2022

Need Finding for an Embodied Coding Platform: Educators' Practices and Perspectives

Tommy Sharkey

Robert Twomey

Amy Eguchi

Monica Sweet

Ying Chun Wu

Need Finding for an Embodied Coding Platform: Educators' Practices and Perspectives

Tommy Sharkey[1,2][a], Robert Twomey[2,3][b], Amy Eguchi[4][c], Monica Sweet[5]
and Ying Choon Wu[6][d]

[1]*Design Lab, University of California San Diego, U.S.A.*
[2]*Arthur C. Clarke Center for Human Imagination, University of California San Diego, U.S.A.*
[3]*Johnny Carson Center for Emerging Media Arts, University of Nebraska Lincoln, U.S.A.*
[4]*Department of Education Studies, University of California San Diego, U.S.A.*
[5]*Center for Research on Educational Equity, Assessment, and Teaching Excellence,*
University of California San Diego, U.S.A.
[6]*Swartz Center for Computational Neuroscience, University of California San Diego, U.S.A.*

Abstract: Eight middle- and high-school Computer Science (CS) teachers in San Diego County were interviewed about the major challenges their students commonly encounter in learning computer programming. We identified strategic design opportunities -- that is, challenges and needs that can be addressed in innovative ways through the affordances of Augmented and Virtual Reality (AR/VR). Thematic Analysis of the interviews yielded six thematic clusters: Tools for Learning, Visualization and Representation, Pedagogical Approaches, Classroom Culture, Motivation, and Community Connections. Within the theme of visualization, focal clusters centered on visualizing problem spaces and using metaphors to explain computational concepts, indicating that an AR/VR coding system could help users to represent computational problems by allowing them to build from existing embodied experiences and knowledge. Additionally, codes clustered within the theme of learning tools reflected educators' preference for web-based IDEs, which involve minimal start-up costs, as well as concern over the degree of transfer in learning between block- and text-based interfaces. Finally, themes related to motivation, community, and pedagogical practices indicated that the design of an AR coding platform should support collaboration, self-expression, and autonomy in learning. It should also foster self-efficacy and learners' ability to address lived experience and real-world problems through computational means.

1 INTRODUCTION

The increasing sophistication and availability of Augmented Reality (AR) devices wield the potential to transform how we teach and learn computational concepts and coding. Instead of interfacing with keyboards, mice, and monitors while seated at a workstation, learners could engage with holographic representations of their code and the output of their code -- both of which are projected into their physical environment. Instead of clicking arbitrary buttons and toggles, users could invoke intuitive gestures and other body movements to combine, execute, and debug elements of code. In place of segmenting code through line breaks or indentations, learners could assign distinct sub-processes different shapes, sizes, and locations in space -- and even enclose some elements within others.

---

[a] https://orcid.org/0000-0001-9705-6788

[b] https://orcid.org/0000-0002-9663-0706

[c] https://orcid.org/0000-0003-1240-9228

[d] https://orcid.org/0000-0002-9382-8805

As these examples illustrate, a potential key advantage of coding in 3D physical space is the opportunity that it affords learners to leverage their existing sensorimotor experiences. This concept hearkens to Papert's notion of body syntonic reasoning (Papert, 1980), whereby young learners rely on their own sensorimotor experiences to make sense of LOGO programming tools. It is also consistent with more recent work; for instance, Fadjo (Fadjo, 2012) explored the pedagogical impact of physically embodying aspects of Scratch scripts on middle school students' own Scratch artifact construction. As an extension of this idea, a number of tangible programming environments allow users to implement and debug commands or programs by acting out commands through physical body movements (Berland, Martin, Benton, Petrick Smith, & Davis, 2013; Berland, 2016; Raffle, Parkes, & Ishii, 2004).

At the core of much of this research is the idea that abstract computational concepts, such as data, operators, and loops, are grounded in embodied representations. For instance, when computer science (CS) students describe algorithms, conditionals, and other computational structures, they frequently gesture in ways that suggest they are conceptualizing interactions with objects (Manches, McKenna, Rajendran, & Robertson, 2019). Further, it has been suggested that stimulating embodied mappings between sensorimotor experience and computational concepts can benefit CS learning. When students physically modelled or "acted out" prewritten code structures, it was found that they tended to produce code that is longer and more mathematically complex (Black, Segal, & Vitale, 2012).

Theories of embodied cognition were originally advanced in repudiation of the view that the human mind can be described in terms of computational operations exacted upon amodal representations (Barsalou, 2008). As an alternative, embodied perspectives emphasize the tight coupling between our capacities for perception and action, on the one hand, and higher-order conceptual processing, on the other. Our ability to understand and reason about abstract concepts such as time, for instance, is grounded in our experience of space (Núñez & Sweetser, 2006). Our ability to comprehend words and language, which are largely abstract, symbolic representations, is mediated by our perceptual experiences of speakers' meanings (Glenberg, Webster, Mouilso, Havas, & Lindeman, 2009). Over two decades of empirical evidence support theories of embodied cognition, ranging from evidence of mental simulations during language comprehension (Barsalou, 2009; Wu & Coulson, 2007; Zwaan, Stanfield, & Yaxley, 2002), to sensorimotor cortical activations accompanying literal and figurative meanings of words such as kick (Hauk, Johnsrude, & Pulvermüller, 2004) or *grasp* (Boulenger, Hauk, & Pulvermüller, 2009), to evidence of mirror-based empathy (Gallese, 2001) and action comprehension systems (Gallese, Fadiga, Fogassi, & Rizzolatti, 1996; G Rizzolatti, Fadiga, Gallese, & Fogassi, 1996; Giacomo Rizzolatti & Craighero, 2004). In education, the notion of embodiment as a force that can drive learning has gained traction as well – perhaps due to the inherent congruence of this idea with constructivist theories of learning, all of which include active components of learning -- or learning by doing -- as central tenets. In embodiment theory, knowledge structures are proposed to be acquired and retained more efficiently when they involve related sensorimotor input (Lindgren & Johnson-Glenberg, 2013). This principle has been illustrated both in simple practices, such as hand gestures to scaffold understanding of mathematical equations (Goldin-Meadow, Cook, & Mitchell, 2009), as well as immersive, mixed-reality environments that allow users to predict with their own body movements the trajectory of asteroids affected by planetary gravitational fields (Lindgren, Tscholl, Wang, & Johnson, 2016). Pedagogical tools and methods harnessing full-body interaction and other aspects of embodied learning have been explored in a wide range of STEM domains, including quantitative reasoning (Davidsen & Ryberg, 2017) and math (Abrahamson & Sánchez-García, 2016; Alibali & Nathan, 2012; Goldin-Meadow et al., 2009), as well as physics (Johnson-Glenberg, Megowan-Romanowicz, Birchfield, & Savio-Ramos, 2016; Johnson-Glenberg & Megowan-Romanowicz, 2017; Yoon, Elinich, Wang, Steinmeier, & Tucker, 2012), chemistry (Johnson-Glenberg, Birchfield, Tolentino, & Koziupa, 2014), and astronomy (Lindgren et al., 2016).

It has been proposed that syntonic mappings between CS concepts on the one hand, and our mental, sensory, and kinesthetic experiences, on the other, can make learning CS easier (Watt, 1998). In the view espoused by Watt, programming languages are learned more readily in cases where the structure of the language exhibits syntonicity, or resonance, with people's existing bodily knowledge and mental models. For instance, in the case of Papert's Logo turtles, children are able to take on the perspective of the turtle and act out procedures that it should undertake in order to accomplish a task (e.g. drawing a circle). Likewise, adults tend to impute

psychological characteristics to different programming elements, such as the operating system, the compiler, and so forth, and tend to reason about them as they would another human being.

In the present study, how to foster syntonic mappings through an AR/VR platform – and which mappings are the most important and the best fit – is an open design question. Through structured interviews with secondary school CS educators in San Diego County, the authors aimed to understand common challenges and pivotal needs faced by students learning to code. Are there specific coding environments, syntaxes, computational structures, or classroom activities that tend to be embraced – or conversely, that tend to elicit frustration or lead to learner disengagement? The interviews were also structured to probe methods for fostering syntonicity already in use in the classroom.

2 STUDY DESIGN

2.1 Subjects

Eight CS educators from the San Diego region in California were interviewed for the study: five secondary school teachers, two informal CS educators, and one post-secondary educator. The five teachers were all instructing either middle- or high-school CS courses at the time of the interview; two at low-SES public schools serving large numbers of under-represented students, and three at higher-SES, highly resourced private schools. The informal CS educators either taught after-school workshops or otherwise supported CS teaching outside of the classroom for ages ranging from elementary through high school. The post-secondary educator has taught at both the community college and university levels. All participants gave informed consent.

2.2 Interview Questions and Thematic Analysis

Participants were asked a series of open-ended questions in semi-structured interviews. Teachers were asked the same set of questions (see Appendix) but were allowed to speak at length and move to topics they found important, giving unprompted evaluations of tools, platforms, concepts that challenge students, and more. Interviewers asked follow-up questions pertinent to individual teacher responses to each question. The average interview length was 1 hour (range: 41 to 107 min). Interviews included questions about specific challenges that learners encounter (e.g. "What do your students struggle with when learning to code?"), classroom activities (e.g. What tasks are students solving?) and perspectives on the current state of CS education (e.g. Where do you see opportunities for improvements in programming education?); a complete list of questions can be found in Appendix A.

Participants' thoughts and opinions were turned into single sentence nodes (324 total) that were then independently coded into themes by four researchers with diverse expertise (CS Education, Cognitive Neuroscience, Data Science, and Human-Computer Interaction). The nodes were created by first manually extracting quotes from the recorded interviews. Any time the participant expressed an opinion, made an observation, or described an activity/scenario, a quote was pulled. Each quote was then paraphrased and separated into independent thoughts (e.g. 'Scratch and Blockly are both great programs' became 'Participant remarks that Scratch is a great program' and 'Participant remarks that Blockly is a great program'). A single individual paraphrased the quotes to help mitigate language biasing.

The four researchers were then given the paraphrased nodes and each individually grouped the nodes into themes. They then met to talk about their themes and resolve discrepancies; ambiguities or disagreements emerging from the integration of coding schemes were resolved through discussion. Once final themes were identified, the group also discussed them to determine if higher-level themes might be uncovered as well. Ultimately, researcher discussions identified the 47 themes of 245 labels collectively applied to single sentence nodes, as well as the six higher-level themes. Each theme, as well as its subcomponents, will be elaborated in the Results section.

3 RESULTS

Thematic Analysis revealed six primary themes labelled as: Tools for Learning, Visualization and Representation, Pedagogical Approaches, Classroom Culture, Motivation, and Community Connections (Table 1). Each theme contains sub-components, as detailed in the Table. Notably, these themes cover a continuum that extends from person-level aspects of computational concept learning to increasingly broader social connections and motivations important for CS education. As will be detailed in the following sections, our analysis revealed that learning and teaching core elements of CS, such as syntax and debugging, are fundamentally influenced not just by

the attributes and knowledge existing within the individual who is learning, but also by dynamics within and across groups of learners – and more even broadly, by dynamics within classrooms and within the learners' families and greater communities.

Table 1: Themes and higher-order themes obtained from thematic analysis of CS educators' interviews.

Tools for Learning	
● Web IDEs with minimal startup cost ● Hardware frustrations ● Forums and communities	● Interoperability between block and text code ● Languages and platforms ● Online courses
Visualization and Representation	
● Enacting metaphors for computational concepts ● Verbal metaphors for computational concepts ● Visualizing in 3D or 2D	● Visualizing concepts and problems (flow charts, memory diagrams) ● Manipulatives (e.g. deck of cards, plastic bags)
Pedagogical Approaches	
● Project-based learning ● Focus on problem solving ● Teach syntax and structure ● Storytelling ● Active learning ● Scaffolding for success ● Fostering design thinking and planning skills	● Modifying skeleton code ● Pair programming ● Peer and group feedback ● Facilitating, guiding, and modelling instead of instructing ● Culturally responsive CS ● Just in time learning
Classroom Culture	
● Making thinking visible (metacognitive awareness) ● Embracing mistakes and accepting uncertainty ● Flexible classroom organization	● Collaboration ● Identifying and responding to obstacles ● Learning community ● Fostering reflection
Motivation	
● Making meaning ● Authenticity ● Self-expression ● Fostering self-efficacy ● Student-centered approach ● Success motivates learners	● Autonomy and self-directed learning ● Patience and perseverance ● Creativity ● Frustration and impatience ● Fear of failure
Community Connections	
● Bridging disciplines ● Diversity in learners ● Access to CS Education	● Showcasing work ● Addressing lived experiences

3.1 Tools for Learning

In all interviews, participants talked about the tools they used in teaching CS to their students and in encouraging their students to learn about CS concepts and apply these concepts to creating successful computer programs. All respondents expressed enthusiasm for streamlined instructional tools and online resources on the one hand, and concern over their educational value on the other. For instance, many participants expressed favorable opinions of web-based integrated development environments (IDEs) – which can be accessed through web browsers and support remote software development using low-capacity local devices. Web IDEs were described as easy to set up and could be configured and managed by the instructor. Other participants expressed frustration with non-web-based IDEs, particularly those for physical computing, where seemingly random glitches would cause immense discouragement among students. On the other hand, one teacher spoke critically of web-based IDEs for reducing student comprehension of how libraries are imported.

Many of these web-based IDEs use graphical metaphors to augment coding. Interview participants praised Scratch and other block-based programming languages for their ability to help students focus on algorithms instead of syntax. However, many teachers also described resistance to block-based languages in older high school students, who perceive the tools as unprofessional or childish. Participant 1 (P1) further noted that as programs become more complex, block-based coding becomes "unwieldy," particularly when it comes to understanding calls between collapsed code blocks. At least two participants observed that knowledge obtained through block-based coding did not readily transfer to text-based coding systems. P8 speculates that this, "might be due to the fact that… JavaScript and Java are very similar… Whereas [block-based coding] just doesn't map well to Java – even though on the backend it is compiling to Java." P8 goes on to describe how they believe this problem might be mitigated by providing students with a visual demonstration of how blocks relate to textual code – an idea that was echoed by several of the other participants.

Finally, several interview participants relied on online resources (e.g. Code.org) or courses (e.g. Code Academy) to cover basic syntax and other concepts. They also encouraged students to exploit online forums such as Stack Overflow in order to increase their own self-efficacy. However, some educators

expressed concern about the effectiveness of these tools. For instance, P3 specifically called out the "knowledge checks" that appear in many online courses for falling short of checking for comprehension and only checking for simple recollection.

3.2 Visualization and Representation

A recurrent theme in the interviews was the importance of visualizing problems and concepts or finding other effective means of representing them. Participants spoke at length on different strategies for making coding concepts more understandable. At least four individuals used physical and collaborative (embodied) exercises to help students understand core CS concepts like function calls and parameter passing, sorting algorithms, procedural instructions and precision of language. For instance, one high school instructor organized students into groups and used sticky notes passed between groups and individuals to represent the passing of values to variables. She described how this activity evolved over time in her classroom, stating that she, "used to just draw diagrams, but think[s] having that physicality helps [students]... understand and form mental models." Participants used this activity to represent more complex concepts as well, using reciprocal movements to represent dependency or even high-level Transfer Control Protocol: "Certain students act as the routers and the other ones are… the packets… When network congestion happens, they're stressed out."

Props were also used, such as plastic bags representing variables and sticky notes placed inside the plastic bags representing different values – or a deck of cards used to demonstrate a sorting algorithm. In a different classroom, paper airplanes were thrown between students, who represented functions, in order to demonstrate the passing of information between functions. Notably, props were often crucial components to the teaching strategy: P4 went so far as to say that, "Teaching sorting and searching algorithms without cards is basically impossible." In addition to props, educators also recalled incorporating metaphors in their lectures. For instance, analogies were drawn between nested statements and Google maps or grading curves. Likewise, the hierarchical structure of HTML pages was likened to Russian nesting dolls.

With respect to planning and debugging, educators also encouraged students to use storyboards, diagrams, drawings, props, and roleplay. Many respondents found role-playing and

diagramming useful strategies for students to adopt – possibly because they helped novice coders intuitively break a problem or a project into subcomponents. Students might imagine themselves as a robot, for instance, and attempt to navigate a room only using their sense of touch. In lessons that involved programming mobile robots, analysis of the robots' behavior in 3D, physical space proved helpful. Respondents also encouraged students to problem solve by creating flow charts, memory diagrams, and other types of drawings in order to represent various states of their systems. P7 placed easels with sticky notes in different locations of the classroom and encouraged students to create "life-size" flow charts, so to speak, anchored in physical space.

3.3 Pedagogical Approaches and Classroom Culture

The bulk of most interviews centered on participants' discussion of teaching practices and methods that they felt were effective. A consistent pattern that emerged was educators' emphasis on problem-solving and planning as core abilities for novice coders to strengthen. This prioritization of design thinking and problem-solving skills was reflected across participants both in their embrace of certain established pedagogical methods and philosophies (e.g. project-based learning, active learning, iterative design, pair programming) and in their high valuation of opportunities for students to reflect, develop metacognitive awareness of their own thinking, and to give and receive peer feedback on work. A specific example: P4 believes that given a problem, students need to, "figure out how to solve the problem without even thinking about coding first, and then translate it."

In keeping with this prioritization of problem solving and planning, respondents tended to view their own role in the classroom as one of facilitating rather than instructing. At least two individuals described a preference for "storytelling" rather than lecturing. One of the high school teachers related how she works on her own project alongside students. Three of the participants described sharing a student's code with the whole class in order to elicit feedback when someone is stuck or prompting students to share their reflections on problems or lessons learned.

Additionally, multiple respondents described measures to create a culture of embracing mistakes and uncertainty rather than becoming frustrated. P6 succinctly states: "50% of this class is learning to be okay with being uncomfortable and not knowing the

answer." Respondents also recognized the importance of scaffolding for beginners in order to ensure success. P5 notes, "Having some successes early on is really important... not just something dumb, but a valuable experience is really helpful." Further, they also valued tactics such as just-in-time instruction to mitigate unnecessary frustration. P1 states, "If it takes me 5 minutes to come over and figure out what is going on - it only takes a few of those times [before the students give up on coding]."

As a corollary to the motif of facilitating rather than instructing, the corpus of interviews also revealed a prevalent pattern across educators of actively fostering collaboration between students. All respondents implemented classroom activities designed to stimulate collaboration and communication -- from group projects to working in pairs on assignments to peer reviews of code and peer teaching. Two respondents described approaches to organizing the physical components of the classroom (e.g. desks, chairs) to support collaboration. Three of the respondents explicitly described strengthening teamwork and communication skills as important achievements in themselves that are orthogonal to other aspects of class performance. Two respondents described strategies for cultivating learning communities within and beyond a classroom. For instance, through a buddy system, older students might be paired with younger ones in order to develop excitement for CS in younger age groups. Or during a lesson, students who had finished a step in an assignment would be asked to mark their name on a board so that other students would know whom to ask for help. A different teacher from a low SES serving secondary school described his use of an online forum where students succeeding in class could help the students that are struggling and asking questions.

Whereas respondents' perspectives on problem-solving, collaboration, and educators' roles in the classroom were clear and consistent, conflicting opinions were voiced on approaches to teaching syntax. On the one hand, some educators felt that classic syntax should serve as the framework for organizing an introductory course. Some responses described regular reliance on skeleton code as a basis from which students could develop their own computer programs. On the other hand, however, some respondents favored approaches that promote creativity and design thinking over a syntax-heavy curriculum. One of the private school teachers elegantly summarized this perspective in the following:

"I think that we overestimate the necessity of… learning the basics before moving on to other things... and that giving kids more freedom and latitude to try new things and jump into things they're not totally prepared to do is, I think, a really productive process. If the basics are so important then they'll find them and learn them during that project."

In line with their statement, this instructor tended to rely on web-based tutorials in order to quickly cover basic syntax so that students could devote more time to projects.

3.4 Motivation and Community Connections

Methods for attracting interest and sustaining engagement also received robust attention from all of the interviewees, along with ways to build bridges from the classroom to other communities. Analysis of their responses revealed that student motivation was closely tied to sub-themes of pedagogical approaches and classroom culture. Perhaps because respondents recognized the intrinsic motivational value of creating a successful computer program, their interviews reflected the intent to foster self-efficacy through diverse means, including opportunities for autonomy and self-directed learning. P5, P6, and P7, for instance, either shape their project assignments around the students, or allow the students to "bend the rules" for the sake of increasing engagement. Conversely, they also recognized factors that discouraged and detracted from student motivation such as the "fear of failure", frustration, and impatience. Some participants sought to mitigate this by fostering a classroom culture that supports both collaboration and learning from, rather than penalizing mistakes. Participants often described the importance of minimizing obstacles to promote this culture, often referencing unpredictable hardware problems as a major obstacle to success.

The second cluster of recurrent thematic elements involved topics related to student-centered teaching methodologies. One respondent expressed a desire for more "culturally responsive teachers" for students from demographics that tend towards non-engineering fields in order to ensure a diversity of learners in CS. Other participants reflected on the importance of encouraging self-expression and creativity and providing opportunities to create meaningful final products in order to sustain learner engagement.

Finally, the majority of respondents recognized that a third key factor driving motivation is

authenticity and connections to a broader community. Seven of the eight participants strongly advocated for expanding the scope of CS teaching to focus on 'real world' tools for solving 'real world' problems and pushing students to make an impact on a community. Two assigned projects that required students to grapple with problems faced by a specific community. Others described simple strategies such as helping students publish their apps to app stores or encouraging students to program robots in useful ways (e.g. to help with household chores or serve as a musical instrument). Another individual praised programs such as Technovation Challenge and Oncoscape as resources that can cultivate empathy in young programmers and help them to relate their coding practices to authentic problems. Importantly, some respondents noticed a positive correlation between student engagement and the applicability and real-world relevance of an assignment – particularly assignments that involved replicating popular phone apps or other familiar interfaces.

In keeping with this idea of building bridges and community connections through CS, three educators expressed a desire for the human side of coding to be more foregrounded in CS education. P5 raises the desire to, "feel more like a whole human being" in a coding environment and responds to this by having students draw on personal experience for inspiration to, "get [students] to connect with their bodies." Further, at least half of the respondents voiced interest in "hybridization across [academic] subjects" – that is, uniting elements from diverse disciplines through the coding process. Respondents used lessons that incorporated music, art, dance, foreign languages, AI, robotics, and medicine with coding.

4 DISCUSSION

Here, CS educator interviews were analyzed to better understand the underlying factors impacting secondary-level CS education, as well as the challenges teachers and students face as they engage in CS education. Developing such an understanding is important, in that it will allow for a better understanding of whether and how AR and VR technologies can be leveraged to support computational concept learning at the secondary school level. This study highlights that for young novice coders, learning to code is not a purely cognitive process -- it is governed by sensorimotor, social and emotional dynamics as well. Just as lessons on loops and the choice of a visual versus text-based programming environment bear significant weight on

learning outcomes, so do opportunities for collaboration and community membership, self-expression and autonomy, and linking lived experience to computing and the outcomes of computing. Further, successful coding is more than implementing proper syntax -- it involves planning, problem-solving, creativity, effective communication, and the ability to work in teams.

Intriguingly, a seemingly conflicted relationship was noted within educators' attitudes towards projects versus learning tools. On one hand, our participants highly valued projects and assignments that promoted authorship, agency, and authenticity – seeking to enable students in their own endeavors at the cost of having a controlled project outcome. On the other hand, participants valued platforms and IDEs with low variability and instant feedback – prioritizing control over freedom and extensibility. How can 3D embodied coding address these diverse, and sometimes conflicting, needs? We propose that this type of coding environment can make computational concepts easier to learn through syntonic mappings to sensorimotor experience. It can provide opportunities for robust collaboration that can facilitate learning with peers. Finally, it can offer a possibility of structuring 3D space in ways that support design and debugging processes. Through these features, it is proposed that learners will be able to achieve higher levels of self-efficacy more quickly and will be ready sooner to enjoy "freedom and latitude to try new things" -- to quote one of the respondents. In other words, they will be more likely to achieve a level of mastery that allows them to engage in self-expression and make connections between computing and other domains of life and academics.

4.1 Collaboration and Problem Solving

Pair programming is a widely used method geared around the affordances of traditional workstations supporting 2D coding on screens. It involves dyadic collaboration in which a driver types lines of code, and a navigator offers guidance and checks the driver's work. This approach has been shown to benefit skill acquisition in K-12 settings (Denner, Werner, Campe, & Ortiz, 2014; L. Werner & Denning, 2009) and was commended by some of the interview participants. It has been praised for aligning with social motivations of some learners who might otherwise be disinterested in CS due to a competitive masculine culture and negative stereotypes (e.g. geeks) associated with the field (L. Werner & Denning, 2009). It also helps students to learn from

their own and their peers' mistakes. Some evidence exists that females benefit from pair programming more than males (L. L. Werner, Hanks, & McDowell, 2004).

Despite these positive results, pair programming methods should be employed carefully. In a study of middle school learners working with LOGO, cases of inequity were found to emerge when one team member dominated in task-related decision-making. Additionally, some students expressed a preference for solo work because they found the frequent switching of roles and obligation to explain their choices cumbersome (Lewis & Shah, 2015). A separate study demonstrates that confident programmers tend to dislike working in pairs. An AR/VR spatial coding platform could address problems such as these by supporting more naturalistic forms of collaboration. Because AR/VR technologies involve an unbounded virtual space, teamwork could be accomplished by larger groups than dyads (teams aren't huddled around the computer screen). The common set of manipulable objects allows team members to negotiate and revise their own roles and problem-solving strategies to suit emerging contingencies of their situation. Students might experience an increased fluidity in team dynamics, where roles shift as the need arises, allowing students to work together as a unit or subdivide into asynchronous units working in parallel. All three of these approaches may benefit learning in different individuals (Maguire, Maguire, & Hyland, 2014). An ideal platform design would include the capability for different users to easily view, manipulate, and merge each other's code. It would also include mechanisms for exporting content into forms that can be shared outside of the AR/VR environment -- for the purposes of class discussion or peer feedback, for instance.

A second important concept to consider when building an AR/VR coding platform centers on problem-solving and planning. Due to the inherent grounding of their affordances in the 3D spatio-temporal world, AR and VR naturally support many of the forms of role play, work with props, and metaphorical mappings to physical objects that educators in this study described using in their classrooms. Students might model system states and dependencies through body movement rather than mental abstraction -- making coding real by walking or gesturing between different locations to ascribe intent. This idea also ties into roleplaying, where seeing is believing. Interacting with holograms (rather than the imagination) provides an important reference and grounding for understanding. Further,

the ability to situate digital storyboards, pseudocode, notes, program elements, etc. in space (e.g., attaching them to an obstacle that a robot has to avoid) provides valuable context and structure to complex and interwoven problems -- enabling kinesthetic learners and allowing students to tap into their spatial intelligence.

Coding in AR/VR might benefit creative ideation as well. A number of studies have demonstrated a positive relationship between everyday physical activity and performance on tests of creativity (Oppezzo & Schwartz, 2014; Rominger, Fink, Weber, Papousek, & Schwerdtfeger, 2020). For instance, people produced higher quality analogies or novel uses for common objects while or just after walking relative to seated controls (Oppezzo & Schwartz, 2014). Related studies have demonstrated enhanced creativity during or just after free and fluid movement versus movement along more structured paths (Kuo & Yeh, 2016; Leung et al., 2012; Main, Aghakhani, Labroo, & Greidanus, 2018; Scibinetti, Tocci, & Pesce, 2011; Slepian & Ambady, 2012). In other words, simply walking or getting basic exercise -- can bolster creativity and may benefit learning. Moreover, it appears that some forms of fluid movement may benefit creativity even more than other body movements.

3D spatial interfaces can support these and other physical activities in a far more seamless manner than the traditional workstation. Rather than restricting programmers to text-intensive development and limiting interaction to the keyboard, mouse, and monitor, spatial representations engage a range of bodily gestures (e.g., pinching, swiping, twisting) and activities (e.g., standing, walking, crouching) to enable assembly, modification, and interaction with code structures as physical forms. It is possible that the opportunities for greater ranges and quantities of body movements afforded by 3D spatial coding can support greater gains in creative approaches to problem-solving and design challenges.

4.2 Embodied Mappings for Computational Concept Learning

Educators commonly relied on physical movements and objects to make computational abstractions easier to understand. These findings extend those found by Manches et al (2019), who demonstrated that elements of code are routinely conceptualized as containers by university students in CS classes. These kinds of metaphors can become the signifiers woven into a spatial platform that visualizes variables as vessels or open boxes. Assigning a value could

involve placing the value inside the vessel. Similarly, the same study also showed how computational processes tended to be described in speech and gestures as motion along a path. This finding suggests that an ideal gesture or controller-based action for executing a command or script would involve a broad sweep of the arm that traces the trajectory of a path.

Another embodied mapping that may benefit learners is the temporal association of events and causality. When two events are experienced in close succession on a regular basis (e.g., a gust of wind and a door slamming shut), it is often inferred that the first caused the second. Likewise, during 3d spatial programming, the execution of sequential commands could be visualized in real-time together with the rendered effects of code outputs. Through this, the user could understand the relationship between each segment of code under evaluation and the resulting effect of that code on his/her creative coding experiment. During program run-time, users could pause program execution with hand gestures or their controller and use the interface to examine run-time variables, program state, alter code, and resume execution. Granted, these types of features are already available in existing debugging tools. However, it is proposed that for learners, temporal associations between the body movements that they produce to execute a specific segment of code and the ensuing output are likely more salient – and hence important for learning – than temporal associations available in 2D displays wherein progressive lines of code under execution may simply be visually accentuated through highlighting.

Finally, a fourth possible embodied metaphor centers on the mapping between physical and conceptual distance. In common experience, objects that are physically connected tend to be encountered in close proximity, while objects that are not connected can be spaced far from one another. As a metaphorical extension, distinct computational concepts, such as input and output, could be visualized in distal spatial locations. For instance, input to a function could be represented on the left side of the code segments that constitute the function, and output, on the right side, with pipeline connectors between them, as is possible in flow and node-based coding platforms. In this way, the cognitive burden of representing the architecture and operation of a function – which are often highly abstract in many programming platforms – can be offloaded through metaphorical mappings to a fully visible framework grounded in aspects of common experience, such as the flow of substance through pipes. Moreover, as users physically orient towards or move between the different spatial locations where input and output are represented, they will themselves physically enact this metaphor.

5 CONCLUSIONS

This work has revealed the importance of cultivating problem-solving and planning skills in novice coders, as well as supporting factors that facilitate behavioral and emotional engagement in CS activities, such as collaboration, autonomy, and self-expression. It has also yielded insight into common methods adopted in the classroom for making abstract CS concepts more understandable through body movements or manipulating objects. Based on needs identified through this study and the unique affordances of AR and VR technologies, we believe that an AR/VR embodied coding platform can facilitate mastery of challenging, abstract computational concepts, allowing learners to achieve higher self-efficacy and independence in their coding practice. This medium would also likely support bodily-kinesthetic learners (Gardner, 1992) and might encourage underrepresented groups that consistently report low confidence in STEM-related abilities (Margolis & Fisher, 2002; Sax et al., 2017; Wang & Hejazi Moghadam, 2017) to pursue CS educational and career pathways by lowering barriers to entry in computer science.

Although the limited number and diversity of our participants constrain the generalizability of these results, this work offers valuable insight for the development of an AV/VR coding platform by highlighting the importance of design features that foster collaboration, simplify planning and debugging, and exploit mappings between computational structure and experience of the physical world. In the future, we plan to study how students working in pairs in an embodied AR/VR coding environment learn coding computational concepts and practices together. It will be examined whether the opportunities for greater ranges and quantities of body movement afforded by 3D spatial coding can support greater gains in computational learning than traditional 2D methods. Further, because 3D embodied coding offers new possibilities for teamwork an additional research goal of this project centers on characterizing the dynamics of dyadic collaboration in the AR/VR environment as they relate to motivation and learning outcomes. Our end goal centers on determining how these types of things could be incorporated into more formal learning environments.

ACKNOWLEDGEMENTS

REFERENCES

Abrahamson, D., & Sánchez-García, R. (2016). Learning is moving in new ways: the ecological dynamics of mathematics education. *Journal of the Learning Sciences*, *25*(2), 203–239. doi:10.1080/10508406.2016.1143370

Alibali, M. W., & Nathan, M. J. (2012). Embodiment in Mathematics Teaching and Learning: Evidence From Learners' and Teachers' Gestures. *Journal of the Learning Sciences*, *21*(2), 247–286. doi:10.1080/10508406.2011.611446

Barsalou, L. W. (2008). Grounded cognition. *Annual Review of Psychology*, *59*, 617–645. doi:10.1146/annurev.psych.59.103006.093639

Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *364*(1521), 1281–1289. doi:10.1098/rstb.2008.0319

Berland, M. (2016). Making, tinkering and computational literacy. In *Makeology: Makers as Learners* (Vol. 2, pp. 196–205). Routledge.

Berland, M., Martin, T., Benton, T., Petrick Smith, C., & Davis, D. (2013). Using learning analytics to understand the learning pathways of novice programmers. *Journal of the Learning Sciences*, *22*(4), 564–599. doi:10.1080/10508406.2013.836655

Black, J. B., Segal, A., & Vitale, J. (2012). Embodied cognition and learning environment design. In D. Jonassen & S. Lamb (eds.), *Theoretical Foundations of Student-centered Learning Environments*. New York: Routledge.

Boulenger, V., Hauk, O., & Pulvermüller, F. (2009). Grasping ideas with the motor system: semantic somatotopy in idiom comprehension. *Cerebral Cortex*, *19*(8), 1905–1914. doi:10.1093/cercor/bhn217

Davidsen, J., & Ryberg, T. (2017). This is the size of one meter": Children"s bodily-material collaboration. *International Journal of Computer-Supported Collaborative Learning*, *12*(1), 65–90. doi:10.1007/s11412-017-9248-8

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, *46*(3), 277–296. doi:10.1080/15391523.2014.888272

Fadjo, C. L. (2012). *Developing computational thinking through grounded embodied cognition* (Doctoral dissertation). Columbia University.

Gallese, V. (2001). The'shared manifold'hypothesis. From mirror neurons to empathy. *Journal of consciousness studies*.

Gallese, V., Fadiga, L., Fogassi, L., & Rizzolatti, G. (1996). Action recognition in the premotor cortex. *Brain: A Journal of Neurology*, *119* (Pt 2), 593–609. doi:10.1093/brain/119.2.593

Gardner, H. (1992). *Multiple intelligences*. academia.edu.

Glenberg, A. M., Webster, B. J., Mouilso, E., Havas, D., & Lindeman, L. M. (2009). Gender, Emotion, and the Embodiment of Language Comprehension. *Emotion review*, *1*(2), 151–161. doi:10.1177/1754073908100440

Goldin-Meadow, S., Cook, S. W., & Mitchell, Z. A. (2009). Gesturing gives children new ideas about math. *Psychological Science*, *20*(3), 267–272. doi:10.1111/j.1467-9280.2009.02297.x

Hauk, O., Johnsrude, I., & Pulvermüller, F. (2004). Somatotopic representation of action words in human motor and premotor cortex. *Neuron*, *41*(2), 301–307. doi:10.1016/S0896-6273(03)00838-9

Johnson-Glenberg, M. C., Birchfield, D. A., Tolentino, L., & Koziupa, T. (2014). Collaborative embodied learning in mixed reality motion-capture environments: Two science studies. *Journal of educational psychology*, *106*(1), 86–104. doi:10.1037/a0034008

Johnson-Glenberg, M. C., & Megowan-Romanowicz, C. (2017). Embodied science and mixed reality: How gesture and motion capture affect physics education. *Cognitive Research: Principles and Implications*, *2*(1), 24. doi:10.1186/s41235-017-0060-9

Johnson-Glenberg, M. C., Megowan-Romanowicz, C., Birchfield, D. A., & Savio-Ramos, C. (2016). Effects of embodied learning and digital platform on the retention of physics content: centripetal force. *Frontiers in Psychology*, *7*, 1819. doi:10.3389/fpsyg.2016.01819

Kuo, C.-Y., & Yeh, Y.-Y. (2016). Sensorimotor-Conceptual Integration in Free Walking Enhances Divergent Thinking for Young and Older Adults. *Frontiers in Psychology*, *7*, 1580. doi:10.3389/fpsyg.2016.01580

Leung, A. K. Y., Kim, S., Polman, E., Ong, L. S., Qiu, L., Goncalo, J. A., & Sanchez-Burks, J. (2012). Embodied metaphors and creative "acts". *Psychological Science*, *23*(5), 502–509. doi:10.1177/0956797611429801

Lewis, C. M., & Shah, N. (2015). How equity and inequity can emerge in pair programming. In *Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER' ' '15* (pp. 41–50). New York, New York, USA: ACM Press. doi:10.1145/2787622.2787716

Lindgren, R., & Johnson-Glenberg, M. (2013). Emboldened by Embodiment. *Educational Researcher*, *42*(8), 445–452. doi:10.3102/0013189X13511661

Lindgren, R., Tscholl, M., Wang, S., & Johnson, E. (2016). Enhancing learning and engagement through embodied interaction within a mixed reality simulation. *Computers & education*, *95*(95), 174–187. doi:10.1016/j.compedu.2016.01.001

Maguire, P., Maguire, R., & Hyland, P. (2014). Enhancing collaborative learning using paired-programming: Who benefits? *The Ireland Journal of Teaching and Learning in Higher Education*, *6*(2), 141(1)–141(25).

Main, K. J., Aghakhani, H., Labroo, A. A., & Greidanus, N. S. (2018). Change it up: inactivity and repetitive activity reduce creative thinking. *The Journal of creative behavior*. doi:10.1002/jocb.373

Manches, A., McKenna, P. E., Rajendran, G., & Robertson, J. (2019). Identifying embodied metaphors for computing education. *Computers in human behavior*. doi:10.1016/j.chb.2018.12.037

Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. books.google.com.

Núñez, R. E., & Sweetser, E. (2006). With the future behind them: convergent evidence from aymara language and gesture in the crosslinguistic comparison of spatial construals of time. *Cognitive science*, *30*(3), 401–450. doi:10.1207/s15516709cog0000_62

Oppezzo, M., & Schwartz, D. L. (2014). Give your ideas some legs: the positive effect of walking on creative thinking. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *40*(4), 1142–1152. doi:10.1037/a0036577

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. dl.acm.org.

Raffle, H. S., Parkes, A. J., & Ishii, H. (2004). Topobo: A constructive assembly system with kinetic memory. In *Proceedings of the 2004 conference on Human factors in computing systems'' - CHI '04* (pp. 647–654). New York, New York, USA: ACM Press. doi:10.1145/985692.985774

Rizzolatti, G, Fadiga, L., Gallese, V., & Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Brain Research. Cognitive Brain Research*, *3*(2), 131–141. doi:10.1016/0926-6410(95)00038-0

Rizzolatti, Giacomo, & Craighero, L. (2004). The mirror-neuron system. *Annual Review of Neuroscience*, *27*, 169–192.
doi:10.1146/annurev.neuro.27.070203.144230

Rominger, C., Fink, A., Weber, B., Papousek, I., & Schwerdtfeger, A. R. (2020). Everyday bodily movement is associated with creativity independently from active positive affect: a Bayesian mediation analysis approach. *Scientific Reports*, *10*(1), 11985. doi:10.1038/s41598-020-68632-9

Sax, L. J., Lehman, K. J., Jacobs, J. A., Kanny, M. A., Lim, G., Monje-Paulson, L., & Zimmerman, H. B. (2017). Anatomy of an enduring gender gap: the evolution of women's participation in computer science. *The Journal of higher education*, *88*(2), 258–293. doi:10.1080/00221546.2016.1257306

Scibinetti, P., Tocci, N., & Pesce, C. (2011). Motor creativity and creative thinking in children: the diverging role of inhibition. *Creativity research journal*, *23*(3), 262–272. doi:10.1080/10400419.2011.595993

Slepian, M. L., & Ambady, N. (2012). Fluid movement and creativity. *Journal of Experimental Psychology: General*, *141*(4), 625–629. doi:10.1037/a0027395

Wang, J., & Hejazi Moghadam, S. (2017). Diversity Barriers in K-12 Computer Science Education: Structural and Social. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education'' - SIGCSE '17* (pp. 615–620). New York, New York, USA: ACM Press. doi:10.1145/3017680.3017734

Watt, S. (1998). Syntonicity and the psychology of programming. *PPIG*.

Werner, L., & Denning, J. (2009). Pair programming in middle school. *Journal of Research on Technology in Education*, *42*(1), 29–49. doi:10.1080/15391523.2009.10782540

Werner, L. L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female computer science students. *Journal on Educational Resources in Computing*, *4*(1), 4–es. doi:10.1145/1060071.1060075

Wu, Y. C., & Coulson, S. (2007). How iconic gestures enhance communication: an ERP study. *Brain and Language*, *101*(3), 234–245. doi:10.1016/j.bandl.2006.12.003

Yoon, S. A., Elinich, K., Wang, J., Steinmeier, C., & Tucker, S. (2012). Using augmented reality and knowledge-building scaffolds to improve learning in a science museum. *International Journal of Computer-Supported Collaborative Learning*, *7*(4), 519–541. doi:10.1007/s11412-012-9156-x

Zwaan, R. A., Stanfield, R. A., & Yaxley, R. H. (2002). Language comprehenders mentally represent the shapes of objects. *Psychological Science*, *13*(2), 168–171. doi:10.1111/1467-9280.00430

APPENDIX

Interview Questions

1. Who are you teaching?
2. What is your background? (age range, location, expectations, previous knowledge)
3. How long is your course? (hours / week and number of weeks)
4. What tasks are students solving? (are they creating visualizations, data structures, etc..)
5. Walk us through one or two example situations?
6. What do your students struggle with when learning to code?
7. Do you see a difference in computer languages with respect to ease of learning? Ex : Python, Java, C#, C++
8. How comfortable are your students with the language? How skillful are they? How quickly do they learn?
9. Is there a learning curve / what is the shape of that curve? Can you draw it?
10. Is there a noticeable difference between reading and writing code?
11. What key concepts do you cover (would you cover) in a brief (5-8 session) introduction to code and computational thinking?

12. Where do you see opportunities for improvements in programming education?
13. Imagine an ideal tool for coding - what would it look like? What would it do?
14. How do you tackle motivation and keeping the students engaged?
15. Have you heard of Active Learning? Do you use active learning approaches?
16. If not, is there any reason?
17. Do you use any libraries or tools to help with the learning process?
18. What metaphors do you use to help your students understand concepts?
19. Are you aware of any spatial metaphors or representations that you tend to use?
20. Do you use any embodied teaching methods?
21. What kinds of gestures do you tend to use?
22. When you are teaching, how do you use space, your body, and props to communicate concepts?