This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

8 Computational Heterogeneity: A Radical Reflection

8.1 Computational Heterogeneity: A View beyond Binaries

But computation—and technology, more broadly—affords a much wider range of experience and possibility than suggested by the box on one's desk.
—Mike Eisenberg[1]

In *Experience and Education*, Dewey argued that humankind is "given to formulating its beliefs in terms of Either-Ors, between which it recognizes no intermediate possibilities." Not surprisingly, educational computing is no exception. One such Either-Or binary that is relevant to our project so far is *computing vs. computational thinking*.[2] A keynote address delivered recently at a computing education conference by the computer scientist and educator Judy Robertson is quite illuminating in this respect.[3] Robertson argued against a blind emphasis on computational "thinking" as the centerpiece of computational literacy. Focusing only on creating SCRATCH programs may lead especially young learners away from understanding how the computer actually works as a "machine." Children's drawings of what is inside a computer show that learning to use computational abstractions that we typically introduce in the K–12 levels—such as variables and loops—does not necessarily help us understand how these abstractions are processed and represented inside the computer. What gets lost in our push to teach children how to code is the *physicality* of computing as a machine, which involves a deeper understanding of the relationship between the hardware and software—and many other forms of computational abstractions that are not even presented to children as part of their experience of coding or computing.

As poignant as Robertson's argument is, we need to be careful before creating and popularizing such binary oppositions in educational computing. Thinking computationally is not merely a solipsistic event, and neither should it imply the reduction of computing and computers to programming and software. Similarly, focusing on computational participation[4] should not negate what

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

can be gained from paying attention to intuitions. The disconnectedness implied in such binaries—hardware versus software, individual versus collective, thinking versus doing, and so on—continues to reify device-level engagement. Within any of these binaries, a shift in focus from one to another simply implies a different form of device-level engagement, for example, a shift from programming to hardware, or a shift from an individual's code to remixed and shared code.

A contrasting image involves fluidity and movement both within and across these binaries. To this end, we offer *computational heterogeneity* as a heteroglossic imagination of code and coding. Committed to accounting for the manifold forms in which code is experienced, it offers an ontological and epistemological reimagination of the boundaries of code. It reminds us of the inseparability of code and its inevitable *other*: the world in which it becomes meaningful in experience, and the voices that are often left out of the folds of computing and STEM.

Computational heterogeneity thus explicitly seeks to counter the scepter of technocentrism, the fallacy of seeking all answers to questions about technology within the technology itself. As we discussed in chapter 1 (see section 1.1), the dangers of technocentrism are two-fold. In the technocentric paradigm, the lack of cultural and institutional supports for computing in educational settings are often treated as a failure of the technology itself,[5] and furthermore, ignoring such factors can lead to positioning historically marginalized students as deficient in computing.[6] In countering technocentrism, computational heterogeneity also involves challenging disciplinary masculinities and hegemonies that have long defined computing and technoscientific cultures.

This implies a shift away from computational artifacts to computational utterances, our central unit of analysis. The forces of centralization and decentralization—that is, the search for coherence as well as the heteroglossic nature that is fundamental to language[7]—both shape computational utterances. It is through the dynamic interplay between these two forces that the experience of coding in STEM shapes the boundaries between code and its *other*—the *social horizon*[8] toward which every utterance is oriented. The separation between what is inside the computer and what is outside becomes malleable, as computer code becomes enmeshed with and refracts through the embodied, material, historical, affective, and social dimensions of experience. In the rest of this chapter, we reflectively examine the epistemological, methodological, and pragmatic significance of the various images of computational heterogeneity that we have presented so far.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

8.2 Advancing Critical Phenomenology

As phenomenologists, we positioned our approach in terms of Merleau-Ponty's notion of the *phenomenal field*, centering our focus on the *sense experiences* of students and teachers. The phenomenological agenda also relies on radical reflection, a form of reflection that must necessarily go beyond the veneer of the sphere of givenness.[9] This means searching for sense experiences of the participants in our studies using lenses that go beyond the scope of "computational abstractions" and "computational thinking" and illustrating how these experiences extend way beyond device-level engagements. They are distributed in myriad forms beyond the computer and the programming language and are also constituted through the interactions between them.

As *critical* phenomenologists, we have presented several cases that highlight the importance of *de-centering*[10] the scholarly conversations around code and coding in K–12 STEM education. Experience, in this perspective, is not universal; rather some ways of feeling and perceiving are privileged whereas others are silenced or excluded.[11] Critical phenomenology should always remind us of the threats of masculinity in technology[12] and the historical marginalization and exclusion of gender minorities and racialized voices from scientific and technological spheres.[13]

This implies that we must learn to see privilege and marginalization as inseparable elements of the social horizon toward which a computational utterance is oriented. The metaphors of voicing and heterogeneity must therefore also confront these issues as *central* to framing code and coding, both in general and in K–12 STEM. Specifically, our work outlines several ways to do so: by centering the work of coding in STEM in students' lifeworlds, by centering voices that have been historically left out from the fields of science and technology, by reconceptualizing disciplinary boundaries and using computational simulations to model and discuss social inequalities, and by explicitly recognizing emotion and carework as central to computing in the science or STEM classroom.

