Title

Bringing static code to life: The instructional work of animating computer programs with the body

Journal

Proceedings of International Conference of the Learning Sciences, ICLS, 2(2018-June)

ISSN

1814-9316

Authors

Flood, VJ
Deliema, D
Abrahamson, D

Publication Date

2018

Peer reviewed

Bringing Static Code to Life: The Instructional Work of Animating Computer Programs With the Body

Virginia J. Flood, David DeLiema, and Dor Abrahamson
flood@berkeley.edu, david.deliema@berkeley.edu, dor@berkeley.edu
University of California, Berkeley

Abstract: In this preliminary report, we propose a previously unidentified role that instructors' gestures may play in helping students evaluate existing computer code. We find that instructors use gesture to animate processes encoded in the static inscriptions of computer programs in order to make invisible, dynamic phenomena perceptible to students. Our findings contribute to a better understanding of the embodied instructional work of teaching programming.

Introduction

The role of the body for communication in science and mathematics instruction is now a central focus in the learning sciences, with numerous investigations appearing over the last twenty years (e.g., Alibali et al., 2014; Hwang & Roth, 2010; Kress et al., 2001; for a review see Singer, 2017). However, studies examining the nature of teachers' embodied, discursive instructional tactics in computer science instruction are rare (Grover & Pea, 2013). In this paper, we report on a novel function of teachers' bodies in programming instruction—*animating processes encoded in the static inscriptions of computer programs*. We present two examples of how instructors use gesture to make dynamic phenomena perceptible that are essential to successfully comprehending and evaluating computer code.

Animating processes encoded in static inscriptions

Teaching programming is challenging (Milne & Rowe, 2002) and students face a number of difficulties both in comprehending and creating programs (Robins, Rountree, & Rountree, 2010). Many innovative tools have been developed and studied to make programming more accessible (e.g., block-based programming languages), but more research is needed to understand how instructors use discursive tactics to make programming concepts and practices intelligible (Grover & Pea, 2013). While there is substantial evidence that students use bodily resources to reason and communicate as they program (e.g., body syntonicity—Fadjo et al., 2009; Papert, 1980), we know little about the communicative role teachers' bodies play during programming instruction.

Case studies by Kwah and Goldman of a high school robotics instructor have shown that pedagogical gesture plays an important role in guiding students' *program creation* (Kwah, 2013; Kwah & Goldman, 2011). The focal instructor they study used gestures to diagram programming concepts for students based on metaphorical and iconic image schemas. He also shifted points of view while gesturing (between first-, second- and third-person), providing opportunities for students to recognize when ideas generalized beyond particular situations. The authors conclude that teachers' use of embodied communicative resources is a vital yet critically understudied dimension of computer science instruction.

In this study, we complement Kwah and Goldman's work by exploring additional ways programming instructors use multimodal resources (e.g., gaze, gesture, posture, talk, object manipulation, etc.). In particular, we focus on resources used to help students interpret code in *existing* programs (i.e., in *program comprehension*). Experienced programmers can evaluate code to make swift, accurate predictions about how a computer will interpret and execute the instructions. However, these same lines of symbolic notation create a perplexing perceptual field for newcomers, with a multitude of potentially relevant features to attend to. To impart disciplinary ways of perceiving phenomena, experienced practitioners and instructors rely on a variety of bodily practices to separate signal from noise for newcomers (Goodwin, 1994; Lindwall & Lymer, 2008; Stevens & Hall, 1998). For example, a mathematics tutor may deny visual access to irrelevant features of a graph (Stevens & Hall, 1998) or an archeologist might enhance focal features in the soil by outlining them (Goodwin, 1994).

In programming, however, shaping how students perceive programs cannot be readily accomplished by merely highlighting or hiding lines of code. When evaluating programs, instructors must help students to "see" dynamic future processes that have no immediate spatio-temporal correlates in the static list of inscribed instructions that is present. Instructors can show students code, but from its inscription alone, it is impossible for students to tangibly appreciate the *active event* of executing this passive list of instructions—i.e. *the processes* encoded in the instructions. Instead, we propose that in order to highlight these invisible processes of code, instructors must make them perceivable and quasi-present to students by *conjuring* them through gesture (Nemirovsky, 2012).

