Ammerman Center for Arts & Technology 16th Biennial Symposium

INTERSECTIONS

Action Coding: Coding as a Calisthenic Practice

Nancy Nowacek

Stevens Institute of Technology
Hoboken, New Jersey, United States

nnowacek@stevens.edu

Abstract

Action Coding is an exploration of computer coding as an embodied performance. This paper presents details of the custom gesture recognition system created, the development process of the project, and the findings after three phases of development.

Keywords

Gesture recognition, performance, motion capture, Kinect, machine learning

Introduction

"Coding is often done in a solitary setting. We sit and think alone, write, revise, possibly submit for code review, revise again, and then merge."[1]

—Emil Ong, a computer coder writing for *Hackernoon*, on Medium.com

"'Coding' is not a musical art, a piano or a violin that a child might need to develop muscle-memory for. It's engineering."[2]

—Attila Vágó, a computer coder writing for *Hackernoon* on Medium.com

These two passages express the underlying reality and culture of computer programming, also known as coding. There is a heroism, a machismo, and an exclusivity implied in these conceptions of coding. The coder, cast in mental battle with his machine, spends hours formulating and typing the perfect chain of commands to build a script that executes without error.

Likewise this image of the coder brings to mind a typical setting: a(n often) male and predominantly white figure hunched over a computer at a table or desk, perfectly still while his fingertips fly across the keyboard, hours passing without notice.

This was not always the case. Women were the first computers and computer operators because of their attention to detail and high threshold for the drudgery of calculations. In performing calculations for rocket trajectories during World War II, women were also the first programmers of the first general purpose electronic computers, because at the time computation and software was not considered 'men's work': circuitry and hardware were. Women also performed the highly physical labor of unplugging and replugging cables of Colussus machines during World War II. However, as digital technologies have become more centralized and prevalent in daily life, so has its culture become more mainstreamed, co-opted and controlled by majority culture: educated white men. Though this reality is slowly changing thanks to Girls Who Code and multiple girls-and-STEM initiatives, the dominant culture of coding is still described in media headlines as 'Bro Culture' shaped and controlled by white upperclass men.

Coding culture holds as a central tenet that coding is hard—it is an exclusive club for the strong-willed. Defined by the tech industry's central focus on efficiencies of speed and process (and thus profit), historic developments in hardware and software have attended to the increasing speed. Successful coders pride themselves on speed, accuracy, and endurance, and text entry via keyboard is the standard for code because of it's speed and reliability. Vágó continues, "What programming requires is analytical thinking, problem-solving attitude, stamina for failed attempts at coming up with the right solution,