Of particular note is the importance of listening to and including teachers' voices in our accounts of coding in K–12 classrooms. This is also an essential element of our critical phenomenological agenda, because teachers' voices remain largely marginalized and are usually excluded in educational computing research. The general approach in educational computing research has been to impose researchers' designs on teachers, and even remove teachers from their classrooms and instead have researchers conduct "intervention" studies. We believe that in order to step beyond technocentrism and technological determinism in the classroom, we must engage with teachers as partners, who in

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

our experience are often women with no prior experience in coding. Chapters 4 and 6 present images of teachers voicing code as an integral part of their mathematics and science classrooms. As we have argued elsewhere, the conceptual dissonance between researchers and teachers that arises in such contexts is central to the heterogeneity that is essential in viewing coding in classrooms as dialogical.[14] The meaning of words and practices that appear disciplinarily profound to reseachers must be negotiated with teachers' perspectives. As researchers, we must remember that the discipline, in a deeply Bakhtinian sense, is only half ours, as it is populated with intentions of others; so, we must learn to see teachers as our essential and inseparable others in reimagining computational science in the K–12 classroom.

Critical phenomenology is thus at once epistemological, methodological, and axiologial. Epistemologically, it is an attempt to reorient our understanding of the experience of computing and coding toward critical, historical, and aesthetic positionings of the participants, away from a view in which coding is tied to device-level engagement. Methodologically, it emphasizes *listening to the heterogeneity in participants' voices* rather than an overt focus on prescriptions of what they should be doing with the computer. Axiologically, there is a distinct commitment against individualism and technocentrism that has largely defined professional cultures of computing and computing education, as illustrated in the work of critical technology scholars such as Noble, Ames, and Irani (section 1.3.2).

A critical phenomenology of code should serve as a caution for us not to rush into designing assessments of coding without paying attention to (1) the complexity and heterogeneity of experiences of coding beyond device-level engagements, and (2) "who" is(are) voicing code, "with" whom, and "for" whom. In urging us to shift from computational artifacts to computational utterances, our goal is to remind ourselves of the difference between the reified text and the heterogeneity inherent in language. This implies a shift from focusing on the computational object as the locus of experience to the ever-changing boundaries between the computer and the more-than-human world as a more complex but authentic site in which coding unfolds.

This is an image of *différance*[15] in which the experience of coding is different and deferred from the symbolic form of code that is given to the learner. It also stands in contrast to a form of authoritarian discourse in which students' computational experiences are primarily evaluated in terms of their computational productions guided by narrow definitions of computational thinking. Computational heterogeneity is an account of how meanings of code and coding emerge through heteroglossic refraction through the alterity and addressivity,

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

perspectivity, and transparency inherent in language, both natural and artificial. In the following section, we present six forms of computational heterogeneity that emerged in our empirical observations and analyses.

8.3 Forms of Computational Heterogeneity

8.3.1 Perspectival Heterogeneity in Coding Science

The first example of a radical reflection on coding is presented in chapter 3, which calls into question a near-axiomatic assumption in the scholarship on agent-based educational computing: taking on the agent perspective is intuitive even for young learners.[16] Chapter 3 suggests that although this may still be generally true, this assumption or claim of intuitiveness of the agent perspective can be seen as an approximation of a much more complex experience when we "zoom in" on the very early moments of how new learners struggle with interpreting agent-based code.

This complexity is revealed in the form of Arnav and Liam's struggles with figuring out what the computational agent stands for (or, should stand for) in their activity. *Thinking like the agent*—which has been shown to be highly effective for helping us understand complex mathematical and scientific phenomena[17]—is by no means itself an easy feat. Multiple perspectival frames or points of view are at work when coding serves as a language for modeling a scientific phenomenon. The heterogeneity arises from perspectives inherent in spoken (natural) language, visuospatial perspectives that are often tied to particular representational formats (e.g., graphs of motion or spatial representations of motion),[18] and then, of course, the perspectives that are prompted by the interpretations of the programming commands. Coding science is therefore a complex language game of a dynamic interaction between perspectives. Add to this the reality of the science classroom, and it becomes even more complex as learners begin to work with others, and teachers intervene. Although "seeing" the world through the perspective of an agent is certainly an important pedagogical goal, paying attention to the "voices" reveals how the agent perspective becomes meaningful only through negotiation with others, even during the early stages of modeling.

What, then, are some implications of perspectival heterogeneity for educational computing researchers? At the very outset, we must be mindful that students' difficulties in understanding and interpreting code, especially in the context of STEM classrooms, may go far deeper than understanding the syntax and semantics of code. It may be that students' challenges arise from the lack of perspectival coherence among the heterogeneous points of view inherent in the different elements of the activity: the phenomena to be modeled,

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

the programming commands to be used, the perspectival complexity of spoken language, the differences in perspectival frames used by different students, and the genre of programming itself (e.g., agent-based programming may present different demands on perspectival framing from other forms of programming such as logic programming). One of the roles that the teacher can play in such settings, as we have illustrated, is to listen carefully for perspectival incoherences and introduce prompts for perspectival shifts accordingly. And finally, we have also pointed out how the modeling platform and activities can be designed in order to support learners to take on multiple and complementary points of view, for example, through supporting both event-based, spatialized representations and graph-based, cumulative representations of motion.