We note that this challenge is similar to one presented by mathematics. At first, mathematics, too, only seems accessible to the senses through inscriptions. However, Nemirovsky and colleagues argue that mathematicians and students use their hands and bodies as key resources for "bringing to life" the static symbols and inscriptions of mathematics, making dynamic processes perceptually available to both animator and audience, and allowing for the collaborative imagining of possibilities (Nemirovsky, 2014; Nemirovsky & Smith, 2013). Inspired by Nemirovsky's work in mathematics, we set out to investigate how instructors use similar resources to make invisible, dynamic processes perceptible in static inscriptions of code.

Analytical approach to investigating instructors' use of multimodal resources

Our example episodes come from an 18-hour video corpus of an eight-week-long, Saturday coding program at an urban nonprofit STEM learning center. In the program, students create emojis using Pixelbots, an in-house-developed programming environment (Figure 1) and were instructed by local undergraduate computer science majors.

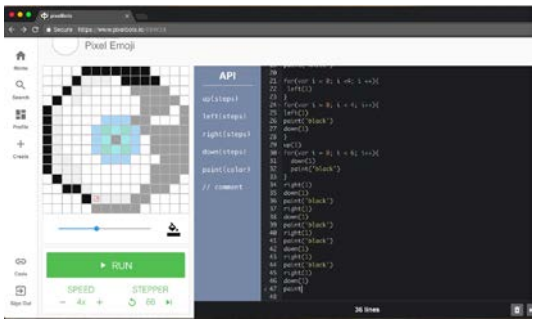


Figure 1. In Pixelbots, students write code to move a square-painting pixelbot around a grid to create an emoji like this eyeball.

We examined this corpus to locate instructional sequences where depictive gestures occurred during explanations about existing programs. Following Streeck (2008), we consider embodied activity *depictive* if it provides a construal that bears recognizable perceptual similarity to objects or processes in the world. From this initial collection of instructional episodes, we selected two examples to present that we believe provide insight into how instructors animate static inscriptions of computer code for students.

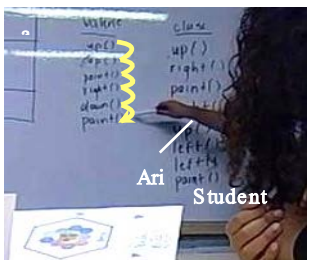
Our multimodal microanalysis of video is inspired by ethnomethodological conversation analysis (Mondada, 2012). Using ELAN we annotated episodes frame-by-frame to determine co-occurring segments of talk and embodied activity. This allowed us to richly characterize the interactional resources instructors use to communicate programming concepts. Video episodes were subject to group analysis to mitigate threats to reliability and validity.

Two examples of animating processes hidden in static computer code

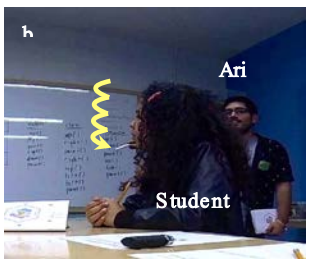
1. Bringing flow of control to life

Our first episode occurs in Ari's class as students learn to evaluate and write code to move a "pixelbot" that paints grid-squares (see Figure 1). Ari writes students' code on the board for the class to evaluate. As they examine different examples together, Ari explains that shorter code is more desirable because a program with fewer lines runs faster.

During this explanation, Ari makes the dynamic *flow of control* in the program salient for students, elaborating his verbal description of the process with his hands (Figure 2). Perceiving different control structures in a program (like sequence, selection, and repetition) is an essential component of evaluating a program (K-12 CSF, 2016) that students must be trained to recognize.



Ari: As it's going it's gonna read line by line by line by line. Does anyone remember what we called that last time?



What that was called when we were reading line by line by line by line by line? Started with an S. Any bells? That's okay. Sequence.

Figure 2. (a) Ari, the instructor, performs a series of jumps with his hand over code written on the board to animate the sequential *flow of control*. (b) He turns to students and re-creates this gesture away from the board while asking students if they remember what this phenomenon was called. *Highlighted text shows speech that co-occurs with the gesture depicted.*

To make the order in which statements are executed visible, Ari uses an environmentally coupled gesture (Goodwin, 2007) to laminate a series of curved jumps with the inscribed statements, embodying the dynamic

sequential flow of control (Figure 2a). Then, he repeats this gesture in the air (Figure 2b) as he asks students for its name. Using his hand to animate the static code on the board, Ari is able to make a dynamic phenomenon temporarily present for students to consider.

