Student Reflections on Relating Programming Concepts to Their Lives in Worked Example Videos

Dr. Kevin Buffardi, California State University, Chico

Dr. Buffardi is a Professor of Computer Science at California State University, Chico. After gaining industry experience as a user experience specialist, he earned a Ph.D. in Computer Science from Virginia Tech. His research concentrates on computer science education, with specializations in broadening participation in computing, software engineering and testing, and adaptive feedback systems.

Richert Wang, University of California, Santa Barbara

Dr. Richert Wang is an Associate Teaching Professor at UC Santa Barbara. He currently holds a joint appointment between UC Santa Barbara's College of Engineering (Computer Science Department) and College of Creative Studies (Computing), and is a faculty affiliate with UCSB's Gevirtz Graduate School of Education. He has taught various introductory and intermediate programming courses (Python, C/C++, and Java), computer networks, critical writing, math (linear algebra and applications of probability in computer science), multiplayer game systems, and computer game design.

Dr. Wang received his Ph.D. in Information and Computer Science at UC Irvine in 2011, and was a lecturer of computer science at UC Irvine before joining UC Santa Barbara in 2017. He has interned at Orange Labs (France Telecom R&D) and Google, and worked as a Software Development Engineer for Amazon.

Student Reflections on Relating Programming Concepts to Their Lives in Worked Example Videos

Kevin Buffardi
kbuffardi@csuchico.edu
Computer Science
California State University, Chico

Richert Wang
richert@ucsb.edu
Computer Science
University of California, Santa Barbara

Abstract

To broaden participation and applications of computing, programming (coding) education needs to represent a variety of contexts and uses. For five years, we have supervised undergraduate students in their creation of coding tutorials that demonstrate fundamental concepts with unique applications that relate to their lives. These "peer-instructor" videos are publicly available on YouTube[1] and also integrated into a website, *Codewit.us*[2], as open educational resources for studying and practicing coding. This paper details a qualitative study of the process, tools, and strategies students adopted to relate coding to their lives, as well as their reported impacts of participating in creating the educational videos.

We conducted two focus groups (n=7) of student content creators. In the focus groups, students reflected on their experiences participating in content ideation, receiving feedback, recording tutorial videos, and iterating on each of these activities. In addition, participants collaborated in recommending techniques to best support other students in creating educational videos that similarly relate coding to their own lived experiences. This paper highlights trends from feedback gathered in the focus groups and recommends a framework for creating coding video tutorials as an educational activity.

Introduction

Broadening participation in computer science (CS) education remains a persistent and multifaceted challenge. Meanwhile, computing is no longer confined to traditional technical domains as it increasingly shapes fields such as the arts, health, education, media, and transforms problem solving. Despite the career and societal importance of computing skills with the increasing integration of CS across disciplines (e.g. bioinformatics, "fintech" financial technology, digital humanities), students from historically marginalized backgrounds are underrepresented in CS classrooms and in the workforce[3]. These trends underscore the need not only to diversify who participates in CS, but also to evolve CS education so that its relevance, applications, and practices resonate with a broader range of students[4] and prepare them to continue innovating novel impacts via computing.

Nevertheless, many traditional educational materials emphasize a relatively narrow scope of CS applications, most commonly with math, electronics, robotics, or video games. Anecdotally, we observed how pervasive these examples are in both formal (e.g. textbooks, lecture materials) and informal learning materials (e.g. online tutorials, YouTube videos), which reinforces a limited perspective of CS and who it is for. Accordingly, our early work found that such applications align disproportionately with the interests of majority student populations—male students in particular—while many students with interests in other areas struggle to see how computing relates to their lives[5]. When introductory materials repeatedly frame CS through the same examples, they risk unintentionally signaling that computing is primarily relevant to a small subset of interests and identities.

To address this limitation, we began a line of work centered on empowering students to relate coding concepts to their own lives, interests, and lived experiences. Rather than prescribing specific "inclusive" contexts from the top-down, our approach afforded undergraduate peer instructors the agency to select examples that were personally meaningful to them and to explain programming concepts *in their own words and contexts*. The goal was to surface a wider breadth of applications and perspectives, while avoiding essentialism or tokenism by affording students the agency to self-identify what is relevant to them.