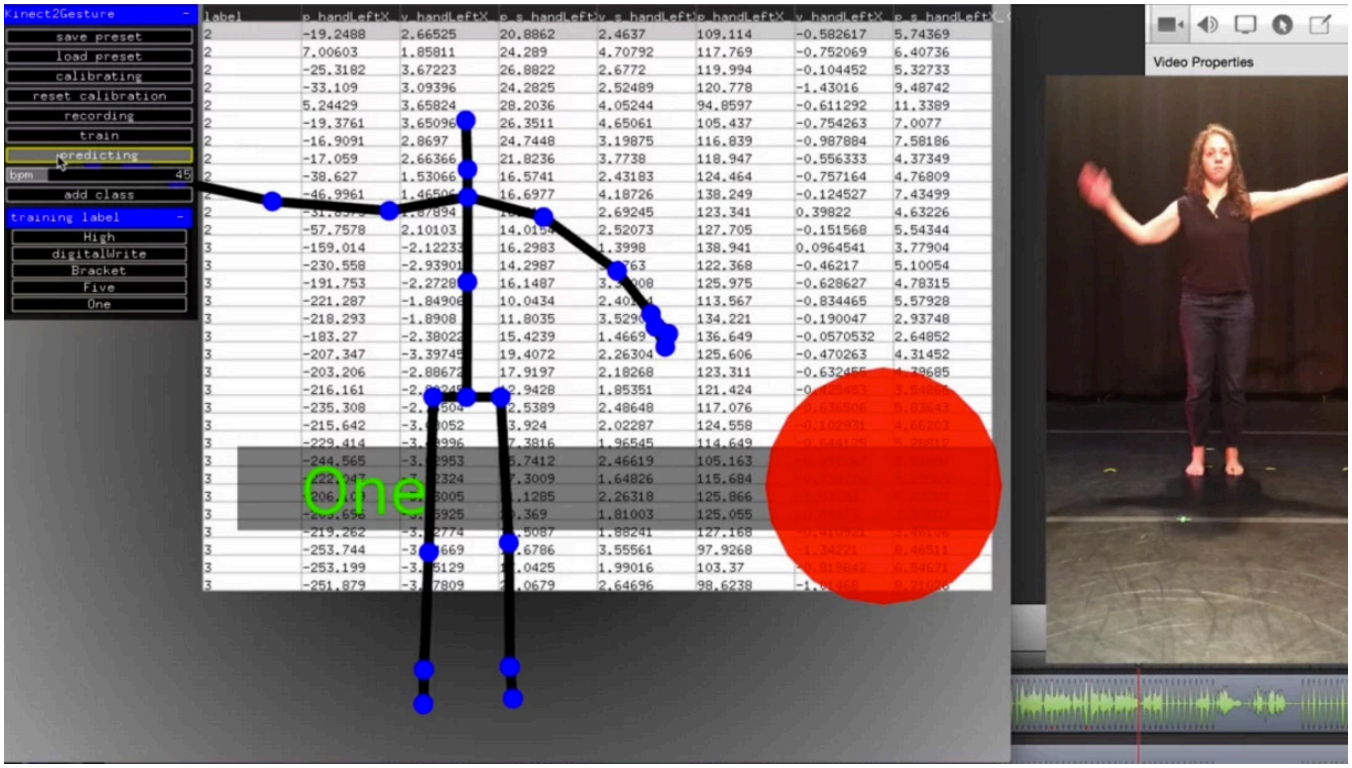
passion for technology, pride in your own code, but maturely accepting someone else's improvements and observations, and a sense of responsibility for any code you write or contribute to. Correct me if I am wrong, but none of these traits are easy to cultivate and develop."[3]

To arrive at that point of cultivation, a first coding language requires multiple forms of literacies, knowledge, and access: the ability to read, a basic understanding of how computers work, consistent access to computer hardware and software, access to a learning environment (a physical classroom, an online course). Coding is traditionally taught in classrooms with a 1:1 student-to-computer ratio. Students are introduced to syntax through introductory scripts. Pedagogy, arising from coding's roots in mathematics relies singularly on interior cognition skills—a silent process of problem-solving in the mind. Learning processes privilege and empower the machine: students, from their earliest moments with code, are fighting against error messages, un-executable scripts and all their attendant failures. Reports of beginning coding classes have described the experience as 'punishing', 'painful', and 'awful'. Reaffirming the image of the lone hero, students must suffer alone, battling against the machine.

Likewise, the environment and conditions within which coding is performed reproduces itself within the products of code. Coding is a seated affair that only employs the fingertips. Software, apps and other digital products are also intended to be used most often while seated, still, and employing only the fingers. This feedback loop reinforces the figure of the lone hero (more often than not male), engaged in solitary activity with his digital device. This condition parallels gender differences in the way men and women work. Men prefer to work alone, women in groups.[4]

Lastly, the ever-expanding reach of digital technologies—under the guise of ease and efficiency—continues the ever decreasing use of the body. Since the industrial revolution, industrial processes, automation, home appliances, and now digital applications and services, have outsourced the work of the body. When compared to these automated and digital options, the body is circumvented because of its comparable lack of speed and efficiency. Outsourcing so much work from the body to other machines and services has an impact. The cultures served by these systems resultantly are in physical crisis. The rise of obesity, depression, and the series of other physical effects of inactivity have been correlated to the increase of prevalence of digital technologies. [5]

These realities construct a space that is highly gendered, exclusive, and alienating, whose long-term effects can also cause physical damage to the body. In response, *Action Coding* is an attempt to create a space for learning code that is physical, cooperative and visi-



Kinect2Gesture tutorial window shows the application interface that includes the coder's tracked skeleton and red time-keeping circle on the left, while Morgan Hille-Refakis performs the gesture on the right.