2. Illuminating the dynamic process of accessing data structures

In our second example, a student is working in Pixelbots with his instructor, Dex. They are examining some code that will randomly select a value (a color) from an array. Dex realizes the student is unclear on how data is retrieved from the array.

Arrays are a type of variable that can store multiple *values*—e.g., "green", "blue," and "red." Values are accessed by referring to their position, or their *index*, in the list: e.g., 0, 1, or 2. The array Dex and the student consider is declared on the computer screen as follows:

```
var list = [`green`, `blue`, `red`]
```

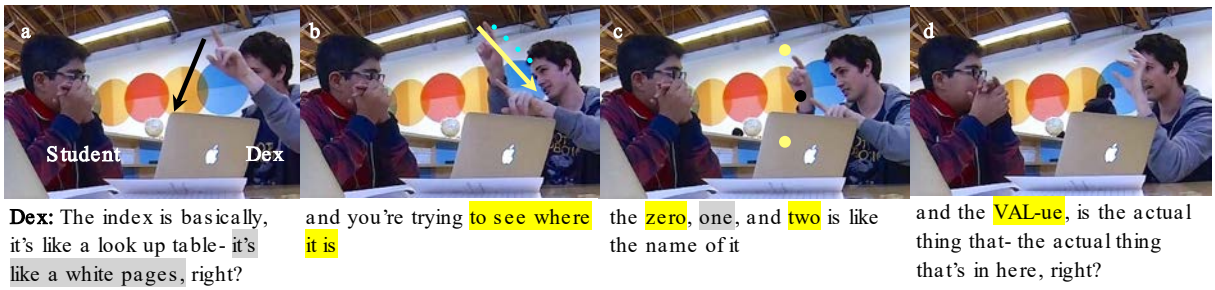


Figure 3. (a) Dex traces his left index finger downwards. (b) Then, he raises his right index finger, squints up at it, and traces it diagonally down towards his left hand. (c) After, he alternately points with both hands to three different vertical locations in space. (d) Next, he makes a precision grip gesture with his left hand. Lastly, he points back to the computer screen (not depicted). *Black arrows show the trajectory of left hand gestures and yellow arrows show the trajectory of right hand gestures. Yellow circles represent the location of pointing with the right hand and black circles represent the location of pointing with the left hand. Yellow highlighted text shows speech that co-occurs with the depicted right hand gestures and grey highlighted text show speech that co-occurs with the depicted left hand gestures. Blue dots show gaze.*

To make the difference between values and indices perceptible, Dex uses his hands to animate what the computer does with each. First, Dex lifts his left hand from the screen and uses his index finger as if tracing down a column, conjuring up the use of an imaginary table (Figure 3a).

Keeping his left hand steady on this imaginary table, Dex raises his right index finger high in the air, squints up at it, and glides it diagonally across the imaginary table, pantomiming a visual search of table contents (Figure 3b). Then, Dex uses both index fingers to locate three different vertical positions on the imaginary table corresponding to the indices "zero, one, and two" (Figure 3c). Lastly, Dex makes a precision grip gesture (Streeck, 2009) with his left hand. This momentarily makes present a new imaginary object—a "value"—in the negative space of his grip (Nemirovsky et al., 2012) and places it in the imaginary table (Figure 3d). Dex then points back to the computer screen (not shown) to link the performance to the code.

By using his hands and gaze, Dex brings to life an analogy to animate how data is accessed in the array. He contrasts the process of *using the indices of a table to coordinate a search for items* with *the actual items* (the values) *themselves*. Through this enactment, Dex provides embodied resources for the student to perceive the different roles of indices and values.

Conclusions

We have shown how programming instructors bring the passive instructions of code to life for students while they examine the static inscriptions of programs together. Instructors animate processes using their hands and bodies to make important, dynamic phenomena perceptible to students. This provides resources for students to evaluate and comprehend computer code in ways that resemble experts'. Our future work will seek to better understand if students take up these resources and how they influence students' comprehension of code over time.

Our findings contribute to a growing body of evidence that STEM instruction, including computer science, is irreducible to written and verbal discourse alone. Multimodal semiotic resources appear to be a pervasive, crucial component in programming instruction. In particular, we add to Kwah and Goldman's studies of pedagogical gesture in programming by contributing a novel function of embodied communicative resources

in teaching students to read code. Our study also extends Nemirovsky and colleagues' findings to a new domain, suggesting that the practice of animating static inscriptions to collaboratively imagine possibilities may be a universal strategy across STEM disciplines. Our ongoing efforts to characterize the range of functions of pedagogical gestures in programming instruction in the wild are important first steps towards more systematically understanding how teachers' use of embodied communicative resources may impact students' learning. We also hope to identify more parallels, as well as distinctions, in the role multimodal communication plays in programming instruction as compared to mathematics and other scientific disciplines.

References

Alibali, M. W., Nathan, M. J., Wolfgram, M. S., Church, R. B., Jacobs, S. a., Johnson Martinez, C., & Knuth, E. J. (2014). How teachers link ideas in mathematics instruction using speech and gesture: A corpus analysis. *Cognition and Instruction*, *32*(1), 65–100.

Fadjo, C. L., Lu, M.-T., & Black, J. (2009). Instructional embodiment and video game programming in an after school program. In *Paper presented at the World Conference on Educational Multimedia Hypermedia and Telecommunications, 2009 Honolulu, HI* (pp. 4041–4046).

Goodwin, C. (1994). Professional Vision. *American Anthropologist*, *96*(3), 606–633.

Goodwin, C. (2007). Environmentally coupled gestures. In S. Duncan, J. Cassell, & E. Levy (Eds.), *Gesture and the dynamic dimension of language: Essays in honor of David McNeill* (pp. 195–212). John Benjamins.

Grover, S., & Pea, R. (2013). Computational thinking in K – 12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43.

Hwang, S., & Roth, W.-M. (2010). The (embodied) performance of physics concepts in lectures. *Research in Science Education*, *41*(4), 461–477.

K–12 Computer Science Framework (K-12 CSF). (2016). Retrieved from http://www.k12cs.org

Kress, G., Jewitt, C., Ogborn, J., & Tsatsarelis, C. (2001). *Multimodal teaching and learning: The rhetorics of the science classroom*. London: Institute of Education, University of London.

Kwah, H. (2013) *Coming to see objects of knowledge: Guiding student conceptualization through teacher embodied instruction in a robotics programming class* (Doctoral dissertation). ProQuest Dissertations.

Kwah, H. & Goldman, R. (2011, April 11). Empathetic embodiments for robot programming. Paper presented at the 2011 annual meeting of the American Educational Research Association. Retrieved May 26, 2016, from the AERA Online Paper Repository.

Lindwall, O., & Lymer, G. (2008). The dark matter of lab work: Illuminating the negotiation of disciplined perception in mechanics. *Journal of the Learning Sciences*, *17*(2), 37–41.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming — Views of students and tutors. *Education and Information Technologies*, *7*(1), 55–66.

Mondada, L. (2012). The conversation analytic approach to data collection. In J. Sidnell & T. Stivers (Eds.), *The handbook of conversation analysis* (pp. 32–56). Boston, MA: Blackwell Publishing Ltd.

Nemirovsky, R., Kelton, M. L., & Rhodehamel, B. (2012). Gesture and imagination On the constitution and uses of phantasms. *Gesture*, *12*(2), 130–165.

Nemirovsky, R. (2014) Animating mathematical symbols. Invited presentation given at the Graduate School of Education Colloquium, University of California, Berkeley. Retrieved May 26, 2016 from https://www.youtube.com/watch?v=P3mot0XA0BE

Nemirovsky, R., & Smith, M. (2013). Diagram-use and the emergence of mathematical objects. In Show me what you know: Exploring student representations across STEM disciplines (pp. 143–162).

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Robins, A., Rountree, J., & Rountree, N. (2010). Learning and teaching programming: A review and discussion. *Computer Science Education*, *2*(13), 137–172.

Singer, M. (2017). The function of gesture in mathematical and scientific discourse in the classroom. In R. Breckinridge Church, M. W. Alibali, & S. D. Kelly (Eds.), *Why gesture? How the hands function in speaking, thinking, and communicating* (pp. 317–329). Philadelphia: John Benjamins.

Stevens, R., & Hall, R. (1998). Disciplined perception: Learning to see in technoscience. In *Talking mathematics in school: Studies of teaching and learning* (pp. 107–149). Cambridge: CUP

Streeck, J. (2008). Depicting by gesture. *Gesture*, *8*(3), 285–301.

Streeck, J. (2009). *Gesturecraft: The manufacture of meaning*. Amsterdam: John Benjamins.

Acknowledgements

This work was supported by NSF AISL #1612660, #1612770, and #1607742.