Across our previous studies, this approach led to several notable outcomes. We iteratively innovated the design of instructional videos to better showcase peer instructors' personalities, voices, gestures, drawing, and live coding[6] as they demonstrated worked examples. Worked examples are pedagogical step-by-step demonstrations and explanations of how to perform a task that enable learners to follow along before trying to model the task independently[7]. We also integrated the videos into an open educational resource (*Codewit.us*) that enables students to interactively practice along with the video[8]. The resulting peer-created content represented a substantially broader range of applications and themes than those typically found in introductory materials, spanning areas such as athletics, music and media, arts and crafts, everyday "adulting" tasks, and personal productivity[9]. Importantly, many of these applications corresponded to interests that students had previously reported as broadly appealing yet not commonly perceived as relevant to CS, suggesting that peer-generated examples can meaningfully expand students' conceptions of what programming can be.

As CS courses adopted our materials[10], teachers and students expressed enthusiasm for the peer-instructor videos. At the same time, teachers repeatedly asked for guidance on how to support students in creating similar content of their own. In particular, they sought how to help students brainstorm personal examples and how to scaffold the process of transitioning from initial ideas to coherent worked examples. This feedback revealed a gap in our understanding: while prior work focused on the outcomes and characteristics of peer-created materials, less attention had been given to the experiences of the content creators themselves and the insights they developed through the creation process.

In this paper, we address that gap by examining the perspectives of peer-instructor content creators directly. We conducted focus groups (n=7) to gain deeper insight into their experiences, with **four guiding research questions**:

RQ1 What motivations, challenges, and successes did peer-instructors experience when creating worked example videos?

RQ2 How can watching and creating worked examples motivate learning computer science?

RQ3 How can the process and tools best support students in creating their own worked examples?

RQ4 How did creating their own worked examples impact their identities and self-perceptions as computer scientists?

By highlighting the voices of peer instructors, this study seeks to inform how educators can more effectively support student-created instructional content and leverage peer perspectives to broaden participation and relevance in CS education.

Related Work

Various research studies have shown that utilizing undergraduate tutors in computing courses provides many benefits including instructional support, providing role-models for students, and improving student performance. A thorough literature review by Mirza et al. summarizes how undergraduate tutors in computing courses have been utilized, and reports on the benefits to various stakeholders including undergraduate tutors, students, instructors, and institutions[11].

Research on undergraduate tutors not only focus on student outcomes, but also on the experiences of the undergraduate tutors themselves. For example, Fryling et. al. shown undergraduate tutors reported that tutoring improved their CS knowledge, increased interest in pursuing education as a career, increased opportunities to know other students in their department, increased their appreciation of CS, and helped them with their courses[12]. Vihavainen et. al. reported various expectations that undergraduate tutors expected to obtain (learning opportunities, teaching skills, and social / interpersonal skills), and acquiring experience that can be used in their own courses and job market[13]. Shirazi et. al. utilized undergraduate tutors to lead coding camps in a Data Science Academy. Their undergraduate tutors (Data Science Academy Leaders) reported their experience increased interest in being involved in other events / communities, enhanced interpersonal skills / teaching, improved relationships with faculty, and increased interest in pursuing a Masters degree[14].

Though undergraduate tutors have shown to provide many benefits, qualitative studies focusing on the undergraduate tutoring students' experiences have been conducted where both benefits and especially challenges were discussed. Krause-Levy et. al. performed a qualitative study on tutoring in online interactions during COVID-19 instruction, and noted a large number ($>60\%$) of tutoring sessions had undergraduate tutors simply providing solutions where a low number of tutoring sessions taught debugging skills[15]. Molina et. al. provides a qualitative analysis on undergraduate tutors exploring their perceived roles (such as improving student learning and

providing socioemotional support), and various stressors impacting their ability to provide appropriate guidance (such as environment factors like student queue length and student behavior)[16].

These studies have focused on utilizing undergraduate tutors interacting with students in synchronous environments with direct interactions with students. Our work focuses on the experiences of undergraduate tutors asynchronously mentoring students via recorded video tutorials. Our undergraduate tutors' experience differs from traditional synchronous interactions in certain ways, but have shown to have common benefits as well. Our paper will discuss common and unique benefits, as well as common and unique challenges, compared to these traditional synchronous undergraduate tutor interactions. Our qualitative study will also lead to suggestions on adopting similar activities for students to obtain the benefits that the undergraduate tutors' acquired.

Methods

The project commenced as a collaboration between *California State University, Chico* (Chico State) and *University of California, Santa Barbara* (UCSB)—both United States *Hispanic-Serving Institutions*—in 2020, amid the COVID-19 pandemic. As health precautions, both institutions temporarily transitioned to online-only education. However, the circumstances reinforced the need for online learning materials that engaged students. Consequently, we adapted to the restrictions by initially hiring and training peer-instructors online, who recorded their videos in their homes using screen-recording software and a picture-in-picture video of their faces via webcam.

Peer-Instructor Content Creation.

