danceON: Culturally Responsive Creative Computing

William C. Payne
New York University
Brooklyn, NY, United States
william.payne@nyu.edu

Yoav Bergner
New York University
Brooklyn, NY, United States
yoav.bergner@nyu.edu

Mary Etta West
University of Colorado Boulder
Boulder, CO, United States
Mary.West@colorado.edu

Carlie Charp
University of Colorado Boulder
Boulder, CO, United States
carlie.charp@colorado.edu

R. Benjamin Shapiro
University of Colorado Boulder
Boulder, CO, United States
ben.shapiro@colorado.edu

Danielle Albers Szafir
University of Colorado Boulder
Boulder, CO, United States
danielle.szafir@colorado.edu

Edd V. Taylor
University of Colorado Boulder
Boulder, CO, United States
edd.taylor@colorado.edu

Kayla DesPortes
New York University
Brooklyn, NY, United States
kayla.desportes@nyu.edu

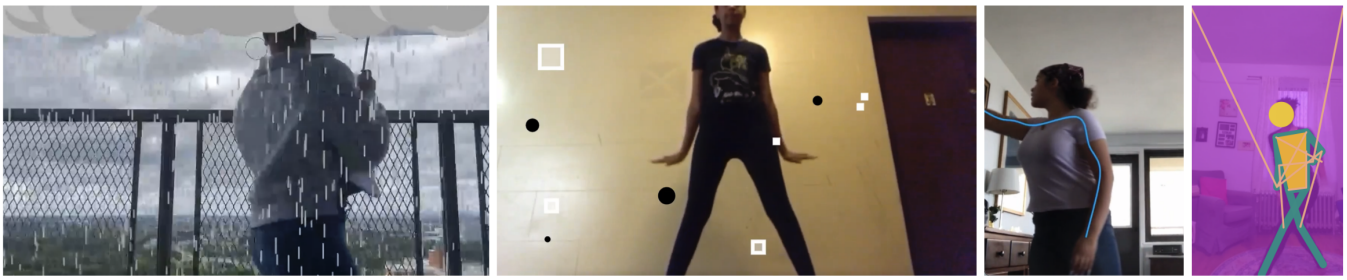


Figure 1: Animations coded in danceON to original choreography. Featuring high school student teams (left to right): Dripping with Power, Black Lives Matter, Love Yourself, and Colorful Escape.

ABSTRACT

Dance provides unique opportunities for embodied interdisciplinary learning experiences that can be personally and culturally relevant. danceON is a system that supports learners to leverage their body movement as they engage in artistic practices across data science, computing, and dance. The technology includes a Domain Specific Language (DSL) with declarative syntax and reactive behavior, a media player with pose detection and classification, and a web-based IDE. danceON provides a low-floor allowing users to bind virtual shapes to body positions in under three lines of code, while also enabling complex, dynamic animations that users can design working with conditionals and past position data. We developed danceON to support distance learning and deployed it in two consecutive cohorts of a remote, two-week summer camp for young women of color. We present our findings from an analysis of the experience and the resulting computational performances. The work identifies implications for how design can support learners' expression across culturally relevant themes and examines challenges from the lens of usability of the computing language and technology.

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445149

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**;
• **Human-centered computing** → *User interface programming*.

KEYWORDS

computing education, data literacy, dance, culturally responsive pedagogy, design based research

ACM Reference Format:
William C. Payne, Yoav Bergner, Mary Etta West, Carlie Charp, R. Benjamin Shapiro, Danielle Albers Szafir, Edd V. Taylor, and Kayla DesPortes. 2021. danceON: Culturally Responsive Creative Computing. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3411764.3445149

1 INTRODUCTION

Data-centric computing systems are increasingly influencing our lives [62]. It is important to empower all individuals to understand and critique these technologies. Both computing and data literacy

education have struggled with inclusion [22, 59, 61]. Researchers have called for culturally relevant learning environments [31, 42, 64] that center computing and data experiences around communities and problem spaces that learners care about [21, 22, 53, 76–78].

