of the COVID-19 pandemic; during the pandemic, all offline classes in our university, KAIST (Korea Advanced Institute of Science and Technology), moved entirely to online classes. This allowed us to gauge various opinions about Cocode from two groups of participants; ones who took *Introduction to Programming* class (CS1 class) in the university with offline social activities like lab sessions and pair programming sessions, and others who took the same programming class run entirely online with the same learning materials.

The user studies showed that the participants felt a less social presence in Cocode than in the offline CS1 class, but more social presence than in the online CS1 class. We found that 64% of the participants who took the online CS1 class in the university actively read other learners' code contents in the co-learner screens, and the answers to the survey showed that all participants in the study felt significantly more social presence regarding their co-learners' context, emotions, and the reality of their presence in Cocode than in the online class.

The main contributions from this work are as follows:

- We present Cocode, a novel system for online programming classes that shares co-learner social presence by showing other learners' activities from their code editors and output screens in each learner's screen. The system asynchronously creates a social presence from the past activity, so that learners can feel this presence whenever they use Cocode.
- We evaluated Cocode through controlled studies with two groups of real university students who took either an online version and an offline version of the same course. The results showed the positive effects of Cocode on learners in an online class and possible opportunities for improvements.

All of the studies in this research project were approved by KAIST IRB office under the condition that all participants must be completely anonymous to each other.

The source code[1] and the interactive demo[2] of Cocode are publicly available on the websites.

2 RELATED WORK

2.1 Social Presence

Previous research tells us that social needs for *online* teaching and learning are not different from traditional in-class learning [33, 40], which means that online learners also need social interactions since they are essential for learning. Garrison et al. [18] describe three core elements of interactions essential to learners according to the Community of Inquiry model: social presence, teaching presence, and cognitive presence. While teaching presence and cognitive presence can be established with carefully designed learning materials and instructions, forming social presence can be relatively tricky in online learning contexts.

Prior research defines social presence as "the degree of awareness of another person in an interaction and the consequent appreciation of an interpersonal relationship" [46], but also defines as the ability of participants in a community of inquiry to project themselves socially and emotionally as *real* people [18]. According to this research, the learners would feel a strong social presence by having interpersonal relationships based on social values like trust and respect; however, the minimum amount of social presence can occur when a person's sensory experience indicates the presence of another intelligence [5]. In offline classes, even when learners do not explicitly communicate with their co-learners, there are many reasons to believe that other learners are real people. Learners form social presence based on visual cues without any activities since the learners can all see each other in the learning environment. However, when learners only use text-based medium to communicate in the course, they never actually *see* each other. This makes a social presence in online learning classes depend entirely on an individual's ability to signal one's state in a virtual environment to the others [28]. According to this research, novice learners do not have this ability. Additionally, if social presence is missing in the learner experience, it may cause learner frustration, anxiety, and reduced engagement [1, 24] and require more self-regulated learning skills from the learners since peer support is missing [2, 3]. Therefore many researchers have proposed various methods to provide increased social presence to online learners [1, 39, 47].

In this research, we propose Cocode, a system that allows online learners to form social presence based on the visual cues from their programming environment in the web pages. Learners do not have to take explicit actions to form social presence, and this helps resolve common problems in previously proposed systems [1, 28]. Cocode collects and saves the learners' activity logs from their normal learning behavior, and visualizes them to the learners who work on the same learning material later to provide an asynchronous social presence.

2.2 Discussion Forums and Comments

Forums, Q&As, and comments are popular solutions for providing social presence to online learners. Forums allow learners to discuss specific topics with their co-learners, and Q&As allow learners to ask questions to the course staff or their co-learners about course-related issues. Comments allow learners to share their opinions about the articles in forums, Q&As, or the class materials. Studies have found that the proper selection of social tools can facilitate the community and may help learners engage in the class [1, 51, 54]. Learners are more likely to remain in a when educators give appropriate and timely feedback to the learners [26], or when the learners participate more

[1] https://github.com/jmbyun/cocode
[2] https://jmbyun.github.io/cocode-demo

actively in the discussion forum [36]. However, many learners are not familiar with these tools, so they must first learn the skills to use the tools for communication in class [28]. Even when they can use these tools, many learners are likely to not participate in the discussions in the online courses. Prior research showed that less than 2% of the learners in online classes with forums are actually posting articles to the forum and forming social presence with the others [51].

On the other hand, Cocode does not require learners to learn specific skills, or intentionally participate in a discussion to share their social presence. Cocode allows learners to form social presence through visual cues created from their learning activities by merely working on their hands-on programming exercises; thus, all learners who are willing to share their social presence with others can contribute to forming a stronger social learning environment.

2.3 Synchronous Video Discussions

The main problem of social presence in online learning is that *"you don't see someone"* in class [10], so previous research approached it by employing synchronous video discussions in online courses. Video discussions provide a robust social presence by allowing learners to benefit from both verbal and non-verbal social cues from the co-learners, which helps them develop feelings of trust [10, 31]. This allows learners to have a more collaborative learning experience, and it comes with many benefits in terms of cognitive, meta-cognitive, and social impact [30]. However, video conferencing comes with technical hurdles such as insufficient bandwidth and high network latency, making such systems impractical for many learners [27]. It may also be hard to find a discussion partner unless there is a synchronous discussion session; and it was shown that the frustration from failed attempts to conduct a video discussion in a class had substantial negative effects in learning [16].

Overcoming the problems of the various proposed solutions, Cocode provides visual cues for co-learner social presence via a text-based approach. This allows Cocode to create social presence with smaller network bandwidth (screen sharing in Zoom requires 50-75Kbps per screen while Cocode requires under 1Kbps [11]). Cocode also causes fewer privacy concerns since learners do not have to show their face and their code contents are not visible to other learners until they explicitly request to see the contents.

2.4 Collaborative Working Environment

Other researchers have developed tools that allow users to share their working environment for collaborative tasks. The tools usually allow multiple users to see the same contents on their computer screens and work on the same tasks together [37]. These tools, especially the tools for distributed pair programming, give users a social presence from their collaborators. They synchronize the contents shown to the various users by sending small snippets of data and event logs to the others. This is relatively network-resource-efficient compared to the existing screen-sharing implementations and is similar to how Cocode is more resource-efficient than video conferencing systems.

In the field of collaborative software development, researchers have developed distributed software development environments that support the social awareness of their co-workers [45]. Multiple studies have introduced distributed pair programming environments and found that the code written within the environments is of a higher quality than the code written by a single programmer [14, 15, 22]. To support communication between the partners, they shared users' current cursor location and latest activities [42, 52], or used eye-tracking and gaze awareness visualization to share what a user was focusing on at the time [12].

However, these research examples were all designed to provide awareness or presence to the programmers when multiple people were working together on the same task. The tools from the examples require users to explicitly communicate and interact with the other users. On the other hand, learners in Cocode do not collaboratively work with other learners but, instead, solve their

own programming exercises individually in the system, in an asynchronous manner. The design of the co-learner screens in Cocode allow learners to view multiple programming environments from the other learners while writing and running their own code, within a single web page. Omoronyia et al. [38] introduced a system that showed programmers' activity logs to other programmers, but this system also serves programmers working on the same project to provide the shared understanding essential in collaborative software development. Cocode, however, is designed to allow learners to feel the social presence from their co-learners while the amount of actual information delivered through co-learner screens is adjustable so that the learners can solve the exercises by themselves.

Other research outcomes on web-based collaborative working environments supported multiple users working on the same tasks. These include studies that introduced co-web-browsing for tasks like Internet shopping, which was found to provide a social presence to users and increase their engagement [43, 53, 57], or collaborative writing environments based on Google Drive[3] that found there are a variety of useful styles of synchronous collaboration in document writing [49, 50, 55].

Unlike these research outcomes, Cocode shows that sharing the learning environment in online programming classes can provide a social presence, even in online classes when there are no exercises that require collaborative work; all learners work asynchronously and individually in the system on their own tasks. To the best of our knowledge, Cocode is the first system that shares other users' activities on their independent tasks to provide a social presence in the learning experience.

2.5 Sharing the Learning Environment

There are other educational tools that visualize multiple learners' working environments, similar to Cocode's co-learner screens. These include RIMES [29] and OverCode [19], where instructors can see all of their learners' responses aggregated onto a dashboard. RIMES allows instructors to embed interactive multimedia exercises into lecture videos, and see all the learners' responses on a single page, while OverCode allows instructors to see learners' answer codes to programming exercises in clusters of similar groups of code. Codeopticon [21] shows multiple learners' code editors on a single web page, and allows instructors to provide real-time tutoring to multiple learners at the same time. These tools have user interfaces that look similar to Cocode, but they serve different purposes. For example, RIMES and OverCode create a summarized report that includes responses from multiple learners, while Cocode shows all activities along the timeline to provide as many visual cues to the learner as possible. Codeopticon is also designed for instructors trying to help learners; therefore, it delivers as much information as possible from the learners.

These tools aim to help instructors and staffs in online classes by reporting how learners are working on programming exercises. On the other hand, Cocode tries to support learners by providing a social learning environment as it displays other learners' programming environments along the right edge of their screen. Unlike other tools, Cocode tries to hide the actual information on the co-learner screens but shows all raw activities to build social presence among the learners.

3 COCODE

According to previous studies, social interactions are essential for learners in online classes [18]. However, unlike in offline classes where learners can see their co-learners' learning activities, social presence from the co-learners in online classes does not naturally emerge during their usual learning activities.