Peer-instructors were recruited through an open call to undergraduate students. Applicants were not required to major in computer science, but were expected to demonstrate sufficient programming proficiency to explain introductory programming concepts. The selection process emphasized communication skills, clarity of explanation, and the ability to relate programming concepts to unique or personally meaningful contexts, rather than demographic characteristics or academic standing. Across multiple phases of the project, twelve peer instructors were hired. Collectively, they represented a range of academic majors, programming pathways, and levels of experience. While recruitment was not targeted by identity, each peer instructor represented at least one historically marginalized demographic, resulting in a diverse cohort of content creators.

Peer-instructors were tasked with creating short (approximately 5—10 minute) instructional videos covering introductory programming concepts in Java, C++, or Python. Concepts were selected to align with common introductory curricula, including variables, conditionals, loops, functions, data structures, and recursion. Rather than prescribing examples, peer-instructors were encouraged to select contexts drawn from their own interests, experiences, or everyday life.

To support consistency and quality, peer-instructors participated in weekly mentoring meetings with supervisors. These meetings were used to discuss topic selection, refine examples, review works in progress, and provide feedback on technical accuracy and clarity. Peer-instructors were

encouraged to live-code examples that could fit entirely on screen without scrolling and to explicitly articulate why their chosen context was *personally relevant*.

As the universities returned to in-person instruction, videos were recorded using a studio setup that integrated live coding within an IDE alongside visual annotation, gestures, and facial expressions, allowing peer instructors' personalities and explanatory styles to be visible. The studio equipment, software, and arrangement are described in detail in previous work[6]. To introduce peer-instructors to the equipment and prompt them to reflect on their pathways and motivations for studying CS, they recorded introductory "Why I Code" videos, helping contextualize their perspectives for learners.

Focus Groups.

During Fall 2025, we recruited current and previous peer-instructors to participate in IRB-approved (*IRB-2024-148*) focus groups[17] to share their insights on their experiences. We offered a $50 gift card to each volunteer who participated in a one-hour focus group session. Two focus group sessions were scheduled to accommodate the participants' schedules, where each participant attended only one session via Zoom online conference. Of the twelve eligible peer-instructors, seven (n=7, 58%) peer-instructors participated.

The focus groups followed a semi-structured format, where the researchers prepared open-ended questions, solicited responses from each individual, and facilitated follow-up questions and discussions to clarify and elaborate on the participants' feedback. Before the focus group began, the researchers emphasized that each participant's feedback was valuable—regardless of whether it corresponded or contrasted with the thoughts shared by other participants—to mitigate groupthink[18] and encourage active participation from everyone.

The focus group protocol included the following prompts to investigate each of our research questions:

RQ1 What motivations, challenges, and successes did peer-instructors experience when creating worked example videos?

- You were hired as a content creator to record videos that related coding to your life. What motivated you to apply for the position?
- A primary mission of the position was to create coding examples that were relevant to you but different from how coding concepts are traditionally taught. Please tell me about how you came up with unique ideas.
- What was the biggest difficulty you faced when coming up with ideas?
- What helped you create videos that were more personal to your life? Looking back, what would have made it easier to come up with examples that are personal?
- What would have helped you relate more ideas to your and your family's cultural traditions and identities?

RQ2 How can watching and creating worked examples motivate learning computer science?

- What kinds of examples do you think will particularly attract diverse groups of students to learn and appreciate coding skills?

- What suggestions do you have for making content that appeals to all students?
- How about students who may have otherwise not considered pursuing coding skills?
- If students in a computer science class had an assignment to make their own examples that relate coding to their own lives, what suggestions would you offer to help them formulate unique ideas?
- When coming up with these ideas in a group of students, are there approaches that would help develop ideas that are unique and innovative?

RQ3 How can the process and tools best support students in creating their own worked examples?

- When considering the technologies you used to record the videos, what best helped you convey your thoughts?
- Are there ways we could improve the technology to support communicating your examples?
- Consider the process of brainstorming ideas, writing code, and recording videos, as well as the feedback you received or revisions you made. What was most helpful in the process? Are there ways the process could be improved?
- If you were reviewing coding examples and videos from others and giving them feedback, what would you concentrate on?

RQ4 How did creating their own worked examples impact their identities and self-perceptions as computer scientists?

- Looking back at the experience of creating content to highlight coding concepts from your unique perspectives, how has it impacted your perception of coding skills?
- How has it impacted your identity as a coder or computer scientist?
- How does your experience creating coding tutorials relate to your career/life goals?