8.3.2 Experiencing Abstractions as Recontextualization

The experience of abstractions, as we remarked in chapter 1, is that of recontextualization. Each of our empirical chapters bears evidence to this effect. The perspectival heterogeneity Arnav and Liam experienced (chapter 3) in order to interpret what a computational agent stands for is an example of recontextualization, in which the complexity of linguistic context shapes the interpretation of code. Our analysis of how new advances in computational modeling of ethnocentrism have emerged as computer scientists have recontextualized and adapted the same algorithm in different phenomenal contexts is an example of another form of recontextualization (chapter 5). The involvement of users within the design process is also an experience of recontextualization of both disciplinary (e.g., mathematical) and computational abstractions, as evident in the work of 4th-grade students described in chapter 4. Teachers, with no prior experience in coding, recontextualize coding as mathematizing in the elementary science classroom, and in the process, they integrate coding, mathematical modeling, and science (chapter 6). In chapter 7, we showed that students in the margins of computing—historically in terms of both their race and their interests—can find their computational voices as their coding brings together different worlds. The virtual world within the computer merges with their lifeworlds outside the classroom. In such contexts, coding becomes carework, the kind that has historically been silenced in the overtly masculine culture of technology design.[19]

It is important to note that this is not a simplistic call to pay attention to the context in which coding is experienced. Very few of our readers would disagree with the general observation that context is central to learning, teaching, and cognition. So, what does our specific observation add here? The most important implication is that the phenomenological frames that we have provided in each chapter—perspectival thinking, transitional othering, designing

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

usable software *for* and *with* others, coding as aesthetic experience, and coding as mathematizing—alert us to specific, heterogeneous *forms* in which coding is experienced in STEM classrooms. This means that our experience of computational abstractions involves the interplay between multiple contexts, and each of the phenomenological frames illustrates a particular form of this interplay. It is this interplay that we term recontextualization. Furthermore, in each form of recontextualization, the relevant experience significantly expands beyond the scope of virtuality on the computer, which in turn necessitates paying careful attention to discourse around computational abstractions, not merely the computational artifacts that embody these abstractions.

8.3.3 Alterity and Addressivity in Computational Design

The notion of disciplinary authenticity runs deep in the scholarship of public education because it determines to a large extent what we want students to know and be able to do, and how we assess learning.[20] What counts as authentic, in turn, is based on our understanding of what disciplinary expertise looks like.[21] What does this mean in terms of educational computing, particularly in the context of coding and modeling in K–12 STEM? One obvious answer is to pay attention to the representational practices that are central to scientific modeling and are also reflexive with programming and computational modeling.[22] Design and perspectival work, as we explained in chapter 2, are examples of two such anchors that can productively pivot us between these worlds.

The emphasis on voicing and heterogeneity brings to light what we believe to be a deeper form of experience underlying authenticity: alterity, or otherness. To remind the reader, alterity is central in Bakhtin's problematizing of the relationship between the "outside" and "inside" of a text, between its origin, context, or referent, and its form or structure.[23] Alterity demands that we pay attention to not only "where" the words are or have come from, but also "whose" words they are. Populated with the intentions of others, according to Bakhtin, language itself occupies the boundary between self and other. A word is thus half ours and half someone else's, and as we argued in chapter 1, what is true of the word must also be true of code.

In chapter 4, we have presented a pedagogical approach in which the "inside" and "outside" of code were reimagined through connecting code with physical models, as well as through actively engaging an authentic audience (mathematics teachers). The experience of the *other*—the authentic audience—took on a central role in the students' design journey. Mathematics teachers unaffiliated with the study served as prospective users of the mathematical machines designed by student dyads and were involved as an integral part of the design experience. What counts as mathematical explanations of code became rede-

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

fined for the students through iterative engagement with the users, which also deepened their engagement with the code.

In Bakhtinian parlance, this is also an experience of addressivity, the constant state of being addressed and being in the process of answering. Paying attention to addressivity implies a shift away from computational artifacts to computational utterances as the focus of computational design. Enlivening addressivity involved moving beyond the technocentric frame in the classroom as the student-designers iteratively interacted with authentic users in two successive user testings. Their teacher, Ms. Lena, also carefully designed classroom instruction through which she helped students recognize the users' feedback as valuable for improving their designs. It was through such experiences that student-designers learned to see their own work as designs *for others*. Through this work, we have also critiqued the individualistic emphasis in computing education. Notions of agency, ownership, and control are repositioned dialogically, as students' computational designs embody iterative improvements that arose from dialogical engagements with each other and the users.

8.3.4 Mathematizing: Computational Heterogeneity and Teacher Voice

In chapter 6, we followed an elementary teacher's (Emma's) class over a span of two years, as she used programming with ViMAP as part of her science and math classroom instruction on a weekly basis. Emma framed coding as "mathematizing," that is, she found a place for coding in her science classroom as a way to mathematically model physical and biological phenomena that students were learning. Within this broader frame of mathematizing, the most persistent finding was the constant movement of the modeling activities across representational systems: embodied modeling, physical modeling, modeling using programming commands in ViMAP, and modeling using mathematical inscriptions (e.g., multiplication tables, hand-drawn geometric shapes, and so forth). Coding took on the form of a phenomenon that circulated across these representations, which involved transforming one form of representation to another.

Integrating coding with K–12 science is not merely a problem of designing programming languages that are aligned with disciplinary expectations; we must also keep in mind the inherent diversity of the representational infrastructure within the classroom that teachers typically work within, alongside the programming language. It is in working with the non-computational elements of this infrastructure that teachers may be able to find their computational voices, as they engage in the work of designing modeling activities that involve both programming on the computer and modeling outside the com-

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

puter. The ruptures that appear at the interfaces between heterogeneous representational forms also become sites for productive inquiry.

By leveraging and making visible the heterogeneity of the representational infrastructure, Emma created opportunities for children by valuing, rather than devaluing, their interpretive dilemmas and moves, and by acknowledging the uncertainties involved in managing the *mangle*[24] of materiality, theorization, and coding in the context of modeling science.

8.3.5 Relational Work: A Critical Aesthetics of Coding

The roots of Logo in feminist epistemology have been well documented in early writings on constructionism. These writings reflect how learning with a computational agent (e.g., a Logo Turtle) is more akin to getting to know a person, rather than a mastery of symbolic forms, and thus highlights a relational epistemology that stands in contrast to technological determinism (see section 1.3.2). But is simply using Logo or agent-based computing (more broadly) enough to break the masculinist hegemony in technological worlds?

Ames' groundbreaking observations of the context of how the OLPC laptops and software were *actually* used by the intended users—children in the Global South (for example, Paraguay)—is a reminder of the limits of such technocentric imaginations, in which the essence of learning is ascribed to the "charisma" of technology. She illustrates the richness of childrens' lives away from the computer and contrasts that with the limited experiences on the computer.[25] In a similar sense, chapter 7 highlights how important the performances and relationships outside and beyond the computer and the classroom are for learners who are at the margins of computing for finding themselves within computing.

Papert's notion of the Turtle as a transitional object is a certainly powerful proposal. The ViMAP Turtle "stood in" for a religious symbol for Shenice, and for Ariana's best friend: Thomas. But caring for Thomas also brought Ariana together with Matt, another student in the course, and created opportunities for humorous banter between them. Ascribing the richness of Ariana and Matt's relationship to the power of the transitional object, however, would be essentially technocentric. The richness of the experiences that unfolded outside the computer brings into account the lifeworld of the learners, including who (not only what) they care about, and even their religious lives, as was evident in Shenice's case.

So, to answer the question we asked in the first paragraph: no, simply using Logo-like programming environments is not the answer. But designing activity systems in which children, especially those who have been historically marginalized due to their race and gender, can meaningfully integrate stories of their loved ones and their objects of affection with disciplinary work, can offer

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

a critical and aesthetic[26] alternative to technocentrism. The focus here is on a deep and inextricable relationship between *history* and *form*, which Bakhtin argued is inextricable in language.[27] The historical dimension of the learners' experiences becomes deeply intertwined with their computational work as they engage in a particular *form* of experience: relational work.

A critical aesthetics of coding thus orients our attention to relationality in learners' experiences beyond the computing device(s), not merely its structural elements. This also includes the personal, social, and historical dimensions of their experiences, as evident in our account of Shenice's work in chapter 7. Caring for the other—where the other stands in for the computational agent as well as friends and loved ones—is an example of such a form of experience that at once is historically grounded in the lives of the learners and also makes explicit a relational epistemology of computing. But, our point here is that this critical, aesthetic manifestation of computing as language becomes visible when we, as researchers and educators, learn to see the happenings beyond the device as the spaces where computing also unfolds. While Ariana and Matt's banter offers a fortuitous example of such a relational unfolding, our inability to recognize and honor Shenice and her friends' invitation to their church stands as an example of our device-centeredness. The charisma of the computer can—and typically does—blind us to such relational unfoldings. Given that relational work is actively discouraged and "disappeared" from the professional worlds of technology design,[28] it becomes even more important for us to value such *forms* of experience in the computing classroom.

8.3.6 Voicing Code as Transitional Othering and Recontextualization

How can computational modeling be used to support critical and difficult conversations about race and inequality in the classroom? In chapter 5, we take on this issue and argue that *transitional othering* can offer us a potential answer. The transitional nature of computational agents can provide students and teachers with a space in which they can explore complex critical sociopolitical issues such as race and economic inequalities by drawing upon their personal experiences, albeit in a manner that allows them to maintain a distance from their personal lives. This is similar to Mead's notion of "distance experience," in which people move "beyond themselves" into the future, constructing imagined worlds for escaping the circularity of social reproduction of racial oppression. The computational agent becomes a projection of the "self," as well as a representation of the "other," whom students would normally not identify with themselves in their daily experiences.

However, it is important to note that our goal is not to position the computational agent as a panacea. The instructional context for supporting expe-

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

riences of transitional othering involved recontextualizing an algorithmically generated multiagent simulation of ethnocentrism in light of the Racial Dot Map. We also provided an example from the field of computational social science that highlights the importance of progressive recontextualization of generalized computational abstractions in advancing scholarship on computational modeling of ethnocentric phenomena. In our classroom study, going beyond the technocentric imagination, we provided phenomenological accounts in terms of the lived experiences that the different participants brought into play in their discussions, once they were able to recontextualize the ethnocentrism simulation in terms of the Racial Dot Map. It is through the experience of recontextualizing that preservice teachers were able to dive deeper into their explanations and understandings of the simulation as a representation of racial and socioeconomic inequalities. The computational abstractions used in the ViMAP simulation (e.g., the agent-level variables) became recontextualized as attributes of and interactions between people represented by the dots in the Racial Dot Map.[29]

Transitional othering is an act of *ostranoniye*:[30] making the familiar strange, and perhaps more importantly, making the strange familiar. A form of ostranoniye that is particularly relevant to our work is what critical race scholars have termed *projectivity*—an imaginative generation of possible alternate trajectories of action by racialized actors.[31] Projectivity allows racialized actors to distance themselves from and challenge problematic habits and traditions. It also creates opportunities to reformulate these habits and traditions that otherwise perpetuate racial oppression, marginalization, or silence around these issues (e.g., in our case, the culture of silence around topics that involve the Civil War in classrooms in the Southern US). However, it is not only racialized people who can benefit from such opportunities. As we show in chapter 5, transitional othering enables a nearly all-White class of preservice teachers to engage in critical conversations and reasoning about the complex relationships of race and socioeconomic inequalities in the US. This is a significant finding, given that it has been shown to be challenging for White teachers to engage in such conversations in their K–12 classrooms, even when they are teaching social studies.[32]

8.4 Epilogue: Lessons for Avoiding Technocentrism

In concluding our book, we return to Papert's call to take the literary metaphor seriously in order to develop a culture of "computer criticism." In going beyond a structuralist analysis of language and literature, literary criticism takes on a problem that is also at the heart of phenomenology: *enframing*.[33] The

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

central purpose of literary criticism is to highlight the complex relationships between what is included within such enframings and what is excluded from them. In Heideggerean terms, the frame itself becomes the meaning of language. What happens when we take the linguistic metaphor seriously in the context of educational computing? Our concern here is to avoid coining aesthetic clichés[34]—that is, phrases that have only rhetorical power—rather than actually orienting our actions toward expansive imaginations of coding. Can computational heterogeneity actually reorient our praxis as researchers?

Epistemologically, this book in its entirety is an argument against a particular form of enframing of the experience of coding that relies only or primarily on device-level engagements, typically amplifying our focus on the structural elements of code. Wing's call for computational thinking—however myopic its interpretation and uptake by the educational computing community has been—was fundamentally an attempt to orient our attention to the experiential nature of computing and coding. However, the discourse around computational thinking is largely dominated by a rather superficial conceptualization of computational abstractions, typically represented in terms of structural elements within the programming language. Such an enframing creates divisions that often might not reveal the underlying complexity of the relationships between elements "inside" and "outside" the frame. This is essentially the problem of technocentrism and technological determinism: elements within the technology become primary or sole representations of our experience. In this concluding section of our book, we reflect on some lessons from the field that we have learned over the last decade of *listening* to the voices of hundreds of students and teachers that can help us look beyond technocentrism and avoid technological determinism in our inquiry.

8.4.1 Lesson 1: Worldliness Beyond Microworlds (and Microcontrollers)

> *In the classroom, we give accounts of the world beyond.*
> —Gayatri Chakravorty Spivak[35]

The worldliness of the computer, since Papert's time, has been rooted in the malleability of computing software and hardware. From the shapeshifting versatility of Proteus, the Greek god, comes the metaphor *protean*, which Papert used to describe the computer's power. Wing's notions of computational abstractions and computational thinking provide another way of conceptualizing the protean nature of computing through the *automation of abstractions*.[36] The "capital" of computing is located in the work of representing the world in a microcosm—microworlds—which, while powerful in a protean sense, can also portray an image of "purity, simplicity, and seclusion."[37]

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

Voicing code fundamentally orients us to the worldliness of code beyond microworlds and the microcontroller, even when every chapter in this book involves participants working with microworlds. The images of coding we have presented illustrate how code is voiced through and alongside heterogeneous elements of students' and teachers' lived experiences: their religious lives, designing not only for but *with* others, carework and personal attachments to friends and family members, conversations about race through transitionally othering code, and teachers' reframings of coding as mathematizing. The meaning of code and coding thus becomes significantly amplified, and perhaps more importantly, *transformed*, as students, preservice teachers, and teachers find their computational voices.

The worldliness of code, for educational designers and researchers, must go beyond representing the world inside the microworld. It must untether our imaginations of code and coding from device-level engagements. The transparency of code and computational agents that can be leveraged to represent urban segregation in a microworld must be complemented by attention to transparency afforded through pedagogical talk. And while microcontrollers like Arduinos can expand computing by enfolding materiality—for example, distributed mathematical machines discussed in chapter 5—it is through enlivening the addressivity in discourse that computational utterances find their social horizons. The alterity inherent in designing a computational artifact must be complemented by *being with* users.

And finally, it is important to remind ourselves that our call for worldliness should not blind ourselves to the oppressive and hegemonic histories and ideologies in which STEM disciplines and computing are entrenched.[38] The world that is revealed to us through a critical phenomenology of code must actively work toward making the professional and pedagogical worlds of computing and STEM less oppressive by centering voices from the margins. For example, we must be mindful of the fact that racialized, immigrant labor constitutes much of the computing industry in the US, and developing a deep understanding of the intersectionalities of gender, race, and migration must also inform our understandings of what counts as authenticity in computing education.[39] Authenticity in K–12 computing and STEM education has also been critiqued due to the cisheteronormativity inherent in these fields of practice, which in its assumed naturalness of gender binaries and heteronormative relationships, has systematically and historically ignored, silenced, and oppressed people who identify as queer and non-binary.[40] Another form of critique of authenticity involves decolonizing computing and complexity education in partnership with Indigenous communities, which can also alter our

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

commonly accepted, Westernized notions of programming and modeling using digital programming languages.[41]

While deeper explorations of these specific issues are beyond the scope of our book, by centering worldliness, we essentially argue that a commitment to centering systemically and historically marginalized voices is urgently needed for the field. This, in turn, necessarily involves paying attention to heterogeneity, and the metaphors that we have presented in this book may be helpful. But more importantly, critical phenomenology is a reminder that what we know as coding may simply be a second-order perception that stands to be changed as we listen carefully to voices from the margins. Worldliness, thus, is not merely a call for orienting our attention to the place where code unfolds; it may necessitate changing the world of code itself.[42] It is a commitment to carrying "the burden of recognition"[43] of the injustices that masculine, individualistic, and militaristic disciplinarities of technoscience have historically and systematically enacted on Black people, Indigenous people, women, people of color, and queer and non-binary people, while simultaneously working toward learning to be in solidarity with them.[44]

8.4.2 Lesson 2: Data as Trouble

The technocentric universality[45] implied in viewing computing as the automation of computational abstractions can stand in contrast to the heterogeneity inherent in modeling science. Whereas algorithms and data structures can be used effectively and reflexively in scientific work, engaging with the material world leads to "trouble": kinks and fractures in the assumed infrastructure that constitutes disciplinary work, and ruptures in the seamlessness implied in the automation of abstractions. The automaticity of algorithmic computations of distance traveled by the ViMAP Turtle on one hand, and the messiness and uncertainty involved in students' measurement of step sizes on the other, is an illustration of this contrast (chapter 6). What emerges then is the value of bringing together both these forms of experience within the fold of computational modeling. Furthermore, as Emma's classroom in Year 2 illustrates, teachers can come to value this uncertainty as a site for conceptual and representational innovations in modeling.[46]

As we have noted elsewhere, Dewey's critique of empiricist ontology is similar.[47] Dewey argued that the quest for certainty is the hallmark of modernity and positioned empiricist ontology at the root of this quest.[48] He critiqued empiricist ontology because it replaces the emergent nature of experience with data. Whereas empiricism often represents data as self-evident, Dewey positions data as signifying a phenomenon for further inquiry. Data, or objects for that matter, then, do not represent things-in-themselves, but "are manifes-

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

tations of particular kinds of novel and complex relationships that take place in and through time,"[49] which can only be understood through further inquiry. This pragmatist critique of empiricist ontology pre-dates Heidegger's work, but bears a deep similarity in pointing out that when data is taken to be "self-evident," then knowledge becomes an *antecedent reality*—a view that is in direct opposition to the différant nature of experiences of coding we have illustrated in this book. In Heideggerian terms, such an approach *enframes* knowledge by hiding the experience behind the data, or even making it irrelevant. In such cases, in Merleau-Ponty's terms, there is no "rupture with familiarity"; instead, we fall back on the spheres of givenness—the second-order perceptions of technology—that continue to recursively appear as the answers to our questions about technology.

In positioning trouble as central, and even desirable, we can confront the myopic and problematic discourses around grit and persistance,[50] and reframe failure as an essential, recursive, and iterative experience *where* learning happens.[51] This is particularly relevant for working with marginalized students who have been *systematically*[52] labeled as "failures," a label that is hard to overcome.[53] However, it is also important to note that the heterogeneity in experiences of trouble necessarily go beyond the image of the individual learner. The charismatic computer must not be replaced by the individualistic image of child-centeredness that ignores or omits the important role of joint activity. This in turn necessitates paying attention to assistance from teachers (chapter 6), other students (chapter 3), and intended users (chapter 4), and must also foreground, rather than ignore, students' interpretive dilemmas and challenges that are at once epistemic and representational in nature (chapter 6).

8.4.3 Lesson 3: Code as a Boundary Layer

Our version of computer criticism is premised on positioning coding as *voicing*, a heteroglossic lens we borrowed from Bakhtin. Voicing presents an image of heterogeneity that renders fluid the boundary between natural and artificial languages, between what lives inside the computer and what's outside. This is essential because in the absence of such fluidity, the experience of code is folded onto code itself. *Voicing* opens the space up for bringing into account sociological, political-historical, and affective realms of experience in which coding comes alive in discourse. The question of learning to code can then be reframed as an inquiry into how students and teachers develop their computational voices. The forces that shape their voices include both univocality and multivoicedness, and it is the dynamics between these forces that becomes the phenomenon of inquiry for us as researchers and educators. One could argue that recent studies that rely on data mining the online computational artifacts

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

created by students also reveal how computational abstractions are "voiced," for example, as students engage in "remixing" others' code.[54] However, in the absence of richer analyses of the non-computational dimensions of the experience of the learners, the image of learning to code gets folded within code itself.

Computational heterogeneity is a form of simultaneous ideological broadening and deepening—an ongoing attempt to redraw lines around what forms of experience should *also* count as coding, especially in K–12 science and math. Gieryn introduced boundary work as an ideological style in scientists' attempts to create a public image for science by contrasting it favorably to nonscientific intellectual or technical activities. So, descriptions of science as distinctively truthful, useful, objective, or rational may also be interpreted as ideologies that, although incomplete and ambiguous as images of science, are nevertheless useful for "scientists' pursuit of authority and material resources."[55] In a similar sense, we wonder if contestations around defining computational thinking can be characterized as ideological contestations and commitments, rather than matters of scientific verity. For example, by positioning computational thinking as thinking about computational abstractions or by claiming that everyday situations can be better understood in terms of computational abstractions, we reveal a commitment to generalizablity (see section 1.3.1). Similarly, the perspective that computational thinking is inseparable from representational practices reveals a different form of disciplinary commitment that values uncertainty in technoscientific work.[56] Our book is also an attempt to argue for an ideological expansion and deepening by arguing for a heightened emphasis on heterogeneity, noting that the nature of computational experiences is *fundamentally* heterogeneous—particularly in K–12 STEM contexts—and far more heterogeneous than what has been previously reported in the literature.

Gieryn reminds us that science acquired its intellectual authority through boundary work: the public demarcation of "science" from "non-science." Although this book is not the public sphere and, like many other academic books, will live in a realm that requires privileged access, it is nevertheless an effort to redraw lines of separation and mergings between code and human experience—a form of ongoing distinction and difference-making that is fundamental to ideological constructions of both theories of computing and theories of learning. Shanahan's metaphor of *boundary layers*[57] then aptly captures our accounts of coding in K–12 STEM: new places of encounters and interactions, akin to the places where different fluids meet and create a boundary layer. When seen in this light, code can become a space where disciplinary worlds of computational science can coexist alongside socially and personally meaningful

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

interpretations beyond the designer's imaginations. For example, a cluster of flocking computational birds can become a discursive space for conversations about overcrowded prisons between a mother and her child.[58]

The *experience* of code is different and much more complex than code itself. The literary metaphor at the heart of our work also reminds us that coding, like Bakhtin's language, is a quest for words that are not our own, that are différant from code itself, an act of "seeing more" rather than merely "seeing as."[59] It is this very act of seeing more in which we engaged—reimagining how to account for experiences of coding in expansive ways—that we invite the field to engage in, rather than unproblematically adopting preset categories for looking at computing, including our own. The essence of coding is the heterogeneity of experience, not device-centered commitments to technology, and educational computing researchers must always be alive to it.

Notes

1. M. Eisenberg (2003). Mindstuff: Educational technology beyond the computer. *Convergence*, 9(2), 29–53.
2. Another example of such an "Either-Or" framing is computational thinking vs. computational participation. We direct the reader to Kafai and Burke's book *Connected Code*, which explores computational participation in out-of-school settings. Y. B. Kafai & Q. Burke (2014). *Connected code: Why children need to learn programming*. MIT Press.
3. J. Robertson (2018). Answering children's questions about computers. *Communications of the ACM*, 62(1), 8–9.
4. Y. B. Kafai & Q. Burke (2014). *Connected code: Why children need to learn programming*. MIT Press.
5. S. Papert (1987). Information technology and education: Computer criticism vs. technocentric thinking. *Educational Researcher*, 16(1), 22–30.
6. J. Margolis (2010). *Stuck in the shallow end: Education, race, and computing.* MIT Press.
7. T. Todorov (1984). *Mikhail Bakhtin: The dialogical principle*. Manchester University Press.
8. Todorov, *Mikhail Bakhtin*, 1984, 56.
9. L. McMahon (2017). Phenomenology as first-order perception: Speech, vision, and reflection in Merleau-Ponty. In *Perception and its development in Merleau-Ponty's philosophy*, edited by K. Jacobson & J. Russon. University of Toronto Press.
10. P. Banerjee & R. Connell (2018). Gender theory as Southern theory. *Handbook of the sociology of gender*, 57–68. Springer.
See also: B. Hooks (2000). *Feminist theory: From margin to center.* Pluto Press.
11. S. Ahmed (2006). *Queer phenomenology: Orientations, objects, others.* Duke University Press.
12. W. Faulkner (2001, January). The technology question in feminism: A view from feminist technology studies. In *Women's Studies International Forum*, 24(1), 79–95. Pergamon.
13. J. Margolis & A. Fisher (2002). *Unlocking the clubhouse: Women in computing.* MIT Press.
See also: S. Vakil (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review*, 88(1), 26–52.
14. A. C. Dickes, A. V. Farris, & P. Sengupta (2020). Sociomathematical norms for integrating coding and computational modeling with elementary science: A dialogical approach. *Journal of Science Education and Technology*, 29, 35–52.
15. J. Derrida (2001). *Writing and difference.* Routledge.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

16. For a review, please see: R. L. Goldstone & U. Wilensky (2008). Promoting transfer by grounding complex systems principles. *Journal of the Learning Sciences*, 17(4), 465–516.
17. Even some of our own previous research has shown that the intuitive alignment of the agent perspective with learners' prior experiences makes it possible for even 5th graders to learn complex scientific phenomena that are typically introduced to postsecondary students. See: P. Sengupta & U. Wilensky (2011). Lowering the learning threshold: multiagent-based models and learning electricity. In *Models and modeling*, 141–171. Springer.
18. O. Parnafes (2007). What does "fast" mean? Understanding the physical world through computational representations. *Journal of the Learning Sciences*, 16(3), 415–450.
19. See our discussion of Joyce Fletcher's work in chapter 7.
20. G. Wiggins (1989). A true test: Toward more authentic and equitable assessment. *Phi Delta Kappan*, 70(9), 703–713.
21. See, for example, the "images" of what scientists do that guide how science education has been conceptualized, in R. Lehrer & L. Schauble (2006). Scientific thinking and scientific literacy: Supporting development in learning contexts. In *Handbook of child psychology*, edited by K. A. Renninger & I. Sigel. V. IV. Wiley.
22. See P. Sengupta, J. S. Kinnebrew, S. Basu, G. Biswas, & D. Clark (2013). Integrating computational thinking with K–12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2), 351–380; See also D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, & U. Wilensky (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147.
23. D. Carroll (1983). The alterity of discourse: Form, history, and the question of the political in M. M. Bakhtin. *Diacritics*, 13(2), 65–83.
24. See our discussion on Pickering's *Mangle of Practice* in section 6.2. See also: P. Sengupta, A. C. Dickes, & A. V. Farris (2018). Toward a phenomenology of computational thinking in STEM education. In *Computational thinking in the STEM disciplines: Research highlights*, edited by M. S. Khine. Springer.
25. M. G. Ames (2019). *The charisma machine: The life, death, and legacy of One Laptop per Child.* MIT Press.
26. See our detailed discussion of Deweyan aesthetic experience in chapter 7.
27. P. N. Medvedev, M. M. Bakhtin, & A. J. Wehrle (1991). *The formal method in literary scholarship: A critical introduction to sociological poetics*, 60–61. Johns Hopkins University Press.
28. See our discussion of Joyce Fletcher's book *Disappearing acts* in chapter 7: J. K. Fletcher (2001). *Disappearing acts: Gender, power, and relational practice at work*. MIT Press.
29. See for example our discussion of Jamie's work in section 5.3.
30. V. Shklovsky (2015). Art, as device. *Poetics Today*, 36(3), 151–174.
31. M. Emirbayer & M. Desmond (2015). *The racial order*. University of Chicago Press.
32. See chapter 5 for a detailed discussion.
33. J. Culler (2008). *On deconstruction: Theory and criticism after structuralism.* Routledge.
34. C. Higgins (2008). Instrumentalism and the clichés of aesthetic education: A Deweyan corrective. *Education and Culture*, 24(1), 7–20.
35. G. C. Spivak (2012). *An aesthetic education in the era of globalization*, 335–350. Harvard University Press.
36. J. Wing (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, 25(2), 7–14.
37. Eisenberg, Mindstuff, 2003, 39.
38. M. A. Takeuchi, P. Sengupta, M. C. Shanahan, J. D. Adams, & M. Hachem (2020). Transdisciplinarity in STEM education: a critical review. *Studies in Science Education*, 56(2), 213–253.
39. T. M. Philip, & P. Sengupta (2020). Theories of learning as theories of society: A contrapuntal approach to expanding disciplinary authenticity in computing. *Journal of the Learning Sciences*. Available at: https://doi.org/10.1080/10508406.2020.1828089.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

40. D. Paré & P. Sengupta. (In Press). Queering computing and STEM education. *Oxford Research Encyclopedia of Education*. Oxford University Press.
41. M. Lam-Herrera, I. A. Council, & P. Sengupta (2019). Decolonizing complexity education: A Mayan perspective. In *Critical, Transdisciplinary and Embodied Approaches in STEM Education*. Springer.
42. See for example: D. Paré, M. C. Shanahan, & P. Sengupta (2020). Queering complexity using multi-agent simulations. In *Interdisciplinarity in the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS)*, 1397–1404. ISLS.
43. Philip & Sengupta, Theories of learning. In press.
44. P. Sengupta (2020). Re-orienting design: An unbearable pain. In *International Conference of the Learning Sciences (ICLS 2020).* ISLS. Available at: http://hdl.handle.net/1880/112215
45. See section 1.3.1.
46. See also: A. V. Farris, A. C. Dickes, & P. Sengupta (2019). Learning to interpret measurement and motion in fourth grade computational modeling. *Science & Education*, 28(8), 927–956.
47. Sengupta, Dickes, & Farris, Toward a phenomenology, 2018.
48. J. Dewey (1929) 1984. *The later works of John Dewey 1929: The quest for certainty.* SIU Press.
49. A. Stoller (2018). Dewey's creative ontology. *Journal of Thought*, 52(3–4), 47.
50. S. Vossoughi, P. K. Hooper, & M. Escudé, (2016). Making through the lens of culture and power: Toward transformative visions for educational equity. *Harvard Educational Review*, 86(2), 206–232.
51. P. Blikstein (2013). Digital fabrication and "making" in education: The democratization of invention. In J. Walter-Herrmann & C. Buching), *Fablabs: Of machines, makers and inventors*, 1–21. Bielefeld (transcript).
52. Vossoughi, Hooper, & Escudé, Making through the lens, 2016, 216.
53. L. Martin (2015). The promise of the maker movement for education. *Journal of Pre-College Engineering Education Research* (J-PEER), 5(1), 30–39.
54. See for example: S. Dasgupta, W. Hale, A. Monroy-Hernández, & B. M. Hill (2016, February). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 1438–1449. ACM.
55. T. F. Gieryn (1983). Boundary-work and the demarcation of science from non-science: Strains and interests in professional ideologies of scientists. *American Sociological Review*, 781–795. Quoted segment can be found on page 793.
56. R. Duschl (2008). Science education in three-part harmony: Balancing conceptual, epistemic, and social learning goals. *Review of Research in Education*, 32(1), 268–291.
57. M. C. Shanahan (2011). Science blogs as boundary layers: Creating and understanding new writer and reader interactions through science blogging. *Journalism*, 12(7), 903–919.
58. P. Sengupta & M. C. Shanahan (2017). Boundary play and pivots in public computation: New directions in STEM education. *International Journal of Engineering Education*, 33(3), 1124–1134.
59. Higgins, Instrumentalism, 2008.

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a portion of the eBook at doi:10.7551/mitpress/11668.001.0001

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

This is a section of doi:10.7551/mitpress/11668.001.0001

Voicing Code in STEM

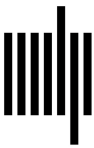
A Dialogical Imagination

By: Pratim Sengupta, Amanda Dickes, Amy Voss Farris

Citation:

Voicing Code in STEM: A Dialogical Imagination
By: Pratim Sengupta, Amanda Dickes, Amy Voss Farris
DOI: 10.7551/mitpress/11668.001.0001
ISBN (electronic): 9780262363075
Publisher: The MIT Press
Published: 2021

The open access edition of this book was made possible by generous funding and support from MIT Libraries



The MIT Press

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024

MIT Press Direct

© 2021 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

The MIT Press would like to thank the anonymous peer reviewers who provided comments on drafts of this book. The generous work of academic experts is essential for establishing the authority and quality of our publications. We acknowledge with gratitude the contributions of these otherwise uncredited readers.

This book was set in Times New Roman by the authors.

Library of Congress Cataloging-in-Publication Data
Names: Sengupta, Pratim, author. | Dickes, Amanda Catherine, author. | Voss Farris, Amy, author.
Title: Voicing code in STEM : a dialogical imagination / Pratim Sengupta, Amanda Dickes and Amy Voss Farris.
Description: First edition. | Cambridge, Massachusetts : The MIT Press, [2021] | Includes bibliographical references and index.
Identifiers: LCCN 2020015012 | ISBN 9780262045117 (hardcover)
Subjects: LCSH: Computer programming–Study and teaching. | Voice computing–Study and teaching.
Classification: LCC QA76.6 .S455 2021 | DDC 005.13–dc23
LC record available at https://lccn.loc.gov/2020015012

Downloaded from http://direct.mit.edu/books/oa-monograph/chapter-pdf/2255857/9780262363075_c001000.pdf by guest on 06 December 2024