An Immersive Environment for Embodied Code

Robert Twomey
Johnny Carson Center for Emerging Media Arts, University of Nebraska-Lincoln
rtwomey@unl.edu

Tommy Sharkey
Computer Science and Engineering, University of California San Diego
tsharkey@ucsd.edu

Timothy Wood
Institute for Neural Computation, University of California San Diego
t2wood@ucsd.edu

Amy Eguchi
Education Studies, University of California San Diego
a2eguchi@ucsd.edu

Monica Sweet
Center for Research on Educational Equity, Assessment, and Teaching Excellence, University of California San Diego
msweet@ucsd.edu

Ying Wu
Institute for Neural Computation, University of California San Diego
ycwu@ucsd.edu

ABSTRACT

The increasing sophistication and availability of Augmented and Virtual Reality (AR/VR) technologies wield the potential to transform how we teach and learn computational concepts and coding. This project develops a platform for creative coding in virtual and augmented reality. The Embodied Coding Environment (ECE) is a flow-based visual coding system designed to increase physical engagement with programming and lower the barrier to entry for novice programmers. It is conceptualized as a merged digital/physical workspace where spatial representation of code, the visual outputs of the code, and user interactions and edit histories are co-located in a virtual 3D space.

CCS CONCEPTS

• **Human-centered computing** → Human computer interaction (HCI); Interactive systems and tools; Human computer interaction (HCI); Interaction techniques; • **Applied computing** → Education; Interactive learning environments.

KEYWORDS

visual programming languages, extended reality, embodied cognition, programming tools

ACM Reference Format:
Robert Twomey, Tommy Sharkey, Timothy Wood, Amy Eguchi, Monica Sweet, and Ying Wu. 2022. An Immersive Environment for Embodied Code. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '22 Extended Abstracts), April 29–May 05, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3491101.3519896

1 INTRODUCTION AND MOTIVATION

It has been theorized that learners' abilities to understand and reason about functions, algorithms, conditionals, and other abstract computational concepts stem in part from more fundamental sensorimotor and perceptual experiences of the physical world. Our own work, for instance, has revealed that computer science (CS) educators incorporate a wide range of metaphors grounded in tangible experience into their lessons on computational concepts, such as demonstrating sorting algorithms with a deck of cards or the transfer of information between functions by throwing paper airplanes [10]. Our long-term research goals center on the question of how a coding platform that supports these types of embodied conceptual phenomena can make learning to code a more intuitive process, and how this alternative modality of embodied experience may boost engagement with and understanding of code for some beginning coders.

This line of work continues a decades-long effort to incorporate bodily knowledge derived from sensorimotor experience into coding logic. In the 1970's, Papert's LOGO system [6] allowed children to learn foundational computational concepts by programming basic movements of a "turtle" in real space. In related work, a reconfigurable system known as Boxer [1] represented code objects in the form of 2D boxes that could be arranged and manipulated on the screen, allowing users to make use of space and relationships in computationally meaningful ways.

Today, visual programming languages (VPLs) are broken into roughly two groups: node-/flow-based environments (e.g. PureData, Max/MSP, cables.gl, Unreal Blueprints, Matlab, and others), and block-based environments (e.g. Scratch, Blockly, Alice). These languages continue to be structured to leverage visuo-spatial metaphors towards making learning to code easier.

The Embodied Coding Environment (ECE) presented here differs from these existing 2D platforms – and even existing 3D platforms such as Facebook Horizon (connecting virtual objects to code) and MITs Reality Editor (connecting physical objects to code) [2] – by allowing users to leverage not only rich spatial metaphors, but also kinesthetic and proprioceptive ones to ground abstract computational concepts in more familiar sensorimotor experiences. For instance, if a user conceptualizes loops as having a circular structure, it is possible in the ECE to create and save gestures that match this conceptualization, and to annotate relevant sections of code with these gestural representations. Similarly, if the user conceptualizes the control flow of a computer program as forward progress through space, they can arrange nodes, connectors, and

CHI '22 Extended Abstracts, April 29–May 05, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9156-6/22/04.
https://doi.org/10.1145/3491101.3519896

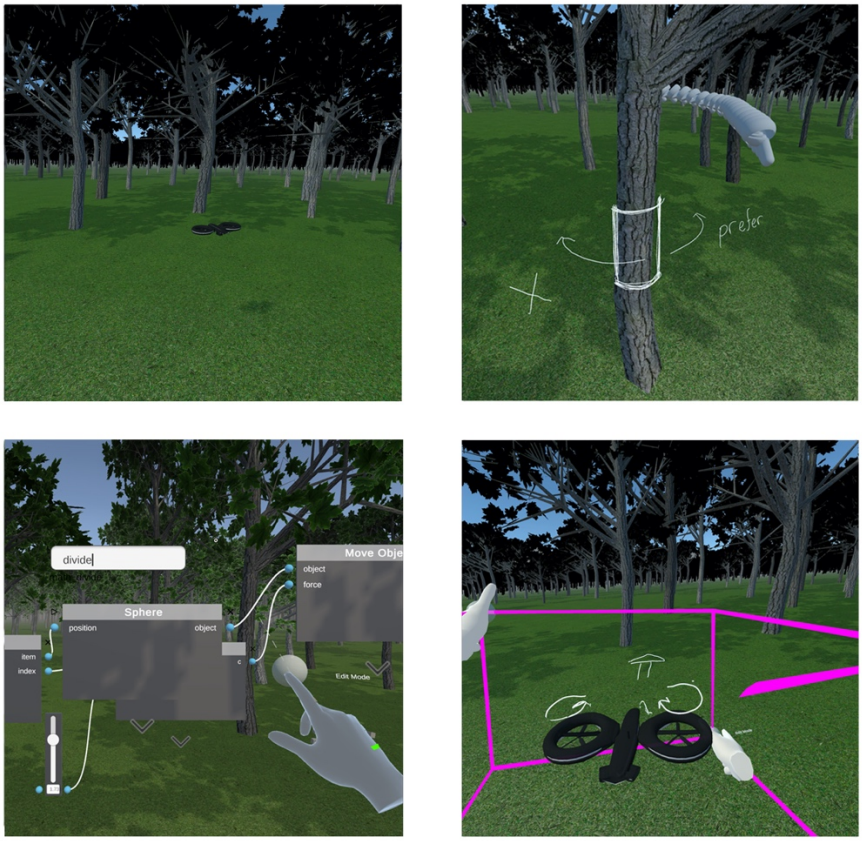

Figure 1: System features (clockwise from top left): Loading a virtual problem space; the Annotation System showing both drawings and gestures; the 3D selection tool; the programming interface and search box.

other elements of code to match this mental model. In engaging these dimensions of somatic experience, our project relates to more explicitly tangible and embodied interfaces: embodied performance environments such as Timothy Wood's SEER [9], and hardware and object-based tangible code interfaces such as littleBits and reacTable [4].

Moreover, unlike other coding platforms, the ECE allows users to exploit features of their virtual and (eventually) physical environment in order to contextualize their code in meaningful ways. For instance, variables can be organized on a bookshelf or table – in keeping with the idea that CS students conceptualize elements of code as tangible objects [5]. Or in the case of programming mobile robots, code and annotations related to the desired actions of the robot can be anchored to the locations where those actions are expected to take place. Utilizing space in these ways to make aspects of code easier to understand may engage some of the same sensorimotor processes thought to make memory palaces an effective memorization technique [7].

Below, we describe our prototype platform, its design, implementation, and key features that serve as contributions, including 1) a novel visual-spatial XR representation of coding allowing immersion in the problem and design spaces, 2) whiteboarding/annotation tools situated in a shared environment with code activities, and 3) gesture and movement paths for the direct specification of program instrumentation and data. We also detail our future plans for this project and research studies exploring its potential impact on high school computer science education, for both students and teachers. Ultimately, this project explores new roles for visual-spatial representations of code and harnesses the affordances of embodied interfaces to scaffold coding and computational thinking.

2 EMBODIED CODING ENVIRONMENT

The Embodied Coding Environment (ECE) currently runs on the Oculus Quest 2 and allows the user to write code both through the connection of nodes and via custom python (IronPython interface to UnityEngine's C# runtime environment) scripting. While the move to 3D (from 2D programming environments) can be justified simply by affording more space to display code, the foundational principle behind the ECE is focused on creating a programming environment that utilizes the inherent and embodied affordances of being a body in space.

2.1 Using the Embodied Coding Environment

When running the Embodied Coding Environment (ECE), users are presented with either an empty virtual workspace or the site/environment of their problem space (which currently must be programmatically generated). Consider a user who is confronted with the task of programming a drone that must navigate through a forest. When entering the ECE, the user will first load a forest environment and a drone (see Fig. 1) – which can be considered a representation of the problem space.

The user then walks around the forest, making mental notes of where the user wants the drone to fly to, and how many trees are in the way. The control algorithm may seem deceptively simple: fly straight toward the target, veering left/right to avoid trees before returning to the straight-line path. But as they begin to role-play this path in the problem space, they come to the realization that they themselves do not adopt this approach when walking through a forest - they often walk far off course to reach a space clear of trees, allowing them to speed up and reach their destination more quickly overall, despite costing them more time initially.

With this revelation, the user begins drawing arrows between the trees and making a diagram of their algorithm. They rely on the ECE Annotation System to produce multimedia notes using drawings, audio, and even gestures in space (see Fig. 1). Some of their annotations are next to the drone, describing power to the motors, some annotations are next to trees and point to the spaces between them.

As the design solidifies, the user begins programming the first part of the algorithm - getting the drone to fly forward. They use their controller to open a search box, spawning pre-made 'nodes,' or code functions that modify inputs and produce outputs (see Fig. 1). The user clicks on these inputs and outputs to connect them with a virtual string (this is like flow-based programming mentioned previously). They want to add a custom function to help synchronize the drone's motors, so they create a new node, press a button to edit its code, and implement their own python script with inputs and outputs. They connect it to the other nodes and can reuse it by searching for it with the search bar from earlier.

The user spatially separates the code into logical groupings by simply grabbing nodes and moving them into position. They then use the 3D selection tool (see Fig. 1) to select the nodes in the first step of their algorithm. Once selected, the ECE allows them to attach the nodes to the annotations made earlier. The user hides/collapses the nodes into their drawing and description of the first part of their algorithm and moves onto the subsequent portions of the algorithm.

Eventually the user comes across a difficult behavior - making the drone bank or tilt slightly when turning. This behavior is complex to program in a traditional environment, involving fine tuning that may prove time consuming. It is, however, easy to demonstrate with the hand and surprisingly easy to translate this embodied motion into code for the drone. The user begins recording a gesture annotation and performs the banking behavior (see Fig. 1). They select and use this gesture as input into the custom node they made earlier. In this way, they specify the drone behavior through their movement and use that specification directly as data in their code.

As shown, the ECE allows the user to think through their problem space, design in-situ in that problem space, and then cluster their code in space and fold it into the representation of their design. As they hover over annotations representing elements of their design, they can quickly look through the code tied to that annotation, tapping into their memory of the spatial layout of elements. Their annotations act as code comments in this way, and as usable input into code in the case of the banking problem.

2.2 Key Features

The Embodied Coding Environment is comprised of a set of tools: annotation tools that include both drawing and recording behaviors, smart selection tools, a smart search/command bar, a text editor with syntax highlighting, simple grouping and movement tools for organizing elements in space, and the ability to save projects to the cloud. Of these features, three are critical to the advancement of an *embodied* coding environment: immersion in the problem space, situated whiteboard/annotations for both designing and structuring code, and user movement/gestures as data and program input.

2.2.1 Immersion in Problem Space. Programmers must contend with three distinct spaces - the problem space (where the problem exists), the design space (where a solution is planned/designed), and the programming space (where code is created in line with the design). Programming environments typically focus on the programming space to the exclusion of the other spaces. The ECE experiments with an alternative where all three spaces co-exist. In the drone-in-a-forest example, programmers are immersed in a virtual forest with a drone (problem space); they can draw anywhere (design space); and place code anywhere (programming space). Many complex relationships arise by contextualizing each space within the others, but two of note are:

- The user's code is contextualized - that is, rather than organizing code in directories, code can be organized by its relative position in space. Code to control the drone's motors can be tied to the drone motors; code to identify trees appears by the trees; etc. ECE changes the metaphors we use to organize our code. Rather than placing code in a centralized location, a user can distribute it throughout the problem space.
- Users can interact with their environment - situating themselves in the problem to glean insight that might be locked away in the embodied experience of the problem space. This is exemplified in section 2.1 when the user realizes a better algorithm for avoiding trees by noticing their own movements in their problem space - learning from their embodied role-playing experience facilitated by the ECE.

2.2.2 Annotation System. To afford user-expression, the Embodied Coding platform allows for an abstract Annotation System where 'annotations' take the form of 3D drawn lines, hand gestures, spoken comments, quick 3D modeling, and more. Free form diagramming or whiteboarding plays a key role in understanding and breaking down problems as well as working through the design of algorithms to solve those problems. In its most basic form, the Annotation System supports these processes with diverse modes of expression - focusing on letting users express their ideas through

simple drawings, through movements, through 3D objects, voice, or whichever means is most natural to the user. The goal is to let them manifest their ideas into the world in whatever way they choose, not to enforce a proper method (like drawing flowcharts).

Importantly, annotations aren't passive in the ECE and can play two different active roles in the system. First, annotations serve to organize code. Rather than using files and directories, code collapses into annotations (like collapsing a function into a function header comment). The user then remembers what code is where both by recognition of the annotation and by the juxtaposition of the content with its surroundings. The second role that annotations play is as code input. The code might use the position, path, rotation, timestamps, duration, velocity, audio, or any combination of properties of the annotation. This information can be used as a direct map - controlling the position of a 3D model with the positions and timestamps from an annotation, or indirectly - using positional changes to ease between two colors. In this way, a programmer can directly specify behavior that may be time consuming to program.

2.2.3 User Movement and Direct Specification. Our system facilitates direct specification, which builds on direct manipulation [8] from historic programming (e.g. SketchPad) and CAD (e.g. AutoCAD) interfaces, and further harnesses the embodied affordances of contemporary immersive platforms. In our platform, we can use spatial signals from hand and controller-tracking to directly select locations in space, to directly specify movements, and control signals and other parameters through gestural time series input. Controller and hand movements can be recorded and visualized as annotation data within the ECE. These data can be linked as input to programming nodes and used for a variety of purposes. In the drone-in-a-forest example, the user specifies the subtle behavior of the drone banking into its turns. Movement data can also be input to programming nodes in real time, allowing the user to build an interactive instrument where the user's direct specification of movement controls a computational process with similarities to the visual art focused 2D platform Dynamic Brushes [3].

3 EVALUATION AND FUTURE WORK

The Embodied Coding Environment (ECE) currently exists as a prototype for exploration into allowing space and the body to play a more pivotal role in the programming process. We plan to conduct a series of studies to learn more about how students use the ECE, with goals of improving the platform, advancing our understanding of how individuals teach and learn computer science, and improving students' interest, engagement, skills, and comprehension of computer science. Our studies will employ mixed methods, including both quantitative and qualitative studies exploring individual features as well as the system as a whole. For example, we are planning a quantitative study for each of the described features comparing how adding/removing that feature impacts the user's understanding of the problem, design of a solution, and implementation of that solution, along with a qualitative analysis comparing how the use of the ECE changes student's approach to programming.

At the same time, the ECE is constantly evolving and expanding. Future and current work includes the move to Augmented Reality and the real-time control of physical robots rather than virtual ones. Additionally, moving collaborative programming (like pair-programming) into a virtual collaboration environment would allow collaborative coding to move past physically collocated dyads (constrained by the number of people that fit around a computer screen). This would allow for new ways to distribute sub-tasks, visualize differences (spatial diffs), and integrate multiple branches of code development. Lastly, to further the physicality of code, we want to expand the system to allow for physics-based representations of code - where functions behave more like gears in a larger mechanism.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant #2017042.

REFERENCES

[1] Andrea A. diSessa and Harold Abelson. 1986. Boxer: a reconstructible computational medium. *Commun. ACM* 29, 9 (September 1986), 859–868. DOI:https://doi.org/10.1145/6592.6595

[2] Valentin Markus Josef Heun. 2017. The reality editor: an open and universal tool for understanding and controlling the physical world. PhD diss. Massachusetts Institute of Technology. Retrieved January 13, 2022 from https://dspace.mit.edu/handle/1721.1/114072

[3] Jennifer Jacobs, Joel Brandt, Radomír Mech, and Mitchel Resnick. 2018. Extending Manual Drawing Practices with Artist-Centric Programming Tools. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, New York, NY, USA, 1–13. Retrieved January 13, 2022 from https://doi.org/10.1145/3173574.3174164

[4] Sergi Jordà, Martin Kaltenbrunner, Günter Geiger, and Ross Bencina. 2005. The reacTable. In *Proceedings of the International Computer Music Conference (ICMC)*.

[5] Andrew Manches, Peter E. McKenna, Gnanathusharan Rajendran, and Judy Robertson. 2020. Identifying embodied metaphors for computing education. *Comput. Hum. Behav.* 105, (April 2020), 105859. DOI:https://doi.org/10.1016/j.chb.2018.12.037

[6] Seymour Papert. 1980. *Mindstorms: children, computers, and powerful ideas*. Basic Books, New York.

[7] Anco Peeters and Miguel Segundo-Ortin. 2019. Misplacing memories? An enactive approach to the virtual memory palace. *Conscious. Cogn.* 76, (November 2019), 102834. DOI:https://doi.org/10.1016/j.concog.2019.102834

[8] Ben Shneiderman. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, 33–39.

[9] Timothy R. Wood. 2021. Embodied Worldmaking. PhD diss. UC Santa Barbara. Retrieved January 13, 2022 from https://escholarship.org/uc/item/66h114tg

[10] Ying Wu, Tommy Sharkey, Robert Twomey, Timothy Wood, Amy Eguchi, and Monica Sweet. Need Finding for an Embodied Coding Platform: Educators' Practices and Perspectives. In *14th International Conference on Computer Supported Education (CSEDU)*.