In addition to the qualitative feedback provided in response to the focus group discussion, each participant also completed an online questionnaire to quantify the degree to which they agreed with the following statement. The questionnaire included the instructions: *On a scale from 1-5, where 1 is "strongly disagree" and 5 is "strongly agree," how would you rate the following statements*:

1. Creating tutorial videos improved my programming skills
2. Creating tutorial videos improved my skills with multimedia tools
3. Creating tutorial videos improved my presentation and communication skills
4. Creating tutorial videos changed my perspective on how computer science is relatable to a diverse range of students
5. Creating tutorial videos changed my perspective on how to approach learning computer science
6. Creating tutorial videos changed my perspective on computer science education in K-12/college/and beyond
7. Creating tutorial videos changed my perspective on the instructor's point of view when teaching computer science
8. If I had access to these videos when I was first learning to code, I would feel a stronger identity as a coder

9. If I had access to these videos when I was first learning to code, I would be more motivated to pursue a career that involved coding
10. If I had access to these videos when I was first learning to code, I would feel more confident in my coding skills

After gathering our notes from the focus groups, two researchers analyzed the results. We performed thematic analysis[19] by independently coding themes and patterns for each research question and then consolidating our observations by identifying commonalities and reconciling differences to arrive at a consensus.

Results

Of the twelve eligible peer-instructors, seven (n=7, 58%) volunteered to participate in our focus groups. We held two focus group sessions to accommodate participants' schedules and after one participant had to reschedule to the other session, the resulting focus groups were coincidentally segregated by gender (3 women in one session, 4 men in the other). Participants were also split between those who served as peer-instructors at each university (4 from *UCSB*, 3 from *Chico State*). Two of the participants had served as peer-instructors during the pandemic (with remote supervision and recording at home) while the other five participants served post-pandemic and used the campuses' recording studios.

We performed thematic analysis[19] from the responses shared during the focus groups and identified themes as they related to our four research questions. To avoid over-generalizing from potential impacts of groupthink in focus groups when identifying themes, we paid particular attention to also acknowledge any dissension or divergence in feedback, even from a single member. In the following subsections, we describe the patterns and themes in the focus group responses to each of the research questions.

RQ1 *What motivations, challenges, and successes did peer-instructors experience when creating worked example videos?*

Participants described multiple motivations for applying to the content creator position, most commonly citing positive prior experiences with tutoring or similar teaching roles, a desire to reinforce their own programming fundamentals, and pragmatic considerations such as employment opportunities compatible with their current skill level. Several participants noted that the role aligned with existing interests in education, mentoring, or communication, while others emphasized the opportunity to strengthen conceptual understanding after recognizing gaps in foundational knowledge through prior coursework or interviews.

Across both focus group sessions, participants reported drawing heavily on personal interests, hobbies, and everyday experiences to generate unique coding analogies. When generating ideas for novel worked examples, they employed strategies included mapping coding concepts onto activities such as art, music, gardening, games, or for those in different majors, relating coding to their primary discipline. They reported that examples came most naturally when they were able to construct a narrative that was grounded in a personal memory. Some participants also described using external sources—through generative AI or prompting peers for ideas—to initiate rough ideas, which they then adapted into more personally meaningful examples.

Participants identified a few challenges coming up with ideas for their examples, particularly as they tried to balance creativity, conceptual accuracy, and scope. One common challenge they reported was experiencing "writer's block" when trying to generate ideas that felt personal yet directly applicable to the target concept. Likewise, they found it challenging to balance the depth of their examples so that they were appropriate for introductory programmers while also demonstrating the concept with enough clarity within the limits of time and scope. Some participants noted difficulty simplifying inherently interconnected coding concepts to try to isolate a concept as much as possible, without assuming prior knowledge (since we prioritized minimizing prerequisite knowledge so that the videos could be used in different sequences in various classrooms), as well as challenges differentiating their ideas from those of peers working on similar topics.

Participants reported that regular brainstorming meetings, peer collaboration, and informal feedback were among the most helpful supports for overcoming those challenges and developing novel, personal examples. They widely agreed that observing peers' demonstrations of work-in-progress examples was particularly effective in sparking new ideas and making their examples distinct from others. In retrospect, participants indicated that drawing more explicitly on personal memories (rather than general interests) and having additional structured prompts could have further supported personalization.

While participants generally felt encouraged to relate coding examples to cultural or familial identities, most reported difficulty doing so in practice. Responses suggested that participants more readily connected examples to hobbies and personal interests than to cultural traditions. Several participants indicated that to reflect their cultural traditions in worked examples that they would need more explicit examples and/or direction for collaborative brainstorming with peers of similar cultural backgrounds.

RQ2 ***How can watching and creating worked examples motivate learning computer science?***

When speculating about what would be effective at motivating diverse learners to connect with computer science, two themes emerged separately from the different focus group sessions. One group emphasized that an engaging presentation style, passion and charisma, and engaging storytelling were most important. In contrast, the second group emphasized that tangible, visible applications of coding——such as games, robotics, simulations, or real-world tools——would help learners see the relevance and impact of computer science; they suggested that seeing coding "in action" was described as particularly important for students who might not otherwise perceive the applicability of abstract concepts.

Participants in both focus groups agreed in maintaining high expectations for learners while providing reassurance and encouragement. Suggestions included avoiding oversimplification, adopting a peer-like instructional tone in the delivery, and investing in production quality (e.g., clear audio, visuals, and editing). Enhanced visual elements such as drawings, animations, or diagrams were frequently cited as ways to increase engagement, especially for learners with limited prior exposure to computer science. They reiterated that illustrations were often helpful in grounding abstract ideas in real life coding examples.

When asked how to support students in creating their own personal coding examples, participants

suggested structured scaffolding combined with flexibility. Recommended strategies included providing guiding prompts, encouraging observation of everyday routines, allowing time for reflection, and promoting discussion in small groups. Several participants emphasized starting with broad ideas before refining them into specific, concrete examples.

Participants viewed group work as highly beneficial for idea generation, describing a "snowballing" effect in which ideas became more creative through discussion. However, participants also voiced a strong preference for a hybrid approach in which students first work individually before sharing and refining ideas in groups. While overlapping ideas occasionally emerged, participants reported that collaboration generally helped individuals differentiate and strengthen their examples. Likewise, they acknowledged the value in peer feedback but unanimously agreed that examples were more personal when each person created their own example rather than trying to collaborate on unified examples in small groups.

RQ3 ***How can the process and tools best support students in creating their own worked examples?***

Participants identified real-time coding environments, screen recording software, and visual tools (e.g., drawing illustrations or diagrams on the studio's lightboard[6]) as particularly effective for conveying their thinking. The participants who created videos at home without the studio during the pandemic emphasized that the ability to draw and gesture would have made it easier to explain their examples. Those who had access to the studios appreciated being able to debug live, draw diagrams, or visually emphasize key code segments to clarify their communication. Meanwhile, one participant in particular valued the ability to use video editing software to remove errors (rather than re-record an entire video) and refine their presentation.

Participants described feedback——from peers, their supervising professors, and self-review——as vital to improving their videos. One participant described getting immediate feedback when collaboratively recording with a peer in the same room and reported that it was especially valuable for rapid iteration. Others emphasized that sharing their ideas and examples during weekly meetings helped them refine their examples. Unanimously, participants suggested that after an initial learning curve of acclimating to the technology and presentation style, they became more comfortable and efficient at creating videos.

When reviewing peers' videos, participants concentrated feedback on clarity, logical sequencing, pacing, and audience perspective. Many highlighted the importance of making implicit knowledge explicit, illustrating diagrams before code for complicated logic, and maintaining engagement by exuding energy, expressiveness, and clear emphasis on key concepts.

RQ4 ***How did creating their own worked examples impact their identities and self-perceptions as computer scientists?***

Across sessions, participants reported that creating instructional content strengthened their understanding of foundational concepts and language-specific details. Preparing to teach programming required deeper engagement with syntax, terminology, and abstract concepts, often leading to improved confidence in both coding and debugging. Most participants described increased confidence as coders, particularly due to their ability to explain concepts to others. Teaching was frequently framed as evidence of mastery, reinforcing their self-perception as

capable computer scientists. A minority of the participants noted that while the experience did not substantially change their confidence in technical abilities, they acknowledged that they gained communication skills that transferred well to professional contexts.

Accordingly, participants linked their experience creating videos to positive broader career development. They cited improved communication, public speaking, interviewing skills, and increased interest in education-related career paths. Several participants reported directly applying the skills they gained through video creation to their confidence and performance in technical interviews as well as communicating effectively in internships and jobs.

Questionnaire.

In response to the questionnaire, each item received overall positive responses. Each item was rated on a Likert-type scale (1 as Strongly Disagree to 5 as Strongly Agree) so that values above 3 represent agreement. **Table 1** summarizes the mean and standard deviation of participant responses to each item.

For more granularity, **figure 1** illustrates the distribution of each answer on the five-point scale. It is worth noting that only one participant answered any disagreement (1 or 2) and did so for every one of their answers. We suspect that the participant confused the scale because their ratings were contradictory to consistently positive statements during the focus groups and consequently their intention was probably to score *Strongly Agree (5)* or *Somewhat Agree (4)* instead. However, we report their scores in this paper in accordance with what they submitted to the online questionnaire. Nevertheless, the responses reflect a consensus toward agreement for every item.

Item	M	s.d.
Creating tutorial videos improved my programming skills	3.71	1.25
Creating tutorial videos improved my skills with multimedia tools	4.14	1.57
Creating tutorial videos improved my presentation and communication skills	4.14	1.46
Creating tutorial videos changed my perspective on how computer science is relatable to a diverse range of students	3.57	1.39
Creating tutorial videos changed my perspective on how to approach learning computer science	3.71	1.38
Creating tutorial videos changed my perspective on computer science education in K-12/college/and beyond	4.00	1.41
Creating tutorial videos changed my perspective on the instructor's point of view when teaching computer science	4.14	1.21
If I had access to these videos when I was first learning to code, I would feel a stronger identity as a coder	3.86	1.46
If I had access to these videos when I was first learning to code, I would be more motivated to pursue a career that involved coding	3.71	1.38
If I had access to these videos when I was first learning to code, I would feel more confident in my coding skills	3.86	1.46

Table 1: Mean and standard deviation of questionnaire items.

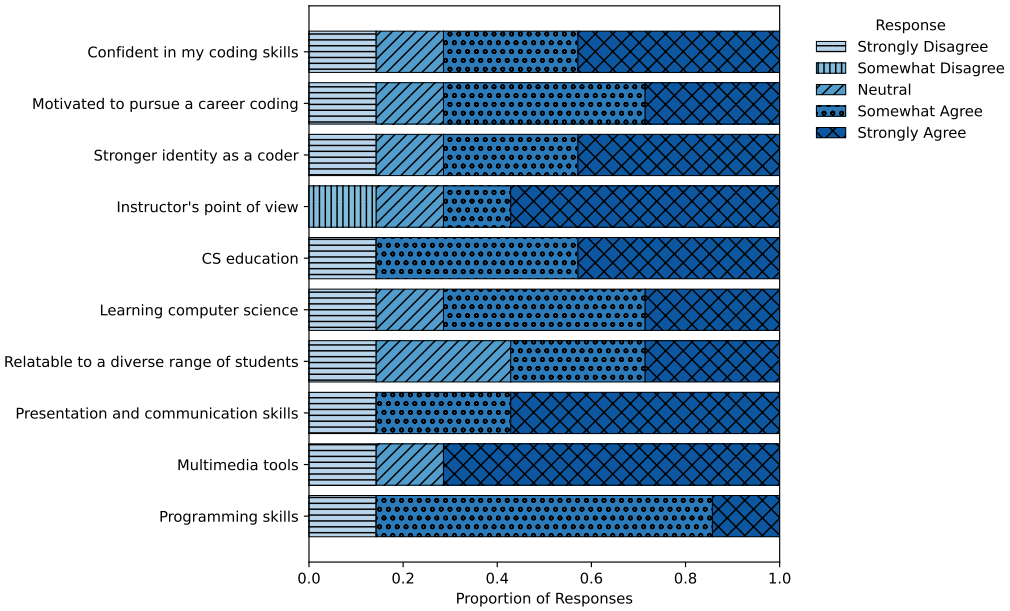

Figure 1: Distribution of questionnaire responses

Discussion

Overall, the results indicate that engaging students as peer-instructor content creators can be personally transformative and also generate teaching material that is relatable to a broader audience. Across focus groups, participants consistently described the process of designing and recording worked example videos as intellectually demanding yet rewarding, requiring them to deepen their understanding of foundational programming concepts while translating those ideas into relatable narratives.

We particularly noted that the incidentally gender-segregated focus groups arrived at distinct insights for *RQ2*. The focus group with women emphasized that personal connections via effective communication and interpersonal skills would motivate new students. On the other hand, the focus group with men underscored the importance of tangible applications of examples. We expect that both characteristics of worked examples will promote student motivation, but future work should investigate the impacts of both.

The consensus for *RQ2* and *RQ4* indicated that the experience of creating personalized worked examples helped reinforce technical knowledge while also boosting self-efficacy, identity with the discipline, and communication skills. All of these outcomes are beneficial for students. Meanwhile, *RQ1* and *RQ3* helped identify what helped facilitate their creation of worked examples and what they recommended to improve the experience. This guidance will help iterate our practices while creating new material for *Codewit.us*[6].

In accordance with the motivation for this study, we are particularly invested in using our peer-instructors' reflections to help teachers facilitate their own students benefiting from creating their own worked examples. Consequently, we recommend the following framework for scaffolding the formative experience of creating a personalized coding worked example.

Personalized coding worked example framework.

The focus group responses particularly emphasized individuals working within small groups to provide each other constructive feedback in creating individual personalized worked examples. While the resulting product may be a single deliverable, such as a recorded video or live lesson, we strongly recommend scaffolding the process. In particular, we propose the following increments of work, which include multiple opportunities for constructive feedback.

1. Ideation: connecting concepts to lived experiences.

In small groups, students should discuss which (of a collection of options) concepts each member would like to create a worked example to demonstrate. In this initial stage, it would benefit students to review the *core features* of the concepts together. For example, they should identify that *arrays* feature information that inherently have a linear order to them; likewise, *functions* feature tasks that can be abstracted how they are performed by specifying their *parameters*. After identifying the core features of their selected concept, students should consider how those core features are reflected in their lives, through their cultural traditions, daily habits, hobbies and interests, as well as their life goals.

With some time provided to generate ideas, students should prepare an explanation of how an example in their personal life maps to the concept's core features. Feedback from their peers should concentrate on clarifying context for the lived experience so that others understand the inherent connection between it and the programming concept.

2. Drafting: translating ideas to code.

After addressing feedback on ideation, students should draft an initial example that demonstrates how their lived experience is exemplified through code. They should aim to provide code to their small group with an explanation that overtly connects their lived experience with how it is demonstrated in the code. Feedback should focus on verifying that the code is functionally correct, follows best practices, and demonstrates a clear connection to the concept's core features.

3. Revision: practicing communicating the worked example.

The previous step should help students perfect the code for their examples. However, the focus groups emphasized that communicating the ideas in an example requires practice and should make good use of illustrations. In this step, students should concentrate on how to communicate their ideas by practicing with their peers. Accordingly, their group should provide feedback to help revise the explanation of the worked example.

4. Teaching: delivering the worked example.

After practicing their lesson in the previous step, students should make revisions to communicate the coding concept and how it relates to their life. Focus group feedback lauded the value of the studio technology that enabled them to draw, gesture, and live code. Ideally, students can take advantage of the technology described in our previous work[6] to create multimodal videos. However, the activity can be approximated in live, in-class demonstrations. If a computer screen is projected onto a marker board, it will afford students many of the same benefits of our studio setup, except that the lightboard in our studio enables our peer-instructors to face the audience

while a marker board will require the presenter to turn their back to the audience to draw illustrations.

Threats to Validity

This study has several limitations that should be considered when interpreting the findings. First, all peer instructors had previously completed an introductory computer science course, meaning they entered the content creation process with stronger conceptual foundations than students who might be asked to create worked examples while concurrently learning introductory material. As a result, their experiences and perceived benefits may not fully generalize to novice students encountering CS concepts for the first time.

Second, the focus group data may be susceptible to selection bias. Participation was voluntary, and all participants were successful applicants for paid peer-instructor positions, suggesting they may share characteristics such as higher confidence, motivation, or interest in teaching than the broader population of CS students. These factors may have influenced both their experiences creating content and their reflections on its impact. Further studies are needed to examine how similar activities are experienced by a more representative cross-section of students.

Finally, the scaffolded learning activity proposed in this work has not yet been implemented or evaluated in a classroom setting. Several questionnaire items asked participants to speculate about how access to relatable worked examples might have influenced their own learning, identity, or motivation. While these reflections offer valuable insight, they do not substitute for empirical evaluation. Future work should investigate the impact of such activities on students' self-efficacy, learning outcomes, and retention when integrated into CS courses.

Conclusion

This qualitative study examined the experiences of peer-instructor content creators who designed worked example videos that relate programming concepts to their own lives. Through focus groups and reflective questionnaires, we found that the process of creating personalized instructional content was both intellectually demanding and affirming, reinforcing foundational knowledge while strengthening communication skills, self-efficacy, and identity as computer scientists. Participants highlighted the importance of agency, structured scaffolding, multimodal tools, and iterative feedback in supporting meaningful connections between coding and lived experiences. Beyond generating more relatable instructional materials, the act of teaching through worked examples emerged as a valuable learning experience for the content creators themselves. By highlighting peer-instructor perspectives, this work contributes practical guidance for educators seeking to adopt student-created worked examples as an inclusive pedagogical practice and motivates future research to evaluate its impact on learning, motivation, and retention in computer science education.

Acknowledgments

This material is based upon work supported in part by the National Science Foundation (award #2315883) and by the Learning Lab, an initiative of California Governor's Office of Planning and Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Learning Lab.

References

[1] Codewit. codewit - youtube, 2026. URL https://www.youtube.com/@codewit/videos.

[2] Kevin Buffardi. codewit.us, 2026. URL https://codewit.us/.

[3] Valerie Barr. Different denominators, different results: reanalyzing cs degrees by gender, race, and ethnicity. *ACM Inroads*, 9(3):40–47, August 2018. ISSN 2153-2184. doi: 10.1145/3239261. URL https://doi.org/10.1145/3239261.

[4] Briana B. Morrison, Beth A. Quinn, Steven Bradley, Kevin Buffardi, Brian Harrington, Helen H. Hu, Maria Kallia, Fiona McNeill, Oluwakemi Ola, Miranda Parker, Jennifer Rosato, and Jane Waite. Evidence for teaching practices that broaden participation for women in computing. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '21, page 57–131, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392020. doi: 10.1145/3502870.3506568. URL https://doi.org/10.1145/3502870.3506568.

[5] Kevin Buffardi and Subhed Chavan. Is programming relevant to cs1 students' interests? *J. Comput. Sci. Coll.*, 37(1):45–53, October 2021. ISSN 1937-4771.

[6] Kevin Buffardi. Codevid studio: Coding videos with multimodal instruction. *J. Comput. Sci. Coll.*, 38(10): 26–34, April 2023. ISSN 1937-4771.

[7] Mark Ward and John Sweller. Structuring effective worked examples. *Cognition and instruction*, 7(1):1–39, 1990.

[8] Kevin Buffardi and Richert Wang. Integrating videos with programming practice. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE '22, page 241–247, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392013. doi: 10.1145/3502718.3524778. URL https://doi.org/10.1145/3502718.3524778.

[9] Richert Wang and Kevin Buffardi. Expanding applications of programming with peer-instructor video tutorials. *J. Comput. Sci. Coll.*, 41(1):33–43, December 2025. ISSN 1937-4771.

[10] Kevin Buffardi, Elena Harris, and Richert Wang. Codewit.us: A platform for diverse perspectives in coding. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, SIGCSE 2022, page 780–786, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390705. doi: 10.1145/3478431.3499398. URL https://doi.org/10.1145/3478431.3499398.

[11] Diba Mirza, Phillip T. Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. Undergraduate teaching assistants in computer science: A systematic literature review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 31–40, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361859. doi: 10.1145/3291279.3339422. URL https://doi.org/10.1145/3291279.3339422.

[12] Meg Fryling, MaryAnne Egan, Robin Y. Flatland, Scott Vandenberg, and Sharon Small. Catch 'em early: Internship and assistantship cs mentoring programs for underclassmen. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 658–663, New York, NY, USA,

2018. Association for Computing Machinery. ISBN 9781450351034. doi: 10.1145/3159450.3159556. URL https://doi.org/10.1145/3159450.3159556.

[13] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Jaakko Kurhila. Massive increase in eager tas: experiences from extreme apprenticeship-based cs1. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, page 123–128, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320788. doi: 10.1145/2462476.2462508. URL https://doi.org/10.1145/2462476.2462508.

[14] Shirin Haji Amin Shirazi, Paea LePendu, and Mariam Salloum. Data science academy: K-12 broadening participation program conducted by undergraduate students. *J. Comput. Sci. Coll.*, 40(9):62–73, April 2025. ISSN 1937-4771.

[15] Sophia Krause-Levy, Rachel S. Lim, Ismael Villegas Molina, Yingjun Cao, and Leo Porter. An exploration of student-tutor interactions in computing. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE '22, page 435–441, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392013. doi: 10.1145/3502718.3524786. URL https://doi.org/10.1145/3502718.3524786.

[16] Ismael Villegas Molina, Jeannie Kim, Audria Montalvo, Apollo Larragoitia, Rachel S. Lim, Philip J. Guo, Sophia Krause-Levy, and Leo Porter. Undergraduate computing tutors' perceptions of their roles, stressors, and barriers to effectiveness. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSETS 2025, page 1155–1161, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400705311. doi: 10.1145/3641554.3701784. URL https://doi.org/10.1145/3641554.3701784.

[17] Catherine Marshall and Gretchen B. Rossman. *Designing Qualitative Research*. Sage Publications, Thousand Oaks, CA, 3rd edition, 1999. ISBN 0803952528.

[18] Carsten De Dreu, Evert Van de Vliert, Marlene E Turner, and Anthony R Pratkanis. Mitigating groupthink by stimulating constructive conflict. In *Using Conflict in Organizations*. SAGE Publications Ltd, London, January 2026.

[19] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006. doi: 10.1191/1478088706qp063oa. URL https://doi.org/10.1191/1478088706qp063oa.