Based on the idea that online learners may feel a stronger social presence when they can see their co-learners' learning activities, we designed and built Cocode. Cocode is a web-based system for online programming classes that allows learners to write, edit, run, and grade their code for

[3] https://drive.google.com

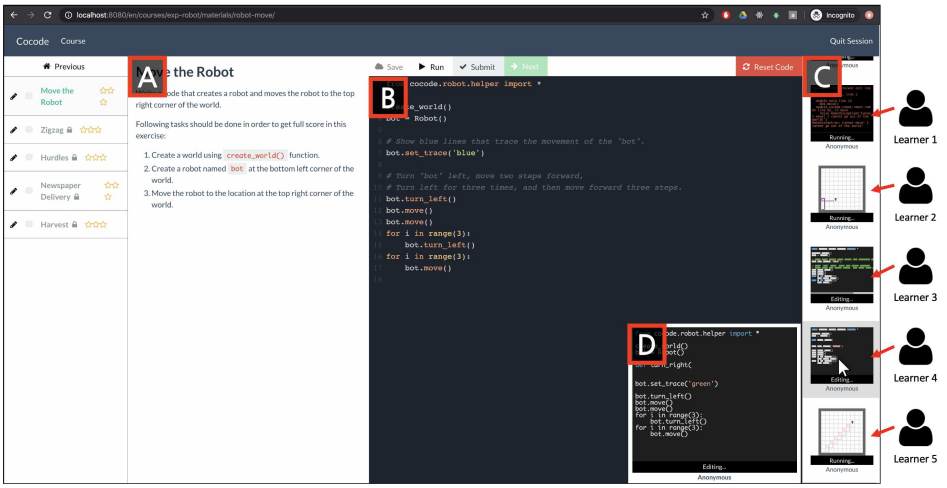

Fig. 1. This is an overview of Cocode's user interface. This shows a web page for a web-based, hands-on programming exercise. There are (A) instructions for the programming exercise, (B) a code editor, and the (C) co-learner screens on the right side of the page where other learners' learning activities are shown. Learners can scroll through the co-learner screens to browse them. When the learner puts the mouse cursor on one of the screen boxes, (D) a bigger screen box is shown in the code editor's bottom right corner. This bigger screen shows the code content of the co-learner, unlike the small screens where alphabetical letters in the code are hidden and shown as square blocks so that they are incomprehensible. The layout of the web page is responsive, but panels are not resizable. The number of co-learner screens visible on the web page depends on the size of the screen; four screens are visible at a time on a small laptop screen (1280x720) while seven screens are visible at one time on a bigger screen (2560x1440).

hands-on programming exercises. While working on the programming exercises, learners can see how other learners worked on the same programming exercise on the co-learner screens along the right edge of the web page. Figure 1 shows the overview of Cocode's user interface.

In this section, we describe a user scenario that shows the learning experience in Cocode, introduce the formative studies and discuss our design rationale, and then describe the key elements of Cocode.

3.1 User Scenario

Elsa is taking a fully online introductory programming class from her university. She opens up the course website on her laptop computer, logs in to her account, and opens up an exercise web page.

Today's exercise requires her to write a Python code to move a robot on the screen and pick up all the items visible in the "robot world". Elsa remembers how to create and move a robot on the screen using the given library and starts to write the code. The co-learner screens on the side show many other learners as they are starting to write the code now, too. She can see the co-learner screens are getting filled with their code in the editor, but the alphabetic characters on the screens are hidden to prevent copying. She feels that this experience is similar to an offline class she took last semester. She had to attend weekly lab sessions in an offline "computer classroom" for that class, and she solved a few hands-on exercises in the classroom. Although there were almost no interactions between the students, she could see that the others were working on the same exercises and it encouraged her to finish her work.

As Elsa continues to work on her code, she realizes that she does not remember the exact syntax for if statements in Python. However, she sees that one of the co-learner screens shows lines of code that look like *if-elif-else* statements by the ways they used *colons (:)* and indentations. She puts the mouse cursor on that screen, and the code content shows up enlarged on her screen, which quickly reminds her how to write an if statement.

After finishing most of the tasks, Elsa is struggling with the last task in the exercise. However, she is motivated to finish the work by herself. She sees the co-learners worked on the same exercise, and continues her work.

Elsa finally finishes the first exercise. After Cocode's auto-grader provides her score, she decides to see how some of the other learners solved the same problems. She looks over a few other learners' codes and finds out that one of them solved the problem with code that is remarkably more efficient than her code. Elsa takes some ideas from that code and makes her code more efficient. She leaves a comment that explains how her code works and moves on to the next exercise. Now, other learners can see Elsa's activity logs from her work on this exercise.

3.2 Formative Studies

We designed and built the prototype of Cocode to help provide a social presence for learners in online programming classes. To explore the use of Cocode, we first published an open course website and collected activity logs from volunteers who visited the website to evaluate the system and improve the learning experience. We then ran a user study with paid participants from Amazon Mechanical Turk (AMT) to collect user feedback.

In this section, we first briefly introduce the findings from our open course website, and then describe the user study in AMT. We will also introduce the design decisions made from the findings.

3.2.1 Open Course. Based on the first prototype, we ran an open course containing 17 programming exercises borrowed from KAIST's CS1 course, covering basic programming concepts like iterations, functions, and conditional statements. For a three-month period from September to December 2019, we conducted several user studies in this open course. We collected the activity logs and user opinions on Cocode from the 300 volunteers among the 2.4K visitors. These volunteers took the course and participated in our user studies. Although we could not get many post-study survey results from the participants (only three of them participated in the survey), we gradually changed some of the features of Cocode based on our observations of the user behaviors.

First, we found that there were too few concurrent online learners. The first prototype was designed to only show the activities of the other learners who were currently online; thus, learners could not see many co-learner screens in Cocode. The average number of concurrent online learners was under three. We could have forced the learners to participate at a specific time so that they could see each other via the co-learner screens, but we would lose one of the main advantages of online classes: the learners being able to engage in the class whenever they wanted. Therefore, we decided to provide the prerecorded version of co-learner screens when there were under eight online learners at the time.

Next, we found that there were no conversations between any of the learners during the study. The prototype originally had a text-based chat feature. However, only a few learners said something in the chat box and no one got an answer. Since no one used the feature, we decided to remove the chat box from Cocode. We also decided to only provide the prerecorded version of co-learner screens and stop providing synchronous co-learner screens from other online learners. It was easier to maintain the quality of co-learner screens using this design, and it did not affect the learning experience because there was no direct communication between the learners.

3.2.2 User Study in AMT. To ask learners to compare the learning experience with and without Cocode's co-learner screens, we ran another user study with paid participants from Amazon Mechanical Turk. We conducted a within-subject study with 27 paid participants (23 males and 4 females, average 33 years old with SD=9.3). Participants were required to have no experience with Python programming.

In the study, we asked the participants to solve the first eight programming exercises borrowed from KAIST's CS1 course. Participants were presented with four exercises with co-learner screens and four without co-learner screens. Half of the participants (11 males and 3 females) were presented with co-learner screens first and then presented with the user interface without co-learner screens afterward, and the other half (12 males and 1 female) were presented in the opposite order to mitigate order effects. The exercises used in the study were designed to teach conditional statements, loops, and functions in Python. It took approximately 90 minutes to solve them all. The participants were required to solve all of the exercises and then answer our survey questions.

After solving all of the given exercises, the participants were asked two open-ended questions: *"How did viewing the code editors and running results of other users help you accomplish the exercise problems? If it did not, why?"* (Q1) and *"How did viewing the code editors and running results of other users help you feel less isolated or disconnected? If it did not, why?"* (Q2). The answers to the open-ended questions were qualitatively analyzed using an open coding method with a bottom-up approach [7]. The first author segmented the answer texts into the smallest logical units, and then the first pass was performed to assign categories to the units. Next, several additional passes were made together with the co-authors to revise and aggregate the categories.

For Q1, 13 out of 27 participants answered that co-learner screens helped them to solve the exercise, 12 answered that the screens were not helpful, and 2 answered the screens were helpful but did not think they should be using the co-learner screens.

The participants answered that checking other learners' code content *helped* them to solve the exercise problems (8/27), and *motivated* them to finish the exercise (5/27). However, others answered that their co-learners' code was *too small to read* so it was not helpful (2/27), and some others answered that it was *distracting* to see their co-learners' screens so they did not want to see them (7/27). Finally, others answered that it felt like they were *cheating* by reading others' code (2/27), so they did not want to use their co-learner screens even when they thought it would be helpful for solving the exercises.

Meanwhile, the answers to Q2 showed that most of the participants felt less isolated due to the co-learner screens in Cocode (21/27). Some of the participants mentioned that they felt like they were *working together* with other learners (10/27), they liked how others *guided* them with their code (4/27), or it reminded them of the experience in a *physical classroom* (3/27). However, other participants thought the co-learner screens made them feel less isolated but *distracted* them at the same time (2/27), or the screens made them feel like it was a *competition* so they did not like the experience (2/27). On the other hand, some of the participants who did not feel less isolated due to the co-learner screens said that they *did not know why* they were not feeling less isolated (4/27), or they *did not use* the feature at all (2/27).

The results showed that about half of the participants thought that Cocode helped them solve the exercises, 77% of them thought that Cocode made them feel less isolated, and they preferred to use Cocode.

Previous research found that some learners (*preferred solitaires*) preferred to study alone than with their peers [9, 48]. Although learners working alone against their preference are not happy and are at risk of dropping out, preferred solitaires may perform better when they work alone [23, 48]. We can expect that preferred solitaires using Cocode would not want to use the co-learner screens; therefore, we decided to give learners the option to completely hide the co-learner screens.

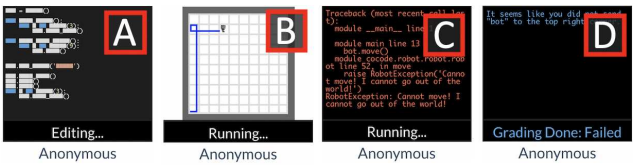

Fig. 2. Co-learner screens show (A) the code editor, (B) running screen, (C) standard outputs and errors, or (D) grading result depending on the learners' current behavior.

Finally, some learners mentioned that they did not want to use the co-learner screens because they felt like they were cheating when checking others' code. To provide visual cues from other learners for social presence without showing the actual code contents, we decided to hide all of the alphabetical characters on the co-learner screens; however, we allowed learners to reveal the characters and view other learners' code by explicitly putting their mouse cursor on the co-learner screens, since many learners said that reading others' code contents helped them or motivated them to finish the exercise. This change also allowed us to see if learners in Cocode viewed other learners' code, or if it was enough to feel co-learners' social presence on their screens without the actual code.