Creative or artistic computing can promote reflection on the intent and perception of a computational art piece and generate shareable artifacts that embody learners' experiences, identities, and communities [46]. Data-driven artistic education has potential to expand our notion of sense-making and how it can occur [60]. Fusing data science and creative computing can support learners' integration of skills and knowledge to explore, represent, and experience data in ways that transcend traditional data representation practices [9, 21, 22, 28].

danceON (dance Object Notation) is an educationally-focused programming system for creating visual animations that respond to data from body movement. The system was developed as part of an ongoing Design-Based Research collaboration [6, 83] with a community organization, STEM From Dance[1] (SFD). Our shared goal is to engage young women of color in creating artistic computational artifacts within culturally relevant dance learning experiences.

This paper describes the danceON system and findings from a pilot integration in an SFD summer camp. The work contributes to the literature in three ways:

(1) We illustrate how declarative syntax and reactive behaviors may be designed to integrate body and code in an educational and artistic context.
(2) We present an artifact analysis from an in-situ study of danceON demonstrating early evidence of embodied creative coding and cultural relevancy, which we analyze from the perspective of our design.
(3) We outline implications for the design of systems intended to introduce data science and computing concepts in a creative and equitable manner.

2 RELATED WORK

danceON aims to foster computing literacy by bridging dance and technology in ways that empower users to create innovative and expressive pieces with minimal barriers to entry. To achieve this goal, we draw on innovations from culturally relevant education, embodied learning, dance and computing education, and work exploring the design of programming languages.

2.1 Dance and STEM/Computing Education

Underrepresentation of women of color in STEM fields, including computing, is a complex and persistent problem of equity and (intersectional) identity [4, 76]. Theoretical foundations for pedagogical reforms use the terms culturally responsive [31], culturally relevant [42], and culturally sustaining [64], with differences in nuance and emphasis. Common to these approaches is a recognition that students' lives outside the classroom matter in designing learning experiences inside the classroom. Learners bring their own personal and cultural experiences, values, and knowledge to formal instruction. González et al. [33] recognize that (mathematical) knowledge is embedded in social context and advocates for the transformation of learners' "funds of knowledge" into meaningful, situated activities. Reviewing some of these efforts, Aronson and Laughter [2] found that culturally responsive teaching and relevant pedagogy are associated with increased student interest, self-efficacy, and ability to engage in content area discourses. As a cultural practice deeply embedded in various communities of color in the United States [37], dance provides rich potential for exploring cultural ways of learning [34].

Recent work on dance and computing learning experiences designed for young women of color has demonstrated opportunities for learners to tap into their funds of knowledge and cultural resources [8, 15, 20, 68]. Exploratory studies of data visualization with high school step dancers showed how grounding educational systems in the experiences and passions of users can create engaging and authentic learning experiences [8]. Researchers have also demonstrated potential for learners to bring self-identified themes into a physical computing environment where learners created dances around socio-political issues such as deforestation and bullying [20]. Developers of Dancing Alice [18, 43], which enables learners to program an avatar to dance, observed that learners' prior experiences with dance supported their efforts reasoning through computing challenges. The Virtual Environment Interactions (VEnvI) [19, 44, 45, 65] builds on this work with a library of motion captured dance movements that users can invoke. This work highlights how dance-computing technology can support learners in using their bodies as an object to think through computational problems. The present work extends these explorations with a focus on programming animations that are overlaid on and responsive to video capture of dance.

In addition to Dancing Alice and VEnvI, some recent educational systems targeting both young learners and content at the intersection of STEM and dance include Embodied Physics[2] and Dance Party[3], a block-based environment for choreography using animated sprites. Whereas Dance Party offers an entry-level experience in imperative programming (e.g., loop blocks) with preset affordances (e.g., backgrounds, sprites, and dance moves), danceON uses a domain-specific language to encourage abstraction both about the human body and about user-defined animated objects.