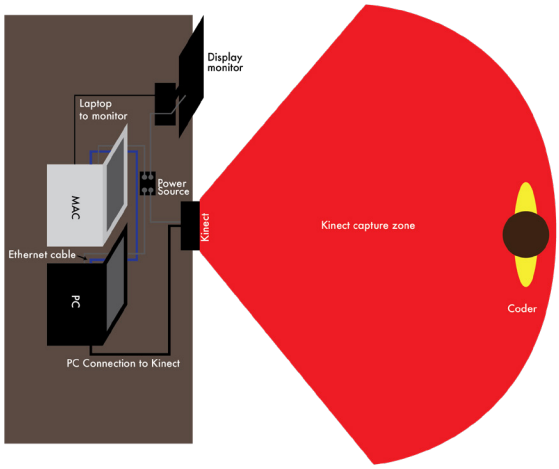
ble. What if code were approached as an externalized, performed activity such as dance or sport? Could code therefore be learned cooperatively, by watching and repeating through the body? If coding is a visible and cooperatively learned, physical experience, what access is afforded? Who gravitates towards this physical process? How does the physical and mental experience differ from the traditional experience of coding? How does the experience and understanding of code change? And more abstractly, what might the products of coding become if performed in this way?

Action Coding: re-connecting the physical and cognitive labor of coding

Action Coding, a series of investigations, is an initial inquiry into alternate methods of performing code with the body—one that, in opposition to Vágó's introductory quote—approaches coding through muscle memory, and asserts a potential relationship between engineering and the body. Over the course of eighteen months, I had the honor of collaborating with talented creative technologists Gene Kogan, David Sheinkopf, and Ramsey Nasser, technologist and dancer Caitlin Sikora, and dancer/choreographer Morgan Hille-Refakis. Together we explored the connections between body and machine, making coding more performative—more visible, tangible, physically strenuous and embodied. The goals of the project are to activate multiple learning senses simultaneously, increase use of the body, and to shift coding to a visible performance that may be learned by watching. [6]

If learning to code is an individual, interiorized process, learning a dance, a sport, or even sign language is cooperative and collective. The cultures of dance, sport, and exercise are based on physical demonstration, repetition, and cooperative dialogue around technique and skill. Learning physical movement activates sight, proprioception, balance, and the full neuromuscular system to see and translate movements through the body. The mind and the body work together to process and express movement patterns.

In *Action Coding*, a coder stands in front of a Kinect. On a large display next to the Kinect, she can see a skeleton version of herself, a big red circle that turns on and off like a metronome, and on another, the window of a coding environment (such as Arduino). Every time the red dot appears (along with a clicking metronome sound), she performs the gesture corresponding to the piece of code she is writing. This could be a command or variable or other necessary syntax. She must complete the entire gesture before the red circle disappears. When it does, the computer has interpreted her gesture and assigned it a class of its library. Simultaneously, the class appears in the coding environment in the adjacent window. Gesture by gesture she builds a script. Depending on the environment, she may have finish her script by performing a final 'Play' or 'Execute' gesture.