3.3 Design

In this section, we explain the detailed elements of Cocode that help provide social presence for the learners.

3.3.1 Non-intrusiveness. Existing solutions for social presence in online classes often require explicit actions from the learners. For example, popular solutions like forums and comment sections require learners to explicitly write and upload postings. However, Cocode helps learners improve social presence with other learners by simply working on the programming exercises. Unlike online learners in other fields, learners in Cocode write and edit their code, run their code, and have their code graded in the system. Since the learners are supposed to do most of the learning activities in the Cocode system, we can easily utilize most of the meaningful activities for use as visual cues for social presence in the co-learner screens. As a result, Cocode can show various types of learning activities in the co-learner screens. The screen can show the code editor, running output, or grading output depending on what the learner is doing at the time. Cocode efficiently shows the co-learners' activities in this way, with smaller screen sizes. Figure 2 shows examples of the screens with various activities.

Co-learner screens cannot be a direct substitute for forums and comment sections because it does not support explicit interactions among the learners. However, the co-learner screens can be used together with forums and comment sections. We believe this combination will give learners an experience similar to what they expect from offline classrooms. In this way, learners can feel a social presence from the co-learner screens and also interact with co-learners in online classes.

3.3.2 Asynchrony. One of the advantages of taking online classes is that learners can set their own learning schedule. However, we lose this advantage if synchronous activities are required in the class. Some of the existing solutions that use video chat naturally make the class partially synchronous by requiring learners to be online together at a specific time. However, the co-learner screens in Cocode are generated from the past logged learning activities so the product can be entirely asynchronous.

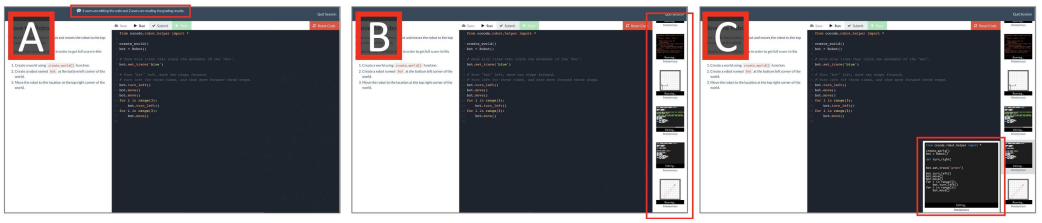

Fig. 3. Cocode allows learners to adjust the amount of information visible from their co-learners explicitly. (A) Learners can hide the co-learner screens if they do not want to see them at all (co-learners' activities are shown in the red box, summarized in short text messages) or (B) show the co-learner screens with alphabetical characters in the code hidden. (C) Learner can also reveal and read the raw code contents by hovering the mouse cursor on one of the co-learner screens.

When learners work on the programming exercises, Cocode records and uploads the activity logs from the learners. Cocode records when the code contents are edited, the cursor position when the cursor is moved in the code editor, the running outputs when learners run the code, and grading results when learners have their code graded. All logs are time-stamped, and the timestamps are also logged when learners open or close the web pages. Co-learner screens in Cocode display these activity logs from specific learners in small boxes, providing visual cues captured from their co-learners. The code editors and output screens from the co-learners' logs are simulated in the boxes using pure HTML elements. When a co-learner closes and opens the web page or is inactive for more than 15 seconds, the co-learner screens remove the inactivity and continues the simulation.

Co-learners to display activities in the screens are randomly selected from all learners who finished the corresponding exercise problem and received the full score. The user interface shows enough number of co-learner screens to fill the column of screens, and the learner can scroll to browse all co-learner screens in random order. All co-learner screens are designed to show the activity logs time-aligned with the viewer to simulate the offline classroom experience. When Cocode simulates the co-learners' environments, when n seconds have passed in an exercise web page, the co-learner screens in that web page show other learners' activities around n seconds have passed in that learner's environment. We expect this to allow learners to identify role models in their peers and compare their behavior with the role models like in offline classrooms. Cocode system is implemented to handle all of these processes from activity logging to visualization in co-learner screens without any human assistance.

3.3.3 Co-learner Visibility Modes. Cocode allows learners to change the visibility of what their co-learners can see in the user interface. By default, screens show the co-learners' working environments with alphabetical characters in the code contents hidden. However, learners can hide all co-learner screens, or reveal and read the raw code contents from one of the screens. Figure 3 shows examples of each mode.

Formative studies have shown that some people are *preferred solitaires* who do not wish to feel a social presence from their co-learners. Previous studies also suggest that these preferred solitaires might perform better when they work alone. Since Cocode attempts to simulate the offline classroom environment and provide social presence with visual cues from co-learners, it is natural that preferred solitaires choose not to see co-learner screens in Cocode's user interface. For these learners, Cocode gives an option to hide the co-learner screens and see others' activities summarized in short text messages. For example, *"Two users are running the code and three other users are editing code contents."* However, according to the studies, preferred solitaires choose to

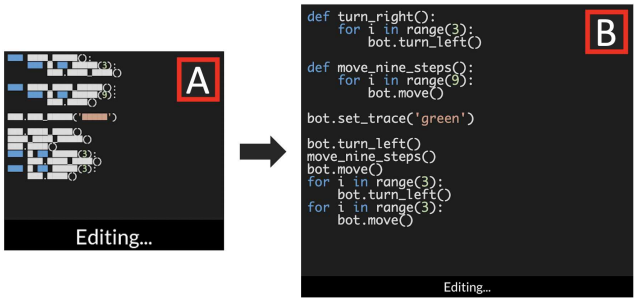

Fig. 4. Cocode allows learners to feel social presence from the other learners without looking at their others' code. (A) A co-learner screen shows other learners' code editors with all of the alphabetical characters in the code hidden. When a learner wants to see the raw contents and puts their mouse cursor on one of their co-learners' screen, (B) a bigger screen box with raw code contents is displayed.

work alone because they feel they work more efficiently that way, they think they do not need others' help, or they think their peers are a burden to them when working together [9]. Since Cocode does not require learners to undertake any explicit actions when sharing their presence with others, preferred solitaires can hide all co-learner screens but still contribute to the community by sharing their learning activities with the other learners if they choose to do so. Considering that around half of all learners were preferred solitaires in our studies, this allowed Cocode to show more co-learner screens in the learning environment to increase the social presence with other learners.

Furthermore, Cocode also allows learners to feel the social presence from co-learners but not see their actual code content so that learners could have opportunities to solve the exercises by themselves. The co-learner screens show the code editor when that learner writes or edits the code, but the alphabetical characters are hidden. Therefore, learners can see that other learners are actively writing or deleting the letters in the code editor, but they cannot comprehend the actual code content. However, code highlighting is still available, and the indentation of the code and the symbol characters are still visible. This allows co-learner screen contents to still look like a snippet of programming code. And since the indentation structure and the symbol characters allow the viewer to guess the outline of the code and the actual algorithm implemented with the code, the co-learner screens are still useful to the viewer. If the learner wants to see the raw code contents from the co-learner screens and see how others solved the exercise, the learner can put the mouse cursor over one of the co-learner screens. Then the bigger co-learner screen with raw code contents will appear at the bottom of the page. Figure 4 shows an example of a co-learner screen with hidden and exposed code contents. We expect this experience to be similar to the offline classroom experience, where learners can see other learners working on the same exercise but cannot see their actual code contents; the small font size of the code editors makes it hard for the learners to read the text on other learners' screens. This feature helps learners avoid getting hints to solve the exercise while maintaining social presence from others. We have seen that some participants in our formative studies preferred to solve the problems by themselves but still enjoyed their peers' presence.

3.4 Implementation

Cocode is a standalone web application based on the Python Django framework and Vue.js. Unlike other web services that support programming courses using Python, Cocode allows learners to

write and run Python code without server-side Python code running. We used Brython to live-transpile Python code into JavaScript code in the client web browsers to run learners' Python code locally in the client browsers. Although the current implementation only supports single-file programming exercises, we can also support multi-file exercises with in-memory file systems for browser environments.

Due to this client-side code running feature, co-learner screen sharing could be supported without additional loads to our servers. The Cocode server only sends text-based data to the client but can still show co-learners' running screens with a graphical user interface because each learner's client-side browser runs other learners' code locally. The co-learner screens are implemented to visualize the code editors and the running environments with raw HTML elements on the client side. As a result, Cocode serves multiple learners with such a dynamic user interface but only requires the server resources as small as an Internet forum application.

4 EVALUATION

To evaluate Cocode, we ran two user studies to answer the following research questions:

- RQ1. Does Cocode with co-learner screens provide more social presence than an offline programming course in physical classrooms?
- RQ2. Does Cocode with co-learner screens provide more social presence than an online programming course with existing social features?

For the first user study (Study 1), we recruited a group of participants who took the CS1 course in KAIST to compare and contrast the experience of using Cocode with an offline programming course experience in the university.

The global COVID-19 pandemic occurred after the first study. Since the disease is more likely to spread when people are physically close [17], all classes in KAIST were converted to entirely online classes to stop the disease from further spreading. Thus, in the second user study (Study 2), we were able to recruit a group of participants who took the same CS1 course in an entirely online format with video chat lectures and lab sessions that were held in web-based programming environments. Here, we tried to compare the experience of using Cocode with the experience from an online programming course in the university.

4.1 Experiment

4.1.1 Participants in Study 1. In Study 1, we recruited 23 paid participants (10 females and 13 males; 17 between 18-25 years old, and 6 between 25-34 years old) from KAIST's web community in February 2020. We required the participants to have taken the CS1 class in KAIST, but not to have majored in computer science or other majors that require significant programming skills (e.g., electrical engineering). We wanted our programming exercises to not be too easy for the participants so that they could have a learning experience. According to the pre-study survey, three participants had 1-2 years of programming experience, and the others had less than one year of programming experience. We also asked them about their offline programming class experiences; five participants took an introductory programming class in high school, and the others took one or two university introductory programming classes. All of the participants took the CS1 class in KAIST within the last three years. Some also took other programming classes that used either C++, Java, or Python.

4.1.2 Participants in Study 2. In Study 2, We recruited 22 paid participants (6 females and 16 males; 21 between 18-25 years old, and 1 between 25-34 years old) from the same university's web community in September 2020. To compare their experience in Cocode against their experience in the university's online programming class, we required the participants to have taken the online

CS1 class during the COVID-19 pandemic. We also required the participants not to have majored in programming-related subjects, the same as in Study 1. According to the pre-study survey, 4 participants had 1-2 years of programming experience, and the others had less than one year of programming experience. All of the participants in Study 2 took the CS1 class in KAIST within the last six months.

4.1.3 Tasks. In both Study 1 and Study 2, we asked participants to solve six hands-on programming exercises borrowed from Codecademy's *Introduction to JavaScript* course. We selected this course because it had exercises designed to teach learners without any additional materials like lecture videos. It was also an introductory programming course, making the learning experience in Cocode more suitable when compared to participants' experiences from the CS1 class in university. No participants learned JavaScript before, so the participants still learn something from the course as well. Before starting to solve the exercises, participants were asked to read a short article that explained how to edit, run, and grade their code, how to show and hide the co-learner screens, and how to read the code contents in the co-learner screens with mouse cursor interactions.

We presented two exercises with co-learner screens and two exercises without co-learner screens to let participants compare the learning experience. Half of the participants were presented with co-learner screens first, then presented without co-learner screens. The other half of the participants were presented with exercises in the opposite order to mitigate the ordering effects. After finishing four exercises with the given environments, participants were asked if they wanted to use the co-learner screens or not on the last two exercises. We showed a dialog to the participants when they first accessed the exercise and forced them to select either a user interface with or without co-learner screens before they started to work on the exercise. All of the participants could see at least 10 other learners when the co-learner screens were enabled, since we prepared the activity logs recorded from volunteers who were also not majoring in programming-related subjects. It took around 20 minutes to solve all exercises, and the participants were required to solve them all. The participants were then asked to answer our survey questions.

During the experiment, we observed and recorded learners' activities to use in our analysis. We observed the amount of time learners spent on finishing the given exercises, the amount of time spent on code writing and editing, the number of code executions, the change logs of co-learner screens' visibility settings, and when and how long the mouse cursor was over the co-learner screens to read the code contents.

All of the participants finished all of the given exercises and answered our survey questions. After solving six exercises with and without co-learner screens in Cocode, the participants were asked to answer post-study survey questions that included: (1) questions to measure social presence in Cocode and social presence in the offline or online CS1 class they took before (5-point Likert scale), (2) questions to compare their experience in Cocode with co-learner screens against the experience without co-learner screens (5-point Likert scale), and (3) open-ended questions for further discussion.

To measure social presence during their learning experiences, we asked participants about the reality of other learners' presence, and also if other learners' context, emotions, personalities, and personal histories were visible in the learning environment, although Cocode cannot deliver other learners' personalities or personal histories due to the limitation of its design. These five dimensions of social presence are based on the findings from Kehrwald et al. [28]. In the paper, the authors stated that these are what compose social presence in distance education since these elements decide co-learners' ability to demonstrate their state of being in a virtual environment.

Open-ended questions compared experiences in various learning environments, asked about the reasons for participants' behavior in Cocode's user interface, and collected their thoughts on how

Table 1. The statistics of the of the post-study survey results. These are mean (and standard deviation) values for the answers to the questions that measure social presence in Cocode with co-learner screens, and in the university CS1 class they took. The Wilcoxon rank-sum test was used to measure the significance of the differences in average answers . For Study 1, the answers to the questions about the social presence in Cocode are compared against the answers to the questions about social presence in the offline CS1 class. For Study 2, the answers to the questions about social presence in Cocode were also compared against the answers to the questions about social presence in the online CS1 class. The questions were: *I felt like I was learning together with other learners* (Q1), *I could see what other learners were doing* (Q2), *I could see other learners were having difficulties* (Q3), *I could learn about other learners' personalities* (Q4), and *I could learn about other learners' personal histories* (Q5). *n=23* for Study 1 and *n=22* for Study 2, and each question used 5-point Likert scale. Strongly Disagree (1) to Strongly Agree (5). $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$.

	Study 1			Study 2	
	Cocode w/ Scr.	Offline		Cocode w/ Scr.	Online
Q1 (Presence)	*3.09 (1.25)	3.91 (1.06)	Q1	***4.09 (0.73)	1.90 (1.16)
Q2 (Context)	3.43 (1.06)	3.48 (0.97)	Q2	***3.68 (0.82)	1.86 (1.29)
Q3 (Emotion)	**2.83 (1.01)	3.91 (0.88)	Q3	***3.55 (1.08)	2.36 (1.19)
Q4 (Personality)	***1.96 (0.75)	3.48 (1.10)	Q4	1.91 (1.00)	1.50 (0.94)
Q5 (Personal History)	*1.48 (0.50)	2.09 (1.02)	Q5	1.32 (0.55)	1.18 (0.39)

to improve Cocode further as follows: *"In the university's CS1 class, what kind of interactions did you have with other people?"*, *"How is the social presence from co-learners in the university's CS1 class different from or similar to the social presence from Cocode's co-learners?"*, *"What was the difference between your experience when there were co-learner screens and when there are no co-learner screens in Cocode?"*, *"Did you put your mouse cursor on the co-learner screens to look at other learners' code contents? Why did you look at them?"*, and *"How can we make the learning experience in Cocode more similar to the experience in offline classes? How can we improve the learning experience in Cocode?"*

All of the questions in Study 1 and Study 2 were asked to the participants in Korean; therefore, all questions and answers from the studies in this paper are translations.

4.2 Results

4.2.1 Cocode vs. Offline Class. In Study 1, we tried to find how social presence in Cocode with co-learner screens differed from the social presence from the offline programming class; thus, we asked a few questions about social presence in Cocode and the offline class from their experience.

Participants' answers to these questions are available in Table 1. The answers show that, as expected, the learning experience in Cocode provides less social presence compared to the learning experience in the offline class. However, the differences in the answers were not significant when the question asked participants if they saw what other learners were doing (Q2). On the other hand, the average values for the answers to four other social presence elements in Cocode were significantly lower than the answers about the offline class.

These are expected limitations for Cocode, since there are no visual cues from people's actual existence. It is critical to see facial expressions and body gestures for figuring out other people's emotions and personalities, and it takes many interactions to learn about other people's personal histories. As expected, the offline classroom environment provides social presence to the learners better; but Cocode still offered some social presence elements to learners in online programming classes.

Table 2. The statistics of the post-study survey results from Study 2 that compare the experience with and without co-learner screens in Cocode. These are mean (and standard deviation) values of the answers to the questions that measured social presence in Cocode with and without co-learner screens. Answers to the all questions were higher for Cocode with co-learner screens. The Wilcoxon rank-sum test was used to measure the significance of the differences in average answers. *n=22*, and each question used a 5-point Likert scale. Strongly Disagree (1) to Strongly Agree (5). $^{**}p < 0.01$, $^{***}p < 0.001$.

	Cocode w/ Scr.	Cocode w/o Scr.
Q1 (Presence)	***4.09 (0.73)	2.64 (1.07)
Q2 (Context)	***3.68 (0.82)	2.41 (1.07)
Q3 (Emotion)	**3.55 (1.08)	2.32 (1.14)
Q4 (Personality)	1.91 (1.00)	1.59 (0.72)
Q5 (Personal History)	1.32 (0.55)	1.23 (0.52)

4.2.2 Cocode vs. Online Class without Co-learner Screens. In Study 2, all of the participants we recruited for the user study took the university CS1 class entirely online; live video-chat lectures replaced lectures in the classroom, and online lab sessions with forums and chat sessions replaced lab sessions in the classroom.

To compare the experience in Cocode with co-learner screens against the experience in the university online CS1 class, we asked questions to measure social presence in each of the learning environments. The questions were identical to the questions used in Study 1: the context, reality of the presence, emotions, personalities, and personal histories of the co-learners.

The answers are also available in Table 1. The results showed that the learning experience in Cocode provided a significantly more social presence in some dimensions compared to the learning experience in the online class, even when the class was based on live video lectures, Internet forums, and chat sessions.

According to the results, participants felt more like they were learning together with other learners (Q1), they could see more about what other learners were doing (Q2), and they could also ascertain if other learners were having difficulties (Q3) in Cocode than in the online CS1 class.

However, the answers also showed that the information about other learners' personalities or their personal history available in Cocode was not significantly different from their online class experience. This result was expected since the co-learner screens only showed how learners worked on programming exercises; thus, it was hard to find out about other learners' personalities or personal histories. This is partly because the IRB approval for our experiments explicitly required all participants to be completely anonymous to each other. We could not even give random nicknames to the participants; all other learners were labeled as *anonymous users* in the user interface. This helped us to minimize the privacy issues that may have prevented learners from using the feature. However, it also limited the dimensions of social presence that Cocode could provide to the learners.

Despite the restrictions, we tried to design and implement Cocode and its co-learner screens to simulate an environment where learners were working on programming exercises in an offline classroom together with other learners.

4.2.3 Cocode with Co-learner Screens vs. Cocode without Co-learner Screens. We asked additional questions to the participants in Study 2 to explicitly evaluate the co-learner screens. Since all of the participants had experienced using Cocode with and without the co-learner screens, we asked these questions to measure social presence in Cocode *without* the co-learner screens. Again, the questions were identical to the questions used in Study 1.

Table 3. Top five most commonly found ideas from the answers (and the numbers of participants that mentioned them in their answers) for open-ended survey questions that asked about the learning experience in Cocode, in Study 1 and Study 2. *n=23* for Study 1 and *n=22* for Study 2.

	Study 1	Study 2
1	Co-learners' various codes are visible like in the offline class (12/23)	Co-learners' various codes are visible unlike in the online class (8/22)
2	It is hard to concentrate on the exercise when co-learner screens are visible (8/23)	Co-learner screens helped me write code for the exercises (7/22)
3	No direct communication is available with co-learners in Cocode (7/23)	I read others' code because I was just curious about how others are doing it (7/22)
4	The progress of co-learners is visible in the co-learner screens (5/23)	It feels like a competition when co-learner screens are visible (7/22)
5	It is hard to solve the exercise by myself with the co-learner screens visible (5/23)	It feels like I can communicate with others when co-learner screens are visible (6/22)

The answers are available in Table 2. The learning experience in Cocode with co-learner screens provided a significantly more social presence in the dimension of context, emotions, and the reality of the presence than the experience without co-learner screens. This result shows that even though Cocode without co-learner screens had some features to provide a social presence to the learners, the co-learner screens in Cocode were essential for providing significantly more social presence to the learners.

The results from Study 1 and Study 2 show that the learning experience in Cocode with co-learner screens provided less social presence compared to the offline class but provided significantly more social presence when compared to Cocode without co-learner screens or the online class with existing features for social presence. This shows that Cocode with co-learner screens can improve the learning experience when it is difficult or impossible to run offline classes, and the class *must* be run entirely online.

4.3 Analysis

In this section, we analyze and discuss the learners' activity logs and survey answers collected from the two user studies. The answers to the open-ended questions were qualitatively analyzed using an open coding method with a bottom-up approach [7]. The first author segmented the answer texts into the smallest logical units, and then the first pass was performed to assign categories to the units. After the first pass, the second author was given the answer texts segmented into the units and the list of categories found by the first author, and then independently performed a pass to assign given categories to the units. Agreement between the coders was high (Cohen's κ = 0.79). After the first pass by the two independent coders, multiple additional passes were made by all of the co-authors to resolve any disagreements and revise the results.

4.3.1 Experience in the University CS1 Class. In Study 1, all of the participants went to lectures in the classroom and had weekly lab sessions with their co-learners when they were taking the offline university class. The lab sessions were three hours long, and learners solved between two and five hands-on exercise problems assigned for that week in the classroom. There were seven or eight teaching assistants in a classroom with about 40 learners; they could ask the teaching assistants questions whenever they needed to. For some exercise problems, learners were asked to try the pair programming with one of their co-learners in the classroom.

We asked the participants to describe the interactions they had with other people in the university class, and most of the participants mentioned that they attended the *lab sessions together with their co-learners* (18/23), they *helped each other* (15/23) in the lab sessions, and they had *pair programming sessions* (15/23) in the classroom. Some mentioned that *the teaching assistants or tutors helped* (5/23) them, or they could *observe co-learners' code* (2/23) in the classroom.

> *"We had pair programming sessions in the lab sessions. We discussed and solved the given exercise tasks together, although we worked on the homework assignments individually." (P1-7)*
>
> *"I could peek and see how others were solving the exercises, or I could see that others were struggling with the problems. I also asked teaching assistants for help a lot in the class." (P1-9)*

On the other hand, in Study 2, all of the participants took the university's CS1 class during the COVID-19 outbreak. All offline activities in the CS1 class turned into online activities. The lectures in the classroom were replaced with lectures via live video chat sessions, and the offline lab sessions became online lab sessions based on Internet forums and text-based chat sessions with the teaching assistants.

The most common answer from the participants in Study 2 was that *there were no interactions with other learners at all* (18/22) in the class, while many of them mentioned that *they had interactions with teaching assistants or tutors* (13/22). Participants who had interactions with other learners said that they *interacted with their friends taking the class together* (4/22). Only two mentioned that they *interacted with other learners in the forums* (2/22).

> *"I could discuss my code with the teaching assistants, but I had no interactions with other students in the class at all." (P2-1)*
>
> *"We had online lectures, so we didn't code together in the class, but we could ask questions to the teaching assistants. I could study using other students' questions in the Q&A forum. And I personally asked questions to my friends sometimes." (P2-3)*

The participants' answers show that the university CS1 class used to have lab sessions where learners could actively interact with other learners and have a social presence in the class (Study 1). However, under the circumstances where offline activities were unavailable, learners did not interact with other learners at all, even when they participated in the live video chat lectures and had forums for communication. Only the learners who took the class with their friends interacted with those friends, and others communicated with the lecturer and teaching assistants in the class only; most of them had no social presence from co-learners in the class.

4.3.2 Cocode with Co-learner Screens vs. University CS1 Class. When we asked the participants *"How is the social presence from co-learners in the university's CS1 class different from or similar to the social presence from Cocode's co-learners?"*, the participants from Study 1 and Study 2 discussed similar topics but in different ways.

In Study 1, the most common idea found from the participants' answers was that in Cocode, *co-learners' various codes were visible* (12/23) and *progress of co-learners were visible* (5/23).

> *"It was similar that I could see various codes. I used to share the answer code with others after finishing and submitting assignments. But it was different from the offline class because in Cocode I could see the progress of others writing code or how others were editing their code." (P1-1)*

However, they also mentioned that in Cocode, *no direct communication was available with co-learners* (7/23), *they could not feel emotions from the co-learners* (3/23), and *the faces of the other learners were not visible* (2/23) unlike in the offline class.

> *"I liked that I could see the progress of others writing their code. However, I was sad that I could not talk about ideas to solve problems like in the offline class." (P1-12)*

Some mentioned that they liked that *it was easier to see others' code contents* (2/23) in Cocode, unlike how it was awkward to explicitly look at others' screens in the offline classroom.

On the other hand, in Study 2, participants felt like they had a more social presence in Cocode in many dimensions than in the online university class. Participants said that in Cocode, *co-learners' various codes were visible* (8/22) and they felt like *they could communicate with their co-learners* (6/22) unlike their non-social online class experience.

> *"I didn't have any chances to communicate with other students in the CS1 class, but in Cocode I felt like we co-learners revisedwere understanding each other since we shared our screen contents." (P2-6)*

They also mentioned that *they could see what others co-learners were doing* (4/22) and *felt relieved because they could see others were struggling like they were* (4/22), and even *they could see others' emotions* (2/22) in the co-learner screens in Cocode, unlike in the online CS1 class.

> *"Since last semester's class was an online class, we didn't have many chances to discuss things with each other, and we couldn't gauge others' feelings. But in Cocode, I can see what kind of difficulties students are having, and what errors they are facing, and share those feelings at least a little bit." (P2-9)*

> *"In the live video lectures in the CS1 class, people talked in the chat section so I felt like we were learning together. However, people asked difficult questions to each other and I just didn't care that much about them. ... In Cocode, it's very different. When there are co-learner screens enabled on the side, they give me a certain sense of relief and feeling of solidarity. I could see that 'This person is working hard' or 'This person is doing the same thing with me' from those screens, and these thoughts gave me peace of mind." (P2-18)*

However, some participants said there was *no communication with the co-learners* (4/22) in Cocode, just like in the university online class. Some also said that *it felt like a competition* (4/22) to work with the co-learner screens.

> *"Communication was not available both in the CS1 class and in Cocode. I like to concentrate on my work when I'm writing code, so others' screens on the right side don't tell me anything more than they are doing something together with me. Since that wasn't even possible in the CS1 class, there may have been a small difference." (P2-19)*

> *"I'm kind of slow when I write code, and I got stressed a little bit from seeing other students finishing the tasks before I got to finish." (P2-22)*

These answers from the participants support the statistics in Table 1. The table shows the results from the survey questions that measure social presence in each of the environments across five dimensions; they show that Cocode with co-learner screens gave more social presence than in the online university class, but less social presence than in the offline university class although the written code and the progress of co-learners are also visible in Cocode.

4.3.3 Cocode with Co-learner Screens vs. Cocode without Co-learner screens. The participants' survey answers showed that they were aware that they had more social presence when co-learner screens were visible. However, that does not mean that all participants liked to have co-learner screens in their working environments. We asked the participants *"What was the difference between your experience when there were co-learner screens and when there were no co-learner screens in Cocode?"*, and the answers showed that the participants had various opinions on having co-learner screens visible in Cocode.

In Study 1, the most common answer was that *it was hard to concentrate on the exercise* (8/23) when there were co-learner screens.

> *"I could concentrate better when I didn't see the co-learner screens, although I got motivated to solve the problems faster with the screens." (P1-17)*

There were also other negative answers like *it was hard to solve the exercise by themselves* (5/23) with the co-learner screens visible, *they felt pressure from other learners' screens* (4/23), or *it felt like a competition* (3/23).

> *"When I saw the co-learner screens, I felt the pressure that I had to do it faster. This program produces competitions." (P1-11)*

> *"My concentration level went down when there were co-learner screens. I had to check if others were working faster or slower than me, and if others were solving the problem in a different way than I was. It felt like a competition, and I wanted to see others' code." (P1-16)*

On the other hand, there were some positive answers like *the co-learner screens motivated them to solve the exercises* (3/23), *they felt relieved to see others were struggling like they were* (2/23), or *the screens helped them writing code for the exercises* (2/23).

> *"I worked harder, too, since I felt like others are working harder." (P1-18)*

> *"I felt relieved after seeing others were also getting poor grading results like me. Others were writing the wrong code, too." (P1-22)*

Finally, some participants said that *their learning experience was not different whether there were co-learner screens or not* (4/23).

However, in Study 2, participants were relatively more positive about the learning experience with co-learner screens. In the second study, more participants felt like *the co-learner screens helped them write code for the exercises* (7/22), *the co-learner screens motivated them to solve the exercises* (4/22), or *they felt relieved to see others were struggling, too* (4/22).

> *"I could see other people's code changing in the co-learner screens, and I felt like I was taking the class together with these people. I read some of the others' code to find my errors and mistakes quickly." (P2-7)*

> *"I felt relieved seeing other people were writing code, too. I was not the only one having a hard time." (P2-8)*

> *"I was motivated a little bit more when looking at other people's screens. I think it made me work faster." (P2-12)*

However, some participants still did not like that *it felt like a competition* (7/22) with co-learner screens.

> *"I usually code slowly, but when I watch other students pass first, I feel pressured to finish faster." (P2-21)*

Two participants thought that *there was no difference when co-learner screens were present or absent* (2/22) in the second study. However, no participants mentioned that co-learner screens disturbed them from concentrating on the exercise or solving the exercise by themselves, while this was the most common answer in Study 1.

These answers are related to the results shown in Table 4. This table shows how many participants chose to see the co-learner screens in the last two exercises, and how many participants explicitly read other learners' code contents from the co-learner screens. We asked the participants in the survey that if they chose to see the co-learner screens and if they explicitly put the mouse cursor on the screens to read other learners' code contents, and their self-reported answers were then

Table 4. This table shows the percentage of user study participants who chose to see the co-learner screens in the last two exercises, and the percentage of participants who explicitly read other learners' code contents from the co-learner screens. We asked the participants if they included the screens in their environment and read others' code, and validated their answers with their activity logs. A similar number of participants chose to see the co-learner screens in Study 1 and Study 2, but the numbers of participants who explicitly read other learners' code contents show a 51% difference between the results in Study 1 and Study 2. The Chi-squared test was used twice to determine whether a learner's experience in the university CS1 class was associated with the behavior of reading the code contents and the behavior of choosing to view the co-learner screens in Cocode. The results showed that there was a significant association between the learner's experience in the university CS1 class and the behavior of reading the code contents ($p = 0.001$), while there is no association with choosing to view the co-learner screens ($p = 0.6$). *n=23* in Study 1 and *n=22* in Study 2.

	Study 1 (%)	Study 2 (%)
Read Code Contents	13.04	63.64
Viewed Co-learner Screens	60.87	72.73

Table 5. The results from the post-study survey questions for comparing the experience in Cocode with co-learner screens against the experience without co-learner screens. Q1 asked whether the participants *preferred to see co-learner screens* in the user interface, and Q2 asked if *co-learner screens helped them to solve the exercise problems*. Positive percentages indicate the proportion of "agree" (4) and "strongly agree" (5), and the negative percentages indicate the proportion "disagree" (2) and "strongly disagree" (1). All other participants omitted in this table answered "neutral" (3). *n=23* for Study 1 and *n=22* for Study 2, and each question used a 5-point Likert scale. Strongly Disagree (1) to Strongly Agree (5).

	Study 1					Study 2			
	Pos. (%)	Neg. (%)	Mean	S.D.		Pos. (%)	Neg. (%)	Mean	S.D.
Q1 (Preferable)	43.48	43.48	3.00	1.29	Q1	36.36	36.36	2.95	0.93
Q2 (Helpful)	52.17	30.43	3.35	1.37	Q2	54.55	27.27	3.32	1.06

validated from the activity logs. Some participants had records of putting their mouse cursors on their co-learners' screens and answered that they did not read other learners' code contents. However, they did not put their mouse cursor on co-learner screens for more than one second; therefore, we surmised that they were not intending to read the code contents but their mouse cursors were just passing across the co-learner screens.

In Study 1 and Study 2, a similar number of participants decided to include the co-learner screens in the environment when we asked them to decide whether or not to include the screens. However, when we counted how many participants used the feature to reveal the raw code contents on the co-learner screens explicitly, the results were significantly different between Study 1 and Study 2. Table 4 shows that in Study 2, 51% more participants explicitly read code contents from the co-learner screens when compared to the result from Study 1; hence more participants in Study 2 explicitly said that co-learner screens helped them *write code*, or it was a relief to see others were also struggling.

However, it does not necessarily mean that the participants in Study 2 liked to use the co-learner screens more than the participants in Study 1. Table 5 shows that there was a similar number of participants who liked and did not like to see the co-learner screens in Cocode, while more than half of the participants thought that the co-learner screens helped them to solve the exercises.

Table 6. The statistics of the learners' behaviors in Study 1 and Study 2. These are mean (and standard deviation) values for the time (in seconds) spent on the exercise web pages, time spent typing on the keyboard in the code editors, and the code execution count. Participants in Study 2 spent 12% more time on the exercise web pages, but spent similar amounts of time on editing code contents. Code execution counts were similar in Study 1 and Study 2. No values were significantly different against the values from the other study. *n=23* in Study 1 and *n=22* in Study 2.

	Study 1	Study 2
Time Spent (sec.)	341.65 (132.48)	383.59 (158.86)
Time Spent in Editing (sec.)	145.75 (33.63)	153.32 (27.75)
Run Count	12.55 (3.89)	12.82 (4.63)

4.3.4 Reading Code Contents from Co-learner Screens. We were also curious about why some participants read the code contents in the co-learner screens, while others preferred to view the co-learner screens with hidden characters.

In Study 1, participants said that they did not read others' code contents because *they wanted to solve the exercises by themselves* (4/23), *they did not have to read because the exercises were easy* (3/23), or they had no reason to read others' code.

> *"I read the other student's code only once when I was stuck while solving the task. I didn't use that feature since I felt like the meaning of learning was reduced a lot when I looked at the co-learner screens." (P1-2)*

Participants who read the code contents said that they *used the feature because they were curious about how others are doing* (3/23).

> *"I was curious about how others were doing. I wanted to see who's got it wrong and who's got it right." (P1-9)*

On the other hand, in Study 2, 51% more participants explicitly read the code contents in the co-learner screens than in Study 1. We tried to find the reason but many participants gave us unclear answers.

They said that they read the code because *they were curious about how others were doing* (7/22); and there were reasons like *to see how others implemented the code* (2/22), *to see others' errors so that they could avoid having the same problems* (2/22), or *to understand the exercise problem correctly* (2/22).

> *"I used the feature to see other students' mistakes so that I wouldn't make the same mistakes." (P2-5)*

> *"I read the code from the students who were really fast in solving the problem. I was curious if they had solved the problem differently." (P2-21)*

Other participants said that they did not read others' code since *they did not have to read because the exercises were easy* (6/23), or simply *there was no reason to read them* (3/23).

> *"I didn't use the feature. The problems were easy, so I didn't need any hints from other students. However, I may want to read other's code if the problems were harder to solve." (P2-2)*

While the most common reason for reading the code was that they were curious and some said there is no clear reason, we could still find some reasons in the answers for the learners' activity logs shown in Table 4. The table shows that significantly more participants in Study 2 explicitly read the code contents from the co-learner screens than in Study 1.

Table 7. Survey answers from two groups of participants who read co-learners' code contents (14 participants) and who did not read co-learners' code contents (8 participants) in Study 2. These are the mean (and standard deviation) values of the answers to the questions that measure social presence in Cocode. Answers to the questions about other learners' context (Q2), emotion (Q3), personality (Q4), and personal history (Q5) are higher in the group who read co-learners' code contents from the co-learner screens. Answers about the reality of presence (Q1) were higher in the second group, but most of the answers were positive in both groups unlike the questions about other dimensions. The Wilcoxon rank-sum test was used to test the significance of the differences in the two groups. Unfortunately, however, no significant difference was found. *n=22*, and each question used the 5-point Likert scale.

	Read Code	Not Read Code
Q1 (Presence)	4.00 (0.78)	4.22 (0.63)
Q2 (Context)	3.92 (0.73)	3.33 (0.82)
Q3 (Emotion)	3.85 (0.77)	3.11 (1.29)
Q4 (Personality)	2.00 (1.04)	1.78 (0.92)
Q5 (Personal History)	1.38 (0.62)	1.22 (0.42)

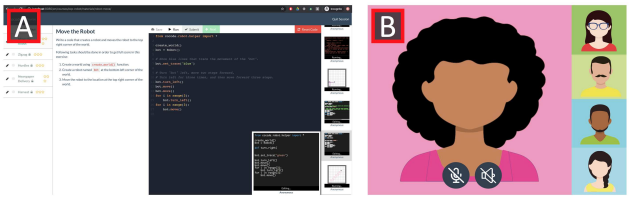

Fig. 5. (A) An overview of Cocode's user interface with co-learner screens and (B) an example of a video conferencing application's user interface. Since they both show other people on the side of the screen and allow users to focus on one of them with mouse pointer interactions, learners who use video conferencing applications in university courses may find Cocode's user interface more familiar to use. Source images by Alexandra Koch, via Pixabay. (https://bit.ly/379qYzD).

Since it is probable that participants in Study 2 may needed more help to solve exercises and copied parts of their co-learners' solutions, we compared the time spent on the exercises and code execution counts in Study 1 and Study 2. However, Table 6 shows that participants in both studies spent similar amounts of time editing code contents, and their code execution counts were also similar to each other. We could not find any difference between Study 1 and Study 2 to support the participants in Study 2 needing more help when solving the exercises.

It would be interesting to know why more participants wanted to read the code in Study 2, because we found that participants who explicitly read the code in Study 2 also felt a more social presence in Cocode than the participants who did not read the code. Table 7 shows the differences between the answers to the questions that measure social presence.

We conjecture that more participants in Study 2 read the code contents because they were more familiar with the user interface of Cocode. Because of the coronavirus outbreak, most of the courses they took that semester had lectures on a video conferencing application (Zoom[4]). This application's user interface is similar to Cocode; there is a strip of video screens on the side of the application that shows co-learners' video from their webcams, and users can click on one of the

[4]https://zoom.us/

screens to enlarge that video screen. Figure 5 shows the user interfaces of Cocode and a typical video conferencing application. Many learners in Study 2 spent a whole semester taking courses featuring lectures via a video conferencing application; thus, they might have found the co-learner screens to be more familiar and less disturbing than the participants in Study 1.

We could run more studies to support this theory as future work. For instance, we could run user studies with a group of participants who took the offline version of the CS1 class, and then spent one or more semesters entirely online to get familiar with online classes using video-conferencing applications. It would be interesting to see how this group of learners measures the social presence in the offline CS1 class and in Cocode with co-learner screens.

5 DISCUSSION

We built and evaluated Cocode with multiple user studies. However, there are limitations in this research due to the limited design of the evaluation studies and the Cocode system.

For example, we observed that the learners using Cocode feel a social presence from their co-learners; however, we do not exactly know how allowing learners to read their co-learners' code affects the learning performance of them. We also randomly selected the co-learner screens to show the learners, while there should be better ways to determine which screens to display. And also, social presence in Cocode may get stronger by allowing learners to directly communicate with each other and build interpersonal relationships. Finally, we ran our user studies with university students who only took the CS1 course; therefore, we do not know if Cocode can provide a social presence to learners with various backgrounds.

In this section, we discuss these limitations and try to find potential solutions to them.

5.1 Learning by Reading Others' Code

Cocode provides social presence to learners of programming by showing their co-learners' working environments. When co-learners' code contents are visible, they may copy others' code. Usually, *copying* has a negative meaning in programming education and it is true that copying others' code without understanding that code does not support learning at all [13, 56]. Some participants in our formative study also mentioned that it felt like they were cheating when they look at others' code while solving the programming exercises.

However, according to *social learning theory*, copying others' code can be one way of learning. Learners can improve their practical knowledge by observing their co-learners and imitate their co-learners' work to change their behavior [20]. It is common that textbooks include many examples to help learners recognize general patterns in the work, and co-learner screens in Cocode provide a similar experience to the learners. Repeated exposure to the examples can provide enough experience to learners so that they recognize the patterns in the examples and figure out which are the good examples as they become helpful resources for learning [8, 56]. Previous studies in programming education also report that example programs are one of the most beneficial resources to the learners [8, 32], and it is common for learners of programming to read code examples as a strategy for solving problems when they are stuck [34, 35].

We expect the code examples provided through the co-learner screens to be helpful for the learners, especially when the learners are taking online programming classes without any co-learners to discuss the problems they encounter. In Cocode, to prevent the learners from copying co-learners' code in all exercises without understanding them, educators can also allow learners to use co-learner screens or to reveal the co-learners' code contents from the screens on specific exercises only. For example, allowing learners to use co-learner screens in every other exercise would force them to solve at least half of the exercises by themselves.

5.2 Selecting Co-learner Screens to Display

The current version of Cocode randomly selects the co-learner screens to display while a learner is working on programming exercises. However, the co-learners' screens that are displayed to the learner should depend on the potential helpfulness of the co-learner screens to the learner. Since the purpose of co-learner screens is to help learners learn to program, there should be better ways to select when and which co-learner screens to show to the learners.

There are dimensions to quantitatively measure the activity logs, for example, the frequency of the learner's activities. Displaying the co-learner screens of more active learners may be more helpful to the learners, or it could be more helpful to display the screens from active learners and less active learners together. There are more dimensions in activity logs, such as the number of code executions that showed an error message, the time spent to solve the exercise and get a full score, or even the level of programming skills of the learners. Previous studies have shown that learners get positive effects when they are working with co-learners who are slightly better at the subject than the learners [6, 25]. It would be interesting if we could find how to select the co-learner screens that are most helpful when they are shown to the learners, and see if the findings from the offline classes also apply in online programming classes in future studies.

5.3 Communication with Co-learners

In this study, Cocode displayed all of the co-learner screens as *anonymous users*, since the IRB office approved our studies under the condition that all participants must be completely anonymous to each other. However, previous studies found that maintaining productive relations with other learners constructed over social values like trust, respect, rapport, and empathy improves the learning experience in online classes, and learners need to recognize these other learners as real people with unique personalities [28, 41, 44]. Therefore, we believe that Cocode with co-learner screens can provide a better social presence for the learners if the co-learners can be explicitly identified as specific people who are participating in the class, not just anonymous individuals. Although we could not use this feature in our studies, the implementation of Cocode allows educators to let learners choose whether or not they want to share any of their activities, share their activities anonymously, or share their activities with their identities.

In our studies, we found that the learners wanted to communicate with their co-learners. We asked the participants to give us their opinions on how we could improve the learning experience in Cocode. The most common idea from their answers was that they wanted to *interact with the other participants* (20/45), and many of them specifically mentioned that they would like to have a *chat feature* (15/45) to directly communicate with the other participants who are online at the same time. Some participants suggested using voice chat or video chat features. Although we already found in the formative studies that it is hard to get answers instantly from other learners in Cocode, we can consider adding features that allow learners to interact with other learners through explicit actions. Future versions of the co-learner screens in Cocode may allow learners to asynchronously exchange short text messages or emojis with each other; learners will be able to check the messages and reply to them when they re-visit the course website so that they can build and maintain social relations with their co-learners throughout the course of the semester.

5.4 Co-learner Screens for Experienced Programmers

In this research, we evaluated Cocode and co-learner screens with CS1 class materials and we showed that learners feel social presence more in Cocode than in other online class environments. However, we also wanted to see experienced programmers' opinions regarding co-learner screens so we conducted one more small study with the latest version of Cocode. We ran a within-subject

study with seven graduate students (5 males and 2 females, average 27.9 years old with SD=2.9) majoring in Computer Science (CS). All of the participants had one or more degrees in CS, and were asked to solve two programming exercises that resembled questions in the programming job interviews: (1) the *buildings receiving sunlight*[5] problem and (2) implementing a palindrome checker that also tells if the given string can be a palindrome by removing one letter. Participants solved randomly picked one problem with co-learner screens and another problem without co-learner screens.

According to the post-study survey, just over half of the participants (4/7) preferred to use co-learner screens if they had to solve one more problem, while the others (3/7) did not. When we asked them to give a reason for viewing the co-learner screens and explain the different experiences with and without the co-learner screens, the participants reported that reading other learners' code content gave them *hints or ideas* to solve the problems (5/7); they were *motivated* to solve the problems faster and in better ways (2/7), or their co-learners made them *improve* their own code (2/7). On the other hand, some others said that it felt like a *competition* and it gave them *pressure* (4/7), or they *preferred to solve the problems* by themselves so they did not look at their co-learners' screens (2/7). The results showed that these participants' opinions on Cocode were similar to the beginner programmers and about half of the participants preferred to use Cocode, suggesting that Cocode may have positive effects on some experienced programmers as well. We may be able to provide social awareness to the programmers working on a project, or simulate pair programming or mob programming asynchronously in future studies.

6 CONCLUSION

In this research, we introduced Cocode, a system for online programming classes that provides a social presence for learners by showing co-learner screens. Cocode provides the social presence by displaying co-learner screens that show activity logs from other learners that are collected when these learners were working on the same programming exercise in the past. These co-learner screens allow learners to work on programming exercises served on Cocode in their own time, but still feel like they are solving the exercises together with their co-learners in real-time.

The user studies showed that Cocode with co-learner screens provided more social presence to the learners than the online class with existing social features like forums, chat boxes, and live video lectures. The participants felt a significantly higher social presence from other learners in Cocode, in the dimensions of reality of the presence, context, and emotions. Although the offline class seemed to be better at providing social presence to the learners, Cocode can be used to provide more social presence in online classes. Since previous research tells us that social presence provides various beneficial effects for a learning experience, we expect Cocode to improve the learning experience in online programming classes.

Throughout the studies, we also have thought of several future directions to improve Cocode. First, we can find the method to select the most appropriate co-learner screens to display depending on the learner currently working on the programming exercise. Previous studies tell that learners learn better when studying with other learners who are slightly better than the learners; it would be interesting to see if that applies to online learning, too. We can also add features to support learners to directly communicate with each other; appropriately designed features for asynchronous but explicit communication between the learners would allow them to feel a stronger social presence in Cocode. And also, we can find another better way to hide the code contents in the co-learner screens. Instead of hiding all alphabetical characters, we can hide specific elements of the code. For example, we can hide conditional statements in the if-statements and while-loops from the

[5] https://www.geeksforgeeks.org/number-buildings-facing-sun/

co-learners' code to give learners more hints about the syntax of the programming language while the semantically important parts of the code are still hidden.

Furthermore, our study can be the first step toward online classes in various fields with social presence from the co-learners. For example, we can use the Cocode-like system to provide a social presence to the learners in math or essay writing classes, as long as the classes are based on computer-supported learning materials. Such follow-up would help the educators and learners worldwide since many learners are taking classes fully online due to the COVID-19 pandemic.

ACKNOWLEDGMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7065962).

REFERENCES

[1] Carlos Alario-Hoyos, Mar Pérez-Sanagustín, Carlos Delgado-Kloos, Mario Muñoz-Organero, Antonio Rodríguez-de-las Heras, et al. 2013. Analysing the impact of built-in and external social tools in a MOOC on educational technologies. In *European Conference on Technology Enhanced Learning*. Springer, 5–18.

[2] Mohamed Ally. 2004. Foundations of educational theory for online learning. *Theory and practice of online learning* 2 (2004), 15–44.

[3] Lucy Barnard-Brak, Valerie Osland Paton, and William Y Lan. 2010. Profiles in self-regulated learning in the online learning environment. *International Review of Research in Open and Distributed Learning* 11, 1 (2010), 61–80.

[4] Kamal Bijlani, P Venkat Rangan, Sethu Subramanian, Vivek Vijayan, and KR Jayahari. 2010. A-VIEW: Adaptive bandwidth for telepresence and large user sets in live distance education. In *2010 2nd International Conference on Education Technology and Computer*, Vol. 2. IEEE, V2–219.

[5] Frank Biocca. 1997. The cyborg's dilemma: Progressive embodiment in virtual environments. *Journal of computer-mediated communication* 3, 2 (1997), JCMC324.

[6] Hart Blanton, Bram P Buunk, Frederick X Gibbons, and Hans Kuyper. 1999. When better-than-others compare upward: Choice of comparison and comparative evaluation as independent predictors of academic performance. *Journal of personality and social psychology* 76, 3 (1999), 420.

[7] Andreas Böhm. 2004. Theoretical Coding: Text Analysis in Grounded Theory. *A companion to qualitative research* 1 (2004).

[8] Jürgen Börstler, Marie Nordström, and James H Paterson. 2011. On the quality of examples in introductory Java textbooks. *ACM Transactions on Computing Education (TOCE)* 11, 1 (2011), 1–21.

[9] Elizabeth Burpee, Cheryl Allendoerfer, Denise Wilson, and Mee Joo Kim. 2012. Why do some engineering students study alone?. In *2012 Frontiers in Education Conference Proceedings*. IEEE, 1–6.

[10] Cynthia Clark, Neal Strudler, and Karen Grove. 2015. Comparing asynchronous and synchronous video vs. text based discussions in an online teacher education course. *Online Learning* 19, 3 (2015), 48–69.

[11] Zoom Video Communications. 2021. System Requirements for Zoom Rooms. https://support.zoom.us/hc/en-us/articles/204003179-System-requirements-for-Zoom-Rooms

[12] Sarah D'Angelo and Andrew Begel. 2017. Improving communication between pair programmers using shared gaze awareness. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 6245–6290.

[13] John Dewey. 1903. Democracy in education. *The elementary school teacher* 4, 4 (1903), 193–204.

[14] Rafael Duque and Crescencio Bravo. 2008. Analyzing work productivity and program quality in collaborative programming. In *2008 The Third International Conference on Software Engineering Advances*. IEEE, 270–276.

[15] Jesus Favela, Hiroshi Natsu, Cynthia Pérez, Omar Robles, Alberto L Morán, Raul Romero, Ana M Martínez-Enríquez, and Dominique Decouchant. 2004. Empirical evaluation of collaborative support for distributed pair programming. In *International Conference on Collaboration and Technology*. Springer, 215–222.

[16] Oliver Ferschke, Diyi Yang, Gaurav Tomar, and Carolyn Penstein Rosé. 2015. Positive impact of collaborative chat participation in an edX MOOC. In *International Conference on Artificial Intelligence in Education*. Springer, 115–124.

[17] National Center for Immunization and Respiratory Diseases (NCIRD). 2020. How COVID-19 Spreads. https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/how-covid-spreads.html

[18] D Randy Garrison, Terry Anderson, and Walter Archer. 1999. Critical inquiry in a text-based environment: Computer conferencing in higher education. *The internet and higher education* 2, 2-3 (1999), 87–105.

[19] Elena L Glassman, Jeremy Scott, Rishabh Singh, Philip J Guo, and Robert C Miller. 2015. OverCode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction (TOCHI)* 22, 2 (2015), 1–35.
[20] Joan E Grusec. 1994. Social learning theory and developmental psychology: The legacies of Robert R. Sears and Albert Bandura. (1994).
[21] Philip J Guo. 2015. Codeopticon: Real-time, one-to-many human tutoring for computer programming. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 599–608.
[22] Brian Hanks. 2008. Empirical evaluation of distributed pair programming. *International Journal of Human-Computer Studies* 66, 7 (2008), 530–544.
[23] Randall S Hansen. 2006. Benefits and problems with student teams: Suggestions for improving team projects. *Journal of Education for business* 82, 1 (2006), 11–19.
[24] Noriko Hara. 2000. Student distress in a web-based distance education course. *Information, Communication & Society* 3, 4 (2000), 557–579.
[25] Pascal Huguet, Florence Dumas, Jean M Monteil, and Nicolas Genestoux. 2001. Social comparison choices in the classroom: Further evidence for students' upward comparison tendency and its beneficial impact on performance. *European journal of social psychology* 31, 5 (2001), 557–578.
[26] Nataliya V Ivankova and Sheldon L Stick. 2007. Students' persistence in a distributed doctoral program in educational leadership in higher education: A mixed methods study. *Research in Higher Education* 48, 1 (2007), 93.
[27] Shraboni Jana, Amit Pande, An Chan, and Prasant Mohapatra. 2013. Mobile video chat: issues and challenges. *IEEE Communications Magazine* 51, 6 (2013), 144–151.
[28] Benjamin Kehrwald. 2008. Understanding social presence in text-based online learning environments. *Distance Education* 29, 1 (2008), 89–106.
[29] Juho Kim, Elena L Glassman, Andrés Monroy-Hernández, and Meredith Ringel Morris. 2015. RIMES: Embedding interactive multimedia exercises in lecture videos. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1535–1544.
[30] Femke Kirschner, Fred Paas, and Paul A Kirschner. 2009. A cognitive load approach to collaborative learning: United brains for complex tasks. *Educational psychology review* 21, 1 (2009), 31–42.
[31] Karel Kreijns, Paul A Kirschner, and Wim Jochems. 2003. Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research. *Computers in human behavior* 19, 3 (2003), 335–353.
[32] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *Acm sigcse bulletin* 37, 3 (2005), 14–18.
[33] Terry Mayes and Sara De Freitas. 2004. Review of e-learning theories, frameworks and models. JISC e-learning models study report. (2004).
[34] Robert McCartney, Anna Eckerdal, Jan Erik Moström, Kate Sanders, Lynda Thomas, and Carol Zander. 2010. Computing students learning computing informally. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. 43–48.
[35] Robert McCartney, Anna Eckerdal, Jan Erik Mostrom, Kate Sanders, and Carol Zander. 2007. Successful students' strategies for getting unstuck. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. 156–160.
[36] Libby V Morris, Catherine Finnegan, and Sz-Shyan Wu. 2005. Tracking student behavior, persistence, and achievement in online courses. *The Internet and Higher Education* 8, 3 (2005), 221–231.
[37] Christophe Mouton, Kristian Sons, and Ian Grimstead. 2011. Collaborative visualization: current systems and future trends. In *Proceedings of the 16th International Conference on 3D Web Technology*. 101–110.
[38] Inah Omoronyia, John Ferguson, Marc Roper, and Murray Wood. 2009. Using developer activity data to enhance awareness during collaborative software development. *Computer Supported Cooperative Work (CSCW)* 18, 5-6 (2009), 509.
[39] Nathaniel Ostashewski, Jennifer Howell, and Jon Dron. 2016. Crowdsourcing MOOC Interactions: Using a Social Media Site cMOOC to Engage Students in University Course Activities. (2016).
[40] Glen Postle, Andrew Sturman, Francis Mangubhai, Peter Cronk, Ann Carmichael, Jacquie McDonald, Shirley Reushle, Lesley Richardson, and Bruce Vickery. 2003. Online teaching and learning in higher education: A case study.
[41] Liam Rourke, Terry Anderson, D Randy Garrison, and Walter Archer. 1999. Assessing social presence in asynchronous text-based computer conferencing. *The Journal of Distance Education/Revue de l'education Distance* 14, 2 (1999), 50–71.
[42] Stephan Salinger, Christopher Oezbek, Karl Beecher, and Julia Schenk. 2010. Saros: an eclipse plug-in for distributed party programming. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*. 48–55.

[43] Stefan Seedorf, Christian Thum, Thimo Schulze, and Lea Pfrogner. 2014. Social co-browsing in online shopping: the impact of real-time collaboration on user engagement. (2014).
[44] Christine Steeples, Chris Jones, and Peter Goodyear. 2002. Beyond e-learning: A future for networked learning. In *Networked learning: Perspectives and issues*. Springer, 323–341.
[45] Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Awareness support in distributed software development: A systematic review and mapping of the literature. *Computer Supported Cooperative Work (CSCW)* 22, 2-3 (2013), 113–158.
[46] Chih-Hsiung Tu and Marina McIsaac. 2002. The relationship of social presence and interaction in online classes. *The American journal of distance education* 16, 3 (2002), 131–150.
[47] George Veletsianos, Amy Collier, and Emily Schneider. 2015. Digging deeper into learners' experiences in MOOC s: Participation in social networks outside of MOOC s, notetaking and contexts surrounding content consumption. *British Journal of Educational Technology* 46, 3 (2015), 570–587.
[48] James Wallace. 1992. Do Students Who Prefer To Learn Alone Achieve Better Than Students Who Prefer To Learn with Peers?. (1992).
[49] Dakuo Wang. 2016. How people write together now: Exploring and supporting today's computer-supported collaborative writing. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*. 175–179.
[50] Dakuo Wang, Judith S Olson, Jingwen Zhang, Trung Nguyen, and Gary M Olson. 2015. DocuViz: visualizing collaborative writing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1865–1874.
[51] Xu Wang, Diyi Yang, Miaomiao Wen, Kenneth Koedinger, and Carolyn P Rosé. 2015. Investigating How Student's Cognitive Behavior in MOOC Discussion Forums Affect Learning Gains. *International Educational Data Mining Society* (2015).
[52] Jeremy Warner and Philip J Guo. 2017. Codepilot: Scaffolding end-to-end collaborative software development for novice programmers. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1136–1141.
[53] Jie Wei, Stefan Seedorf, Paul Benjamin Lowry, Christian Thum, and Thimo Schulze. 2017. How increased social presence through co-browsing influences user engagement in collaborative online shopping. *Electronic Commerce Research and Applications* 24 (2017), 84–99.
[54] Diyi Yang, David Adamson, and Carolyn Penstein Rosé. 2014. Question recommendation with constraints for massive open online courses. In *Proceedings of the 8th ACM Conference on Recommender systems*. 49–56.
[55] Soobin Yim, Dakuo Wang, Judith Olson, Viet Vu, and Mark Warschauer. 2017. Synchronous Collaborative Writing in the Classroom: Undergraduates' Collaboration Practices and their Impact on Writing Style, Quality, and Quantity. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 468–479.
[56] Carol Zander, Lynda Thomas, Jan Erik Moström, and Anna Eckerdal. 2020. Copying Can Be Good: How Students View Imitation as a Tool in Learning to Program. In *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.
[57] Lei Zhu, Izak Benbasat, and Zhenhui Jiang. 2010. Let's shop online together: An empirical investigation of collaborative online shopping support. *Information Systems Research* 21, 4 (2010), 872–891.

Received October 2020; revised January 2021; revised April 2021; accepted May 2021