2.2 Programming Languages and Learning

Learning to program is difficult. Decades of research has shown that even when demonstrating success in undergraduate programming courses, learners still struggle to solve computational problems that experts deem trivial [79, 81]. Blocks-based programming languages, such as Scratch and Blockly, were developed specifically to support novices. Research has demonstrated that these drag-and-drop programming interfaces can improve the usability of the coding experience [39, 52] and provide pedagogical benefits such as improved learning and a more discoverable language [71, 85, 86]. However, they are sometimes dismissed as inauthentic to "real" programming [23, 84], and can lead learners to create syntactically perfect, yet buggy programs [50, 80]. danceON pursues a different, complementary, strategy for supporting learning. It is a text-based language, and supports exploration of ways in which alternative

[1] https://www.stemfromdance.org

[2] https://www.terc.edu/projects/embodied-physics/
[3] https://code.org/dance

programming paradigms can ameliorate issues learners face with interpretability and applicability of code.

When programming, learners often face difficulties understanding what a computer can and cannot do [63, 66]. Even for experienced programmers, the meanings of programs may be surprising [41, 70], in part because languages describe computational processes as a function of the lexicon and behaviors of the programming environment rather than of the domain. A programmer must map the nouns, verbs, and relations of a non-computational domain to those of the programming paradigm (e.g. loops, variables, etc.), inventing an ad-hoc machine-readable representation of their domain to express solutions to a target problem [63]. Beginner programmers must do this translation into computational form while simultaneously learning the programming language, paradigm, and notional machine that they are attempting to use [24]. Domain Specific Languages (DSLs) offer one solution to this.

DSLs, such as the music coding environment *Impromptu* [7], close the distance between the semantics of a domain and those of a programming environment used to construct tools for use within that domain. Researchers have just begun to investigate the affordances of DSLs and related TSLs (Task Specific Languages) for easing students' and teachers' entry into programming [17, 55, 56]. danceON explores how a DSL can support learners' semantic understanding of animations that respond to body movement and position data into a culturally relevant experience.

2.3 Opportunities in Declarative and Reactive Languages

Computing education intended to cultivate Computational Thinking (CT) – defined as the use of abstraction and automation for problem solving by an information processing agent [57, 87] – has typically made use of general-purpose, imperative programming languages. These educational experiences and tools emphasize concepts and practices like sequencing instructions, using variables to store and access information, and controlling program flow with conditionals and loops [3, 11]. In contrast, danceON explores the use of a declarative language where one does not specify the sequential processes for program execution, but instead *declares* what the properties of a desired program would be. danceON is heavily influenced by Vega-Lite [73], a declarative DSL for creating data visualizations. Vega-Lite's users declare properties of a visualization, including the type of representation (e.g. bar graph), the mappings between data attributes and visual elements, and the interaction possibilities available to a viewer. Vega-Lite's user community[4] demonstrates the affordances of a declarative DSL to enable users to craft clear and sophisticated programs without need for imperative programming. danceON attempts to offer these same affordances, adapting their approach for use in generating dynamic visual overlays for dance videos.

In addition to being declarative, danceON integrates a reactive programming paradigm. Similar to declarative programming, reactive programming does not require the programmer to describe all of the sequential operations that the computer must execute. Instead, one describes sets of relations between values, and the computer automatically propagates changes to dependent values [5]. To illustrate reactivity, consider following code:

```
a = 5
b = 3
c = a * b
b = 0
```

If executed by Python (an imperative language), the value of `c` at the end of this program's execution would be 15, but in a reactive language, the value `c` could be `0` because the value of `c` is a product of `b`, and `b` is `0`. The lines of code do not denote the sequence of program execution but a set of relations. Microsoft Excel formulae are also reactive expressions. The reactive paradigm has been used successfully in education: The Elm, Racket, Flapjax and Pyret programming environments [51, 69] all provide reactive frameworks for creating interactive Graphical User Interfaces (GUIs). Racket [27, Section 2.4] has been used extensively within introductory computing education [26, 54, 75]. We are intrigued by the potential of the reactive approach to reduce the amount of state, and thus complexity, that must be maintained by programmers, novice and otherwise, who wish to create event-driven visualizations.

Researchers across data visualization, robotics, and programming languages have explored fusing declarative and reactive programming paradigms. For instance, Satyanarayan et al. [74] describes the design of Reactive Vega, a language that allows declarative description of visualizations, while Peterson et al. [67] and Huang and Hudak [40] describe DSLs for declarative and reactive robot control. In this paper, we explore how the fusion of declarative and reactive programming might be applied to support computing education for dynamic and artistic dance video overlays.

3 DESIGN OBJECTIVES

3.1 Context of Design Research

We developed danceON as part of a collaborative Design-Based Research investigation [6, 83] with STEM From Dance. SFD is a community organization that creates dance and computing experiences intended to introduce young women of color to STEM fields. SFD was founded in 2012 with the goal of "tackl[ing] diversity in the STEM workforce" through providing girls of color with "access to a STEM education by using dance to empower, educate, and encourage them as our next generation of engineers, scientists, and techies." SFD programs are intended to develop participants' STEM and dance skills, and to facilitate their development as confident, resilient, curious, and creative people. These experiences are meant to be fun and not feel like school or forced learning. SFD wants learners to take with them an understanding that STEM fields apply to their interests and that they have the capabilities and support to continue on. The community extends beyond the current participants, instructors, and organizational staff to professionals in the STEM workforce and alumni from SFD who frequently return to share their experiences. Family and friends join participants on "family days" to witness final performances. In order for new technology to be adopted, the designs must first and foremost adhere to SFD's core values of making learners feel supported and capable, and equip learners with tools to make polished art pieces that they are proud to share.

[4] https://github.com/vega/vega-lite/issues

Before our engagement, SFD already had curriculum for programming animated projections with Processing [49] and wearable electronics with microcontrollers but wanted to expand their offerings. In past activities and performances, technology typically consisted of blinking LEDs and projected animations that were hard-coded to be synchronized with the music and were not interactive. SFD's founder hoped that technology could be more integrated with the learners' dancing such that it could "inform their dancing." Additionally, she wanted to broaden the types of STEM disciplines that participants could experience. Data science and machine learning offered SFD value because they are "hot" topics that their partners and funders would recognize, and they provide additional avenues to create responsive technology.

3.2 Formative Design Work

We determined the design requirements of danceON through 11 formal interviews and 20 design meetings with SFD participants. The semi-structured interviews were conducted with the CEO and founder of the organization, six instructors, and four learners. The interviews were designed to identify the values of the participants, their experiences in the organizational activities, their current challenges, and their future goals. Four researchers employed thematic analysis [10] on the transcribed student and instructor interviews in which we iteratively honed themes through a recursive process involving identifying, defining, applying, and refining. A detailed analysis of this data is outside the bounds of this paper which centers on the danceON system and is in preparation for separate publication. The analysis informs our understanding of the philosophy, values, and culture of SFD that led to embedding these values in our design.

In conjunction with the interviews, the researchers engaged in a series of weekly meetings with two of SFD's leads: their founder and CEO, and their Director of STEM and Art Education. The meetings took place six months prior to the summer program in which danceON was eventually deployed and provided a space to share ideas, receive feedback on curricular and technology designs, and develop plans for implementation. Throughout these meetings, we refined a set of design goals. While we initially intended to work with learners in person, COVID-19 forced us to move the summer program to a remote format, to explore how to teach dance and computing in a virtual, collaborative context, and to define what a virtual performance would constitute. To guide these adjustments, the organizational leads gathered and shared with us artistic performances they thought would be culturally relevant and engaging for their participants to emulate in projects integrating computing and dance. We watched and discussed music videos including Bruno Mars's *That's What I Like*, famous dancers on Instagram like Kida The Great (@kidathegreat), and collaborations between artists and technologists, like Maya Mann and Google Creative Labs.

3.3 Design Goals

We derived three core principles for danceON from our interviews and discussions with the SFD leadership: 1. danceON should be personally and culturally relevant to learners, 2. it should situate art and code as mutually informing, and 3. it should deeply support embodied learning of computer science concepts. We further divide these principles into six design goals:

3.3.1 Personally and Culturally Relevant.

G1 Low barrier to entry: Clearly, accessibly, and immediately incorporate learner interests, abilities, and the styles and cultures of dance they care about.

G2 Wide room for expression: Enable learners to progress on meaningful projects in which they are not constrained and are able to express their complex identities, beliefs, passions, and interests.

3.3.2 Creative Coding: Computing and art making are intertwined and bidirectional.

G3 Art motivates code: Ideas, music, and choreography present clear opportunities to code unique, reactive animations.

G4 Code motivates art: The act of writing code and observing the results leads to new opportunities for creative expression through remixing the code and making new movements.

3.3.3 Embodied Learning: Body and movement are engaging metaphors for learning important computing concepts.

G5 Data literacy - Make transparent the body position data captured by computers and/or sensors while providing an accessible interface to empower learners to understand, use, and manipulate their own data for the purposes of making art.

G6 Machine learning - Help learners understand the limitations and biases embedded in the processes with which computers see them, and introduce learners to the process of training a machine.

For the preliminary design of danceON, we focus on the first four of these principles to understand the usability, authenticity, and creative capacity of danceON. Notably, the bidirectional relationship between art and code (G3, G4) echoes the "augmented expression" guideline for professional interactive performances identified by Gonzalez et al. [32]. While danceON is architected to consider G5 and G6, understanding the pedagogical utility of danceON is critical future work and the subject of on-going longitudinal study.

4 THE DANCEON SYSTEM

danceON enables users to upload dance videos (or use their webcam) and write code that creates responsive animations based on pixel location data and pose-detection data (Figure 2). The live coding environment implements a Domain Specific Language (DSL) that updates the video panel with animations as the user is coding allowing the user to quickly test and iterate on their code. The web-based IDE provides feedback to users as they are working. Users can toggle between overlays to gather information on pixel location and pose-detection to assist them in making decisions as they work. In the following section, we describe a subset of danceON's core features, highlighting how they tie back to our design goals (§3.3) and meet the needs of novice programmers integrating dance and animation.

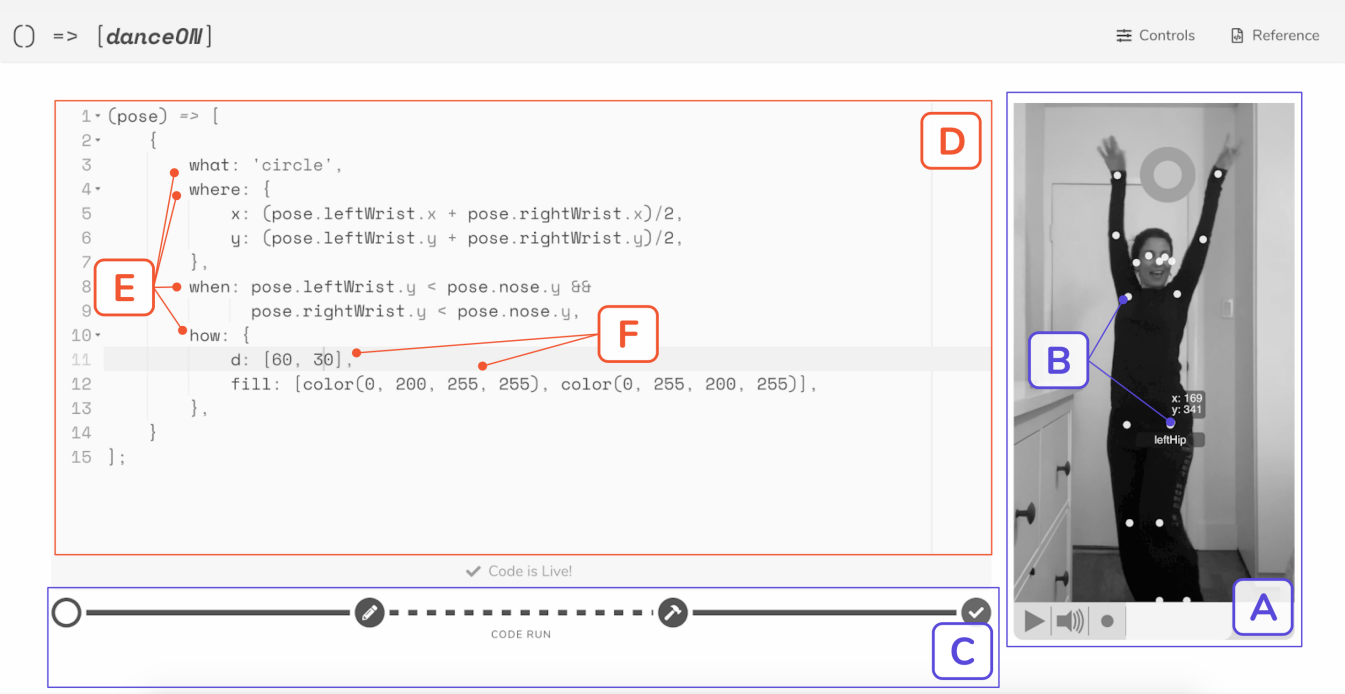


Figure 2: danceON IDE: A. Video/webcam player (4.1) with built-in pose detection and classification (4.2); B. Skeleton and Cursor overlays (4.3); C. Live coding environment features including feedback bar (4.4); D. Code editor (4.5.1); E. Natural language object properties (4.5.2), F. Automatic lifting or list handling (4.5.3)

4.1 Video Window

danceON integrates a media player supporting webcam and uploaded video inputs, multiple pose detection/classification tools, and interactive overlays for problem-solving and translating between virtual and physical spaces. The webcam mode allows users to capitalize on live video to quickly test and iterate on code. For example, a learner may define a pose condition, move their body in real-time to test the boundaries of that condition, and then update their code. Additionally, instructors can demonstrate new concepts, techniques, and learner ideas in real-time (G3: Art motivates code, G4: Code motivates art). Because of the computing power necessary to run pose detection in real-time, users can also upload dance videos for improved pose-detection accuracy. Another benefit is that code is always live even when a video is paused. Animations remain reactive while the performer can be frozen in place. Thus, users may isolate specific regions of video and scrub frame by frame to check if and when a pose condition is met or fine-tune the behavior and appearance of an animation (Figure 3). When finished, users can download their dance videos with the programmed animations now integrated into their dances.

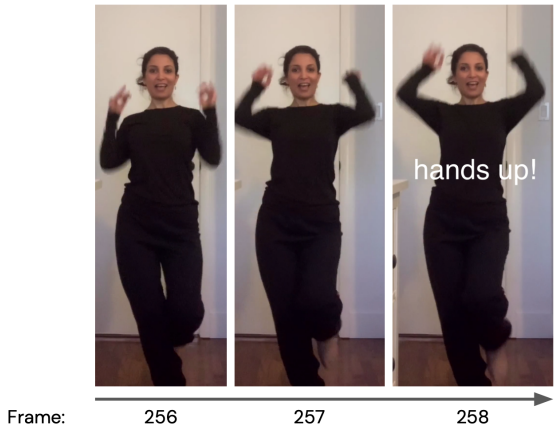


Figure 3: A user can scrub one frame at a time to observe and fine tune the behavior of their code including to check the exact frame when a condition based on body position evaluates to true. In this example, the text "Hands up!" is displayed when both wrists are above the nose.

4.2 Pose Detection: PoseNet, OpenPose, & Teachable Machine

danceON is intended to support video processing of dances recorded in home environments without any additional sensors or equipment such as a green screen (G1: Low barrier of entry). Pose estimation in danceON uses the open source computer vision libraries PoseNet [82] and OpenPose [13] and provides horizontal and vertical coordinate estimates of body parts (specifically: eyes, nose, ears, shoulders, elbows, wrists, hips, knees, and ankles). PoseNet runs in the web using TensorFlow.js supporting real-time pose estimation, but in

early testing we found that OpenPose produced more accurate offline results. Given our goal of supporting learner growth through enabling high-quality projects (G2: Wide room for expression), we did not want data quality to be a limiting factor. In our current system, learners may upload their videos to be processed remotely via OpenPose and then can upload the resulting JSON files to danceON.

danceON also integrates with Teachable Machine, a web-based tool in which users can rapidly train models to differentiate between poses [14]. Once they train a classifier, they can export the model and upload it to danceON. The classifier can then be used to trigger animations on poses its been trained it on. While integrated into the tool, this feature was not tested in our study due to timing constraints.

4.3 Position & Pose Overlays

To facilitate learners connecting their body and movement to its data representation, we implemented two toggleable overlays: Pixel Position and Skeleton overlay (shown in Figure 2). The Pixel Position overlay displays the x and y coordinates of the mouse pointer. It is intended to aid reasoning within the virtual canvas helping learners find points of interest, e.g. to find absolute coordinates to relate body position against and to render a shape over; or to estimate the relative distance between body parts while a video is paused. The Skeleton overlay draws indicators over all body parts captured by the computer vision algorithm. When hovered over, the Skeleton overlay displays each part's ID for reference in code (e.g. `leftAnkle`, `rightWrist`, etc.). The Skeleton overlay also clearly exposes inaccurate and missing data. While the overlays highlight data representations of one's body (G5: Data literacy), they are also intended to aid in problem solving (G1: Low barrier of entry) and converting artistic ideas into code (G3: Art motivates code).

4.4 Live Coding IDE

danceON is a live coding environment that integrates features for programmers such as syntax highlighting, error checking, and auto complete for functions, objects, and properties. Live coding provides real-time feedback and connections between artistic output and code as evidenced in prior work with music [1, 48, 72] (G3: Art motivates code, G4: Code motivates art). The program is always active providing immediate feedback [58] as virtual objects displayed always reflect the current state as long as there are no syntax errors. In practice, live coding manifests in two distinct ways depending on which media stream is currently active (§4.1). If the webcam is toggled, the user simultaneously codes and physically moves their body to render animation changes. If a pre-recorded dance is used instead, the user codes and manipulates a virtual body through playing/pausing/scrubbing the video. Both cases constitute live coding as programs render immediately and react to changes in the active media.

A consequence of liveness is the regular occurrence of errors that appear as the user adds and edits code. Syntax errors (e.g. missing commas or brackets) prevent code from being interpreted while semantic errors (e.g. undeclared variables) produce unexpected behaviors. danceON tracks code across four states: empty, syntax errors, semantic errors, and correct. It communicates the current state to the user via a bar below the editor (Figure 2) and an error message overlay on the video panel if applicable. The intention behind this feature is to make clear that errors are not mistakes or barriers but an ongoing part of the programming process, as the indicator shifts constantly across states as the user drafts code. Intended to reduce frustration, danceON maintains a history of error-free code segments, allowing a learner to "Revert Back" to prior working code (G1: Low barrier of entry).

The interface comes pre-populated with code examples and a short dance performance video to start experimenting with in the IDE (G1: Low barrier of entry). The code examples are intentionally plain encouraging users to understand and remix the code and develop their own style (G2: Wide room for expression). In contrast, we asked a staff member from SFD to record a short video of herself dancing in her own style at home to serve as a model for the learners (G1).

4.5 danceON Language Features

danceON is a Domain Specific Language (DSL) [16] that is built in Javascript and uses the browser-based drawing library p5.js [49]. The language supports numerous shape primitives including circles, squares, lines, triangles, points, etc.; text; and various parameters of customization including size, fill (color and transparency), and stroke.

4.5.1 Declarative Grammar, Reactive Behavior. danceON uses a concise, *declarative grammar* to specify the position and behavior of virtual shapes and *reactive behavior* meaning that virtual shapes always reflect the current state of an incoming data stream, e.g. the current position of a moving dancer. danceON aims to clearly define the boundaries between the physical body, body data capture, code, and visual representation.

Consider a learner who wants to hold a glowing orb in her hands. The following code written in Javascript using p5.js composes [25] the statements that integrate body, code, and animation with other constructs to draw a circle in between her hands.

```
let xPos, yPos, pose;

function setup() { ... }

function draw() { // main loop
    pose = runPoseDetectionAlgorithm();
    if (pose != undefined) {
      xPos = (pose.leftWrist.x + pose.rightWrist.x) / 2;
      yPos = (pose.leftWrist.y + pose.rightWrist.y) / 2;
        circle(xPos, yPos, 30);
    }
}
...
```

The danceON code below removes the imperative description of how the program should execute, including variable assignments, a conditional statement, and the draw loop. By reducing need for plan composition in a general purpose programming language, danceON foregrounds the relationship between body data and the properties of a shape drawn in reaction to those data both lowering the barrier to entry (G1), and expanding opportunities for movement and code to interact (G3, G4).

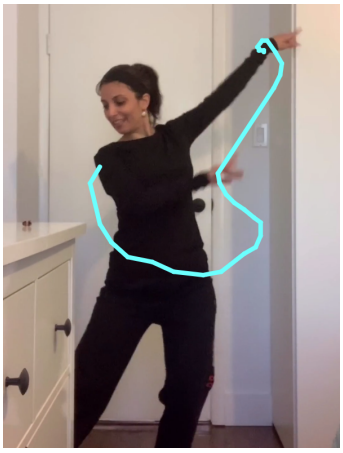


Figure 4: Painting to the screen using past position data.

```
(pose) => [
    {
        what:'circle',
        where: {
          x: (pose.leftWrist.x + pose.rightWrist.x) / 2,
          y: (pose.leftWrist.y + pose.rightWrist.y) / 2,
        }
    }
];
```

4.5.2 Natural Language. A danceON program consists of a list of objects that define the behavior of virtual shapes and text in natural terms using the keywords `what`, `where`, `when`, and `how`, (Table 1). For example, `where` applies to coordinates and geometry, (e.g. "`where` should that line connect to?"), while `when` applies to conditionals, e.g. ("`when` I move my hands up, show the words "boom!"). While these generic words might be ill-suited for use in a general purpose programming language where they could interpreted in limitless ways, their meaning is heavily constrained in the context of danceON, a DSL supporting very specific and limited effects. The `what` property is mandatory (i.e. there will be an error message without it) while other properties possess default behaviors if not specified. For example, if the user does not specify `where` to draw something, danceON will draw it at random locations on the screen. This lowers the requirements to create a "valid" program so users can more swiftly implement and then expand their code using additional keywords to control their animations (G1: Low barrier of entry).

4.5.3 Automatic Lifting. danceON supports "auto-lifting" in which properties of objects may be either single values or lists. This technique has been implemented in other computing curricula and has demonstrated promise for simplifying what would normally require iterative or recursive constructs [27, 29, 36, 75]. The chief data source in danceON is a stream of movement data up to the current moment in time–i.e., a list named `poseHistory`. Auto-lifting eases accessing that data and supports a conceptual and creative leap from working solely with the current body position or `pose` ("where am I now?") to working with the history of positions up to the current point ("where have I been?") (G3: Art motivates code, G4: Code motivates art). The position history can be used in estimating physical properties like velocity as well as underpin sophisticated animations like tracing along the screen, silhouettes that move in canