International Society of the Learning Sciences

Making Computer Science Concepts Visible and Virtually Tangible through Creative Coding in Virtual Reality

Ying Choon Wu, ycwu@ucsd.edu, UC San Diego
Amy Eguchi, a2eguchi@ucsd.edu, UC San Diego
Robert Twomey, rtwomey@ucsd.edu, UC San Diego
Mayumi Otsuki, mayumirowan@gmail.com, Coastal Academy

Abstract: Embodied Code is a visual programming language in virtual reality (VR). It introduces novices to fundamental computing concepts and immersive game engines through hands-on creative coding. Unlike traditional creative coding toolkits, this system harnesses the visuospatial and kinesthetic affordances of VR to engage users in embodied computer science learning. Coders are afforded considerable flexibility in placing, rearranging, and manipulating elements of code (nodes and connectors) and its output such that space and movement can be leveraged as organizational and conceptual scaffolds. Further, assembling nodes and connectors is guided by two simple principles – input versus output and events versus data. These design principles were adopted to foster analogical mappings between physical experiences of working with code and output in an immersive virtual space and perception and action in the real world. Further, they were purposed for exploring different levels of coding abstraction in classroom use.

Introduction and Background

The processes of acquiring and using knowledge are not isolated to the mind. Individuals rely on their sensorimotor systems in socio-culturally mediated ways to engage the material structure of their immediate surroundings as they formulate and reason about abstract concepts (Erny, 1992; Fofang et al., 2021; Goodwin, 2000), solve problems (Thomas & Lleras, 2009), and communicate and co-ordinate (Wu & Coulson, 2007, 2015). Building from this idea, the embodied learning movement explores knowledge formation through scaffolds that encourage learners to draw mappings between their subjective somatic experience and the conceptual structure of the content to be learned. For instance, reading comprehension might be enhanced in children who act out the story narrative (Skulmowski & Rey, 2018). In computer science (CS), Papert's Logo turtle (Papert, 2020) is a clear example of a platform that helps learners draw a connection between the commands they use to manipulate the turtle (e.g., forward, right, and left) and their own body movements and social interactions.

The present paper explores how embodied mappings that support CS learning can be exploited by coding in virtual reality (VR) using a new creative coding platform, Embodied Code (EC), developed by the authors for novice secondary school coders (Lay et al., 2024; Twomey et al., 2022). Through the platform's node-based interface, programming is accomplished by loading and connecting nodes, each of which has a set of input and output ports carrying events (green) and data (blue) (Figure 1). The choice of a node-based framework was motivated by the outcomes of a need-finding study. Interviews with secondary school CS educators in the San Diego, California region revealed a consistent demand for tools supporting planning, problem solving, and creativity during computer programming, whereas learning the syntactic idiosyncrasies of different programming languages was largely viewed as an impediment to mastering the more critical higher-order abilities (Sharkey et al., 2022). For this reason, EC is designed to serve as a rapid prototyping tool, directing a learner's focus to the level of visual planning, code structure, and program flow. It hides node implementation (in Python) behind the visual front end – however, new nodes can be flexibly created through an online integrated development environment (IDE) supported by the Embodied Code Server.

Embodied Code draws inspiration from popular creative coding systems such as puredata (pd) and p5.js. It represents a creative coding toolkit in the sense that it prioritizes making/doing/creating with computational tools, supports self-paced learning, demonstrates applications to key creative domains (e.g., sound, kinetics, 3D graphics), and is user-extensible through community contributions. It is also inspired by 2D block-based platforms for novices, such as Scratch, However, Embodied Code differs from existing 3D implementations of block-based coding, such as Blockly VR (Hedlund et al., 2023) and Cubely (Vincur et al., 2017) – not only in its focus on creative coding, but also along several key parameters of user interaction design. For instance, rather than porting the flat logic of stacked blocks onto 2D working planes situated in a 3D environment (BlocklyVR), EC allows for unconstrained, fluid arrangement and re-arrangement of nodes. Further, rather than simply stacking code elements (Cubely), EC accomplishes the control flow through ports and wires that express code execution and data

transmission. As will be explored in the remainder of this paper, these features are important for supporting the emergence of embodied understandings that build from mappings between physical experience and CS concepts.

Space as a Medium for Conceptual Mapping

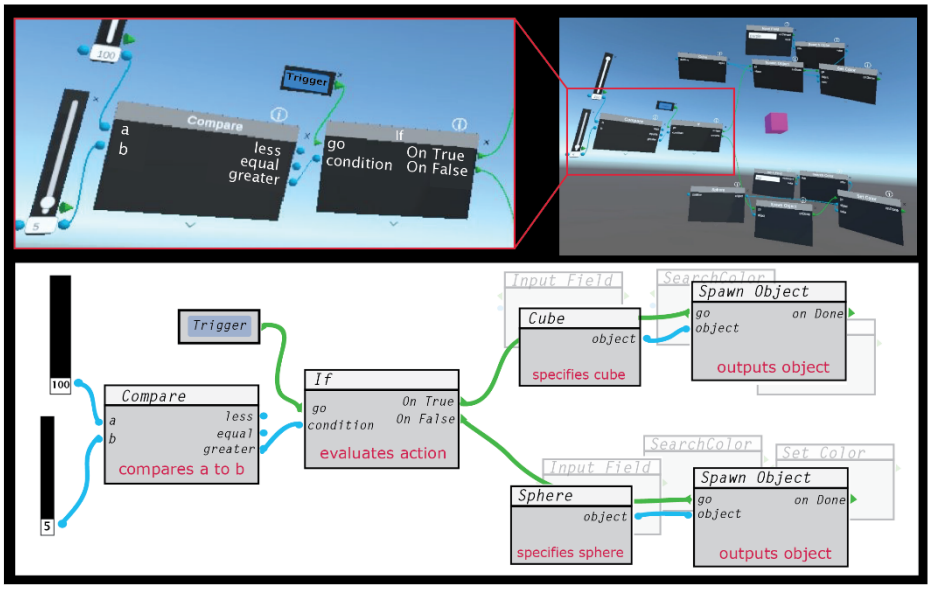
A significant challenge to learning computer programming is the abstract and invisible nature of many computational operations – a challenge encapsulated by the concept of a *notional machine* (Sorva et al., 2013). Within CS tools for novices, many approaches to making runtime dependencies and processes explicit rely on visual metaphors. For instance, in one approach, classes in Java were visualized as filing cabinets, and objects as file folders (Gries & Gries, 2002). Analogously, PlanAni (Sajaniemi & Kuittinen, 2003) is a system for visualizing variables and their roles in object-oriented programs. Fixed variables, for example, are represented as carved stone tablets, whereas variables that step through a sequence of values are represented by footsteps with their current and adjacent values displayed.

Some approaches adopt animations that increase interactive capacity. Examples include OGRE, in which objects and variables were manifested as small 3D shapes on a plane, and the assignment of values to variables was visualized as the travel of cylinders along connecting pipes (Milne & Rowe, 2004). Even more interactive systems involve users in various forms of role-play, such as creating a program by simulating the functions of a desk clerk using tools (e.g, a calculator, a stack of paper, a clipboard with worksheets) or controlling virtual avatars that stand for objects in an object-oriented program (reviewed in (Sorva et al., 2013).

Like these predecessors, Embodied Code is designed to support transparency of process through its visual and interaction interfaces, but it does so in a more flexible fashion that leverages the affordances of 3D space. Figure 1, for example, depicts code for a simple program that compares two values and spawns a purple cube if value [a] is greater than value [b] and a yellow sphere otherwise. Input is specified by means of the sliders, and the program is executed by pressing the trigger button. Upon execution, the two slider values are compared. The program is organized such that nodes representing code executed when the comparison evaluates as true are clustered in the upper split of the network, whereas nodes for code executed if false are clustered on the lower portion. In this way, the metaphor of branching structure commonly ascribed to conditional statements is manifested within the arrangement of nodes. Although it may be an obvious point, the authors would like to underscore that the operation of a conditional statement inherently does not involve any physical branching – this term is simply invoked in CS to benefit understanding. Through a branching visualization, it is hoped that novice users will find it easier to draw conceptual mappings between their past experiences with branching configurations (e.g. tree limbs, roads, rivers) and conditional control of command execution. Further, beyond mappings based on visual parallels between the arrangement of nodes and other branching structures, interaction with the nodes through the VR controllers may facilitate this proposed conceptual mapping process as well. For instance, constructing the branching arrangement illustrated in Figure 2 necessitates reaching to upper and lower portions

Figure 1

A program in Embodied Code. Screen capture from VR (top) and schematic view (bottom).

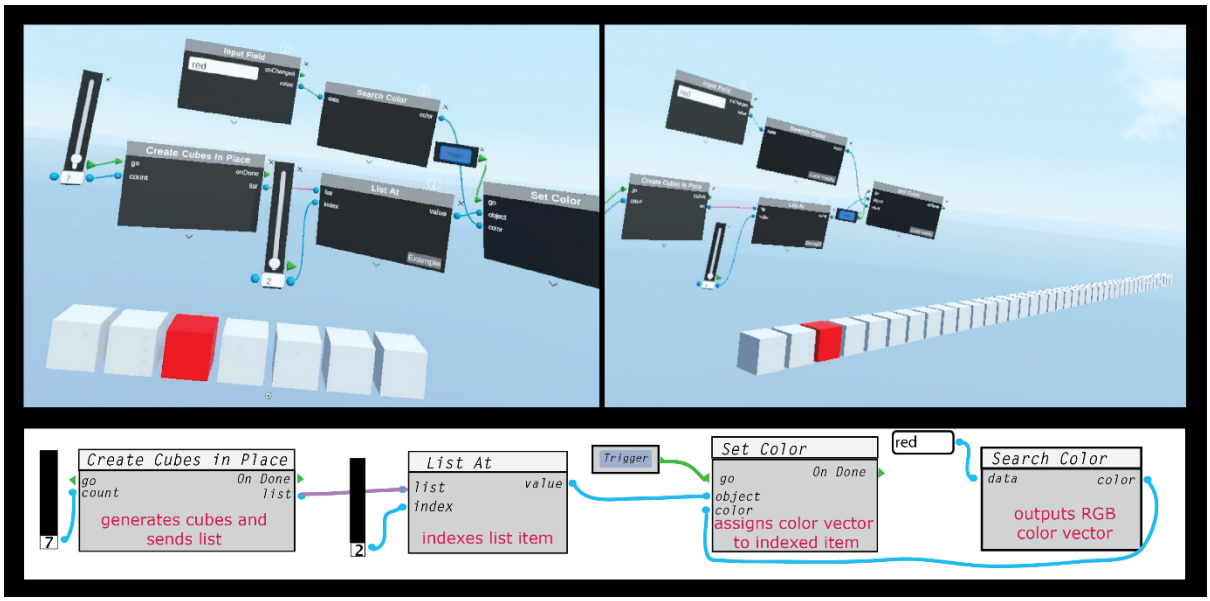


of the virtual space, reinforcing the parallel between spatial distance separating the paths and functional differentiation in the execution chains.

It should be noted that the arrangement of nodes in Figure 1 is not fixed. Users are free to fluidly configure elements of their code in whatever ways make sense to them. This design feature was adopted to support personalized and emergent conceptual mappings (Twomey et al., 2022), as the process of manipulating nodes and exploring possible patterns of organization can yield new understandings or support existing ones that are idiosyncratic to a specific user. For instance, in our coding workshops, we have observed how users may place nodes with homologous functions (e.g. specifying a cube versus specifying a sphere) adjacent to one another, expand or reduce the size of certain nodes, or place some nodes that they are currently working with the foreground while pushing others into away into distil space for later.

In addition to possibilities afforded by the flexible manipulation of nodes, embodied conceptualization is also supported through interactions with the output of code. Figure 2 depicts a simple program wherein a user-specified quantity of cubes is spawned in a row, and the cube indexed by the user-specified input to the List-At node (in this case, 2) changes to the color defined in the input field (in this case, red). This program was created to help users to grasp the concept of lists and indices. Building from common approaches to drawing illustrations of lists or arrays as rows, columns, or matrices of cells, each cube in the output represents an indexed item on a list, and the red color represents the value of the current index. By manually adjusting the value of the two slider inputs, the user can dynamically lengthen or reduce the list and change which cube receives red highlighting. List length and index magnitude are metaphorically represented as horizontal extension in space, with index 0 associated with the left-most cube from the perspective shown, and progressively increasing indices associated with successive cubes situated further and further away. By conflating space with numeric value in this interactive visualization, it is anticipated that a user will find it easier to draw mappings between the abstract notion of lists and personal perceptual and somatomotor experiences of movement over distance. Further, the conflation of index specification (2) and the outcome of index specification (the highlighted red cube) is expected to help users to build a conceptual model of indexing around the common perceptual experience of salience through color contrast – that is, in computer programming, selecting an item on a list by its index value is akin to perceiving an object in a collection when it stands out due to its salient visual properties.

Figure 2.
A list is manifested as an extended series of cubes (top). Schematic of code for specifying red cube (bottom)



Demo: Cutting across Levels of Coding Abstraction

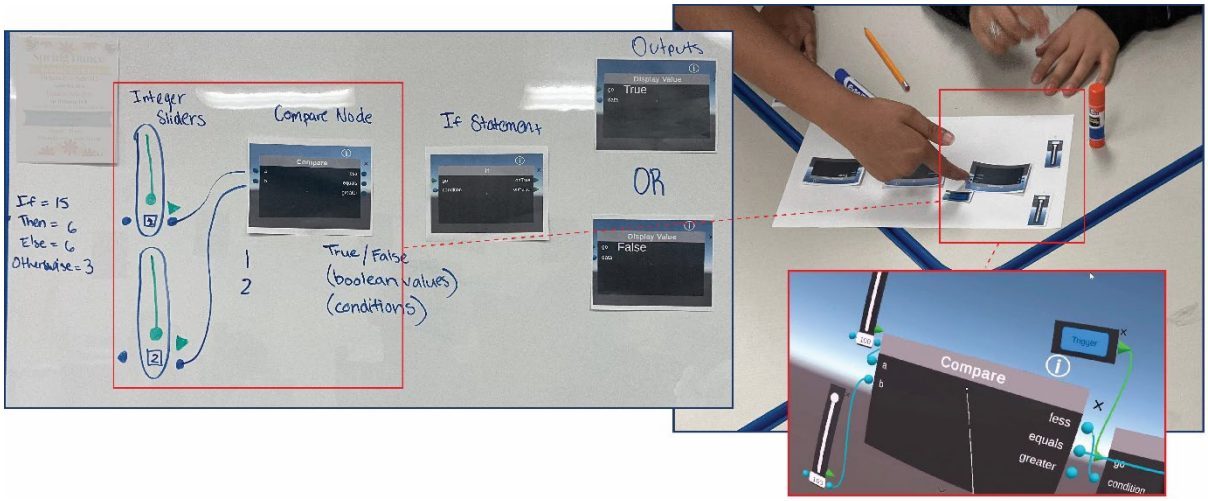
This demo will offer hands-on exploration in VR of the examples described above. It will also feature paper-based Parsons problems that were implemented in a series of lessons using Embodied Code as part of an introductory high school Computer Science class. Over the course of fifteen consecutive school days, sixteen ninth graders worked in pair programming dyads, creating progressively more complex code capable of spawning

and transforming objects, evaluating conditional statements, and iterating over items on a list. As the final project of the unit, they designed and programmed their own immersive scenes, which included a snowman, a sandcastle, a tower, and more (see (Lobo et al., 2025)).

Because the platform is built around only two elements of code – namely, nodes and connectors – each with a simple input/output structure centered on events and data – we found it easy to align lessons and concepts explored in VR with complementary unplugged approaches that allowed students to examine the same concepts through different forms of engagement outside of the headset. For instance, unplugged classroom activities included discussions of simplified code networks (Figure 3) using images of nodes printed on paper and taped to a whiteboard. Students also completed Parsons problems by arranging and gluing printed cut-outs of nodes on paper and drawing the connections between event and data ports (Figure 3). Additionally, role-playing activities were adopted at the outset of each new topic. For conditionals, students were asked to stand in a row and execute an if-else statement through their actions (e.g., "Step forward if you are wearing black; else, remain in place"). For lists, students were asked to line up and sort themselves by index number or perform other actions according to their index. These unplugged activities themselves are embodied learning experiences in a physical space, designed to support students' understanding of the CS concepts they explore in the VP headset.

This multi-modal approach allowed us to explore the same computational principles through different levels of coding abstraction (Waite et al., 2017). Role-playing exercises were developed to support the formation of higher-order representations of conditional statements and indexed lists as forms of logic. Whiteboard work and Parson's problems served to highlight the algorithmic instantiation of that logic, drawing focus to the architecture of the requisite nodes and principles for combining them. Finally, complementary work inside the headset gave prominence to running (and debugging) the code. In the future, it would be useful to develop approaches that combine role-playing and VR-based exercises more explicitly.

Figure 3.
Conditional statements explored in different modalities and at differing levels of abstraction.



Conclusion

By rendering computer programs and their output as virtually tangible objects in 3D space, Embodied Code creates opportunities for merging physical and conceptual play with computational skill acquisition. In the examples outlined above, coders' engagement with CS concepts is mediated through engagement with objects, movement, and locations that can foster the emergence of analogical understandings that are grounded in perception and action. Two primary design principles that support this mapping process are the prioritization of streamlined coding elements and possibilities for their flexible organization and arrangement. This emphasis on open-ended simplicity may seem to run counter to the view that richer perceptual experiences can support deeper embodied learning through more complex mental mappings (Black et al., 2012). However, it is important to note that Black et al advanced this postulation in response to observations of students engaged in the study of physics. Our own work with Embodied Code gives rise to the possibility that embodied learning in different disciplines may be better served by different sets of learning design principles.

References

Black, J. B., Segal, A., Vitale, J., & Fadjo, C. L. (2012). Embodied cognition and learning environment design. In *Theoretical foundations of learning environments* (pp. 198–223). Routledge.

Erny, P. (1992). Lave (Jean).—Cognition in practice. Mind, mathematics and culture in everyday life. *Revue Française de Pédagogie*, *98*(1), 122–122.

Fofang, J. B., Weintrop, D., Moon, P., & Williams-Pierce, C. (2021). Computational bodies: Grounding computational thinking practices in embodied gesture. *Proceedings of the 15th International Conference of the Learning Sciences-ICLS 2021.*

Goodwin, C. (2000). Action and embodiment within situated human interaction. *Journal of Pragmatics*, *32*(10), 1489–1522.

Gries, P., & Gries, D. (2002). Frames and folders: A teachable memory model for Java. *Journal of Computing Sciences in Colleges*, *17*(6), 182–196.

Hedlund, M., Jonsson, A., Bogdan, C., Meixner, G., Ekblom Bak, E., & Matviienko, A. (2023). BlocklyVR: Exploring Block-based Programming in Virtual Reality. *Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia*, 257–269. https://doi.org/10.1145/3626705.3627779

Lay, R., Bhutada, R., Lobo, A., Twomey, R., Eguchi, A., & Wu, Y. C. (2024). Embodied Code: Creative Coding in Virtual Reality. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2*, 1926–1926. https://doi.org/10.1145/3626253.3635428

Lobo, A., Eguchi, A., Twomey, R., & Wu, Y. C. (2025). Balanced Creative Coding for Motivation and Learning Transfer. *Proceedings of the 56th ACM Technical Symposium on Computer Science Education*, *2*, 2 pages. https://doi.org/10.1145/3641555.3705134

Milne, I., & Rowe, G. (2004). Ogre: Three-dimensional program visualization for novice programmers. *Education and Information Technologies*, *9*, 219–237.

Papert, S. A. (2020). *Mindstorms: Children, computers, and powerful ideas*. Basic books.

Sajaniemi, J., & Kuittinen, M. (2003). Program animation based on the roles of variables. *Proceedings of the 2003 ACM Symposium on Software Visualization*, 7. https://doi.org/10.1145/774833.774835

Sharkey, T., Twomey, R., Eguchi, A., Sweet, M., & Wu, Y. C. (2022). Need Finding for an Embodied Coding Platform: Educators' Practices and Perspectives. *CSEDU*, *1*.

Skulmowski, A., & Rey, G. D. (2018). Embodied learning: Introducing a taxonomy based on bodily engagement and task integration. *Cognitive Research: Principles and Implications*, *3*(1), 6. https://doi.org/10.1186/s41235-018-0092-9

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education*, *13*(4), 1–64. https://doi.org/10.1145/2490822

Thomas, L. E., & Lleras, A. (2009). Swinging into thought: Directed movement guides insight in problem solving. *Psychonomic Bulletin & Review*, *16*(4), 719–723. https://doi.org/10.3758/PBR.16.4.719

Twomey, R., Sharkey, T., Wood, T., Eguchi, A., Sweet, M., & Choon Wu, Y. (2022). An Immersive Environment for Embodied Code. *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 1–4. https://doi.org/10.1145/3491101.3519896

Vincur, J., Konopka, M., Tvarozek, J., Hoang, M., & Navrat, P. (2017). Cubely: Virtual reality block-based programming environment. *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, 1–2. https://doi.org/10.1145/3139131.3141785

Waite, J., Curzon, P., Marsh, W., & Sentance, S. (2017). K-5 Teachers' Uses of Levels of Abstraction Focusing on Design. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, 115–116. https://doi.org/10.1145/3137065.3137068

Wu, Y. C., & Coulson, S. (2007). How iconic gestures enhance communication: An ERP study. *Brain and Language*, *101*(3), 234–245.

Wu, Y. C., & Coulson, S. (2015). Iconic Gestures Facilitate Discourse Comprehension in Individuals With Superior Immediate Memory for Body Configurations. *Psychological Science*, *26*(11), 1717–1727. https://doi.org/10.1177/0956797615597671

Acknowledgements

This study was supported by the National Science Foundation (award #IIS-2017042).