Action Coding *system diagram*

Building a system where none exists

Substituting the body for keyboard necessitated creating a new kind of gesture recognition system. Because no pre-made system existed to achieve the goals of the project, one was bricolaged from available technologies and augmented by custom applications. Built with the goal of at-home/consumer access, the Kinect was chosen to capture gestural data, and the rest of the system was built around it. More expensive and advanced motion capture hardware systems were rejected for their lack of access and availability beyond professional or academic settings. To employ the Kinect in this way, Gene Kogan conceived and wrote Kinect2Gesture, a custom machine learning application. In addition to the Kinect (and PC laptop through which to run it) and Kinect2Gesture, the system required a second laptop and ethernet cable (or strong wifi connection) through which to transmit data, and a large monitor on which the coder can see the Kinect2Gesture and her coding environment. The Kinect/PC configuration transmits data via OSC to another laptop (we used a Mac) running Kinect2Gesture, which translates the data into a gesture prediction, and then inputs that data into the chosen coding environment (in the case of this project, Arduino, p5.js, and BodyLang).

Kinect2Gesture is a free and open source application which uses a neural network to automatically classify, in real-time, the physical motions of the full-body coder who is being tracked by the Kinect's infrared camera. When

the coder performs a gesture that has been associated with a particular class or follow-up action the application sends the classification prediction over a network to other computers or applications which act upon the data, for example an Arduino or audiovisual software like MaxMSP, simultaneous to the performance. This has the effect of augmenting the dancer's movements across multiple modes and media.

Kinect2Gesture differs from other full-body gestural systems in that it uses machine learning algorithms in the creation of gesture libraries. Users may devise a series of gestures and train the computer to recognize any single gesture performed within a pre-set time-frame set by the user in Kinect2Gesture to define the start and end parameters of the movement.

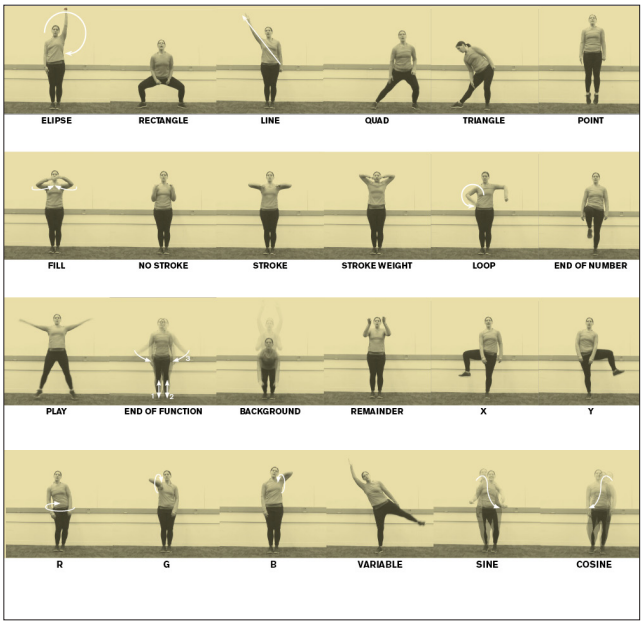
To train the system in a new gesture, the gesture must be performed repeatedly (20-60 times). Each repeated performance generates data. In running the 'Train' function in Kinect2Gesture, the machine parses all 20-60 data sets for each gesture 'learning' the physical definition of each class entered into its library. The wider the variation in subtleties of the movement in style and speed during the training process, the greater the prediction accuracy. Not only can users define their own gesture libraries, they can also apply those libraries to a variety of coding environments. As an application, Kinect2Gesture is not constrained to any particular development environment, nor is anyone who might want to engage with Kinect2Gesture in this manner be constrained to a limited library of pre-made gestures.

Once the system has been trained on the series of gestures corresponding to the necessary commands and inputs of a coding syntax, 'Prediction' mode is selected. In Prediction mode, Kinect2Gesture compares Kinect data to the spectrum of data sets it has learned for each gesture and predicts the gesture that has been performed. In order to capture a usable data set, the application uses a visual metronome in the form of a large red dot that signals to the coder when the system is 'watching' and thus the window within which she needs to perform the gesture in full. The prediction is output on screen immediately, and that data is input into the selected coding environment.

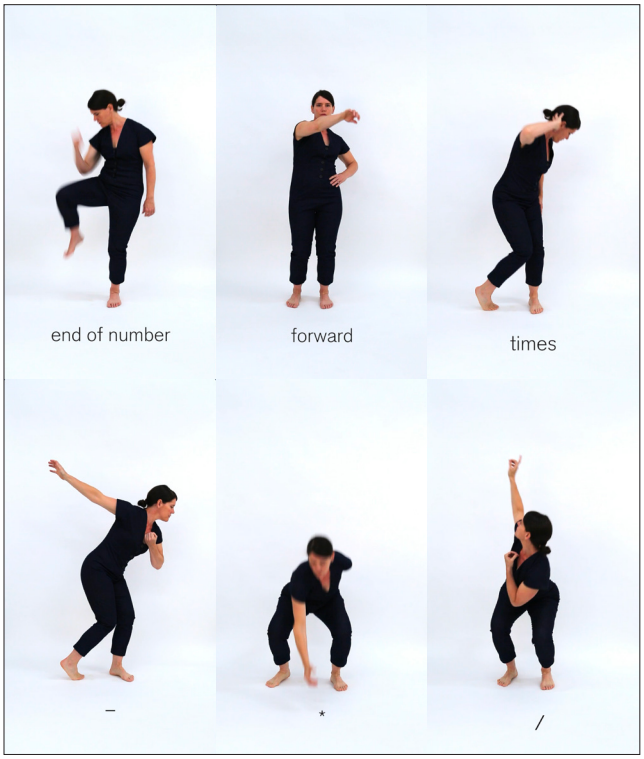
Learning by doing: an iterative process in three phases

Action Coding consisted of three distinct research phases. The first phase—in collaboration with Gene Kogan, David Sheinkopf and Morgan Hille-Refakis—tested the initial intention of the project by building a small gesture library that could be used to write Arduino scripts to turn LED lights on and off. Additional considerations of this phase focused on the body. To shift the labor of writing of code from the fingertips and keyboard to the full body, a series of parameters focused on the health and wellness of the body guided gesture creation and performance. To avoid fatigue and/or repetitive stress of a particular joint or limb, gestures should engage the full body, from head to toe; and gestures should be balanced across the body in lateral, frontal and sagittal planes. Secondarily, was a goal to create gestures that were meaningful, memorable and pleasurable guided this phase. 'Meaningful' gestures concerned connect-

Morgan Hille-Refakis performs components of the first Arduino gesture 'dictionary'.



Segment of p5.js dictionary, performed by Nancy Nowacek.



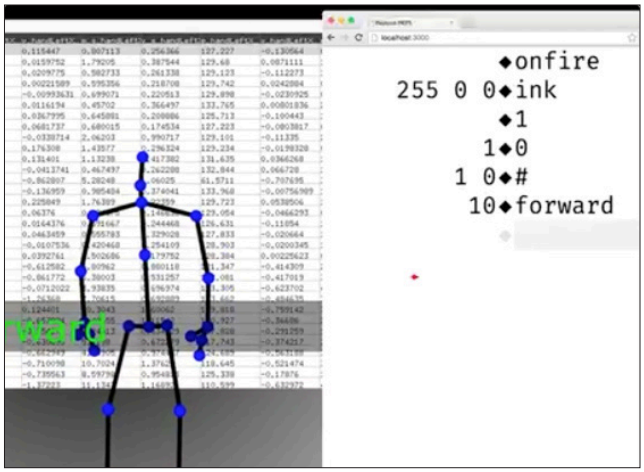
Segment of BodyLang dictionary, performed by Nancy Nowacek.

ing the semiotics of the movement with code semantics. 'Memorable' gestures combined meaning with variety and unique body positions. 'Pleasurable' gestures were those identified as fun to perform and repeat.

We began by working with a series of three scripts: to turn a light on, to make a light blink rapidly, and to slow the blinking down and turn the light off, and worked with two other dancers, Carlo Antonio Villaneuva, and Morgan Preston, neither of whom had prior coding experience. Each script was deconstructed into necessary functions and inputs, for which a gestural 'dictionary' was created. This dictionary included gestures for the full set of commands and syntax necessary to write each script such as 'VoidSetup' and syntax such as the semicolon. We realized that in this first test phase, the code performer was simply 'performing typing' with the whole body, so the goal of the second phase was to create gestures for full functions.

This first phase of the project was realized in three public presentations, two in dance performance contexts—the Your Move festival, and Movement Research Spring program at Judson Church in New York City.

Motivated by the observed potential of the gestural input system to correlate with graphic focus of p5.js, technologist and dancer Caitlin Sikora and I, with Morgan Hille-Refakis, developed a second library of gestures. Building on experiences from the Arduino 'dictionary',



Monitor display for BodyLang: minimized Kinect2Gesture window on the left, BodyLang code in progress on the right.

our goal was to seek a more direct relationship between gesture, meaning, and function. Two sample scripts—one to draw a circle and the other to draw a series of vertical lines to create a square—were our basis for dictionary creation. Based on learning from the first phase, this phase increased the envelope of movement and number of entries in the dictionary. Single gestures in this library were created to communicate functions such as 'stroke weight' and 'end of function' in addition to 'ellipse' 'rectangle' 'line' and numbers 0-9 and involved different forms of jumps, squats, and directional arm movements, amongst others. This phase moved further away from performing typing, though still directly related to the syntax of p5.

For example, the command to draw a circle was performed by moving the right arm in a large circle. The command to draw a rectangle was represented by a lower body shape, with knees out to each side at 90° angles over ankles, that resembled a rectangle. We built a larger library of gestures with the goal of clustering p5 syntax in meaningful way, such that a single gesture could correspond to strings of syntax. This phase was presented publicly at The School for Poetic Computation's spring exhibition as a video work.

In the third phase, Ramsey Nasser wrote a custom language, BodyLang, to create the most direct connection between gesture and code. Based on Logo, BodyLang is stack language for drawing. Of all three iterations of the project, Nasser's stack language allows the most direct connection between gesture and code. As new lines are added to the stack, the code executes in real time without need to compile, upload, or play. This supplies instant feedback to the coder, and supports improvisation and play with the system more so than the other environments. The goal of this phase was to 'perform code' in its most elegant sense, moving as far from

the 'performing typing' paradigm as possible. Additionally, the goal of the gestural language of it's dictionary was an experiment shifted the sense 'meaningful' in the previous sense, to 'being recognizable' and intensifying 'memorable' and 'pleasurable' by using vernacular gestures from sports and hip hop cultures. Gestures were inspired by basketball, football, and music videos. The "free-throw", the "end-zone slam," and the "stank leg" represented commands such as pen-up, end of number, and repeated cycles. This phase of project development was presented publicly alongside the first two phases in a solo art exhibition at Eyebeam ("Easy Is Not A Concept"), a public workshop, and a college class workshop at Scripps College—a women's college.

Limits and potentials

Findings can be separated into four categories: the limitations of the Kinect, the limitations of the system, the limitations of the coder, and the potentials of the system.

Much time was spent exploring the spectrum of movement within the Kinect's capabilities. Crossing limbs, spins, or shifts in head and hip position were not readily detectable by the Kinect. Therefore, gestures were constrained to the frontal plane and focused on the shapes produced by the body, with focus on the arms and legs. Often gestures needed revising because they were too subtle to be detected. The resulting learning is a correlation between Kinect vision and cheerleading: the bigger the shape created by the body, and the more crisp its execution, the more clearly it could be consistently perceived.

The Kinect's data capture impacted the rest of the system. Though all gestures were devised to be as visually unique as possible to enable successful prediction—especially with regard to gestures that may be used consecutively such as numbers—the data captured by the Kinect and used by Kinect2Gesture often resulted in prediction errors. Each code library contained no more than 30 entries, which was considered to be a very small and symbolic amount by the team. However, it became clear that the maximum number of discreet detectable gestures, within this system, could be no more than 5-10.

Furthermore, the system's complexity also limited its success. Each component of the system—Kinect, PC, Mac, wireless communication and other peripherals—introduced potential failure points and factors that could be accounted for but not easily addressed.

When the system was performing at it's best and fully functioning, the results satisfied the original intent of the project. Participants reported that they felt like they were playing a game: trying to perform the correct gesture in the correct sequence was fun, exciting, and engaging. Participating dancers commented that performing coding gestures in collaboration with the system also felt like 'performing good technique', and that seeking the best expression of each gesture was a stimulating challenge much like performing a proper grand plié.

Participating dancers who had no prior experience with computer coding, reported that they were able to connect to the code through movement. Memorizing the choreography of a particular script, in effect taught each dancer the basics of the code. Embodied experience of a script built muscle memory but also by repeating movement patterns, dancers gained a sense of what should 'naturally' come next. Once dancers learned scripts, they were able to manipulate and iterate the code because they also had built an operational sense of the code through their bodies.

Project participants given the choice of which dictionary to learn most often gravitated to the BodyLang dictionary. Participants seem to gravitate towards these movements because they seem more 'familiar' and 'fun' and 'dance-y' than those used in P5 and Arduino which seem more 'basic' and 'like exercise'.

No matter the physical training or coordination in participant, limitations of the body were observed. Participants shared feelings of panic and confusion when first attempting to perform gestures in time with the Kinect-2Gesture's rate of capture. Though that rate can be set by the user to any beats per minute, the factor of time for new participants create a pressure that short-circuited the mind-body connection. Because the body and mind are so rarely employed in simultaneous labor such as this, a new coder can be easily overwhelmed by the dual process of recalling a gesture and performing it to a set beat per minute. However, when scripts are learned and memorized 'offline', they can more easily—and pleasurably—be performed in collaboration with the system. From this we learned the need to introduce participants to the gestural languages offline, followed by memorizing a beginning sample script completely before engaging the system online. Secondly, we learned that once a basic script has been memorized, it can be more easily expanded upon or manipulated by the coder. The 'processing load' when working simultaneously in the mind and body impacted advanced users as well. For more advanced users capable of improvising and coding on the fly, there was a repeated and consistent need to pause the system in order to compile the next string of inputs in the brain before performing them in time to the system's beat. Gestures for 'pause' and 'resume' were added to the BodyLang library.

Lastly, a consistent observation throughout the project was general body fatigue after a couple hours' work with the system. Likewise, a brain fatigue was also observed in conjunction with the tired body, where the ability to recall or correctly perform a specific gesture began to decrease over long stretches of time. This fatigue was not perceived as negative, however. Often the phrase 'a

good tired' was employed to describe the fatigue, accompanying reports of an overall sense of satisfaction from have the feeling of [bodily] accomplishment.

Conclusions and future work

In the context of *Action Coding*, a visible series of full-body actions aids in the formation of the building blocks of coding: the mind learns through the body. Syntax becomes a swing of an arm, a jump, or a squat; and logic becomes repeated movement patterns and pairs, much like dance.

If coding becomes visible, tangible, embodied, and cooperative, who feels invited to code? The participants most drawn to the project in public presentations were female. Many interested participants approached the project because they studied dance at one point, or were still practicing dance. Others gravitated towards the project because of the unique quality of the movements and interest in the visible interaction with the Kinect.

One of *Action Coding*'s most important achievements is a system in which anyone can create a series of full-body gestures and train the system on them. This capability offers a wide variety of users with a spectrum of mobility capacities to participate in motion capture and gesture recognition technologies. For example, a user who sits in a wheelchair or who cannot stand for long periods of time can create a library using only the upper body.

If coding is a physical process, how is learning code impacted? The introduction to coding through an embodied process was shown to be very powerful for those new to coding. Working offline with beginning coders through physical movements, and then applying those movements to a gesture recognition system such as *Action Coding* could be an exciting and effective method for teaching beginning code. The procedural memory required by the physical process amplifies the procedural memory required by computer coding; and the motor programs acquired by this process underscore the computational programs of code. In this way, the project reimagines coding as a function, in part, of motor learning; a new coder may learn and internalize syntax and logic patterns more quickly and because they are taken in through the full neuromuscular system, retain them longer [7,8,9]. As artistic research it suggests a pedagogy of experimentation, fun and one driven by iteration and play, not failure.

And more abstractly, what might the products of coding become if performed in this way? No conclusions towards specific products have surfaced, however, the positive effects of working with code in a physical, cooperative, embodied manner challenge dominant capital-driven definitions of efficiency. Coding through this system is admittedly not efficient, it is inefficient. However, the affective outcomes of the project are robust: participants who normally spend the bulk of their work and recreational time on computers reported feeling tired but good. To reiterate, they felt a feeling of accomplishment and achievement. These outcomes suggest an economic ecology that includes wellness in its profit motivations and organizational goals.

Future work in this domain will continue, and seek larger groups of participants from multiple ages, racial and ethnic backgrounds, and abilities. Future iterations will start by seeking to revisit the gesture recognition system components in hopes of refining the motion capture capabilities and reducing the complexity and opportunity for error. Investigation will continue around gesture, semiotics, and syntax.

The objective of *Action Coding*—as a research-based art project—was not to revolutionize digital industries, but it suggests modes of cooperation and human-machine interaction that could sought to address it's historic culture and diversity problems by inviting new groups of coders to learn and use code in untold of new ways, and to create new digital products that engage full use of the full body.

References

1. Emil Ong, "Practicing coding as a performance," *Hacker Noon (a Medium publication)*, May 24, 2017, https://hackernoon.com/practicing-coding-as-a-performance-c0ca0ec7261c.

2 Attila Vágó, "Coding Has Become Pop Culture," *Hacker Noon (a Medium publication)*, Jan 18, 2017, https://hackernoon.com/coding-has-become-a-pop-culture-939100f84b0c.

3. Ibid.

4. Derek Thompson, "Why Women Prefer Working Together (and Why Men Prefer Working Alone)," *The Atlantic*, August 21, 2013, https://www.theatlantic.com/business/archive/2013/08/why-women-prefer-working-together-and-why-men-prefer-working-alone/278888/.

5. Anusuya Chatterjee and Ross C. DeVol. *Waistlines of the World: The Effect of Information and Communications Technology on Obesity*. Santa Monica: Milken Institute. August 2012. http://www.milkeninstitute.org/publications/view/531

6. Diane Solway, "How the body (and mind) learns a dance." *New York Times*, May 28, 2007, http://www.nytimes.com/2007/05/28/arts/28iht-dance.html.

7. Danielle S Bassett, Muzhi Yang, Nicholas F Wymbs & Scott T Grafton. "Learning-induced autonomy of sensorimotor systems," *Nature Neuroscience*, 18, pages 744–751 (2015), http://www.nature.com/neuro/journal/v18/n5/full/nn.3993.html

8. Colin Barras, "What to learn quicker? Use your body," *BBC.com*, March 21, 2014. http://www.bbc.com/future/story/20140321-how-to-learn-fast-use-your-body

9. Lee, D.T., & Schmidt, A.R.. “Motor Control and Learning: A Behavioural Emphasis.” (4th ed). Windsor, ON: *Human Kinetics*, 2005.

Author Biography

Nancy Nowacek is a Brooklyn-based artist, designer, and educator. Her interests are based in urban design, technology, body literacy, and communication.

Nowacek is currently a research resident with Cape Cod National Seashore and Freshkills Park (Freshkills R&D). She was previously a fellow at Eyebeam, and has previously been supported by residencies through the Lower Manhattan Cultural Council, Recess, Signal Fire and the Sharpe Walentas Studio Program. She teaches art and design at the Stevens Institute of Technology, and organizes exhibitions, panels and events devoted to waterways and climate change as well as bodies and technology. She has presented works in New York, Los Angeles and the Bay Area, Canada, South America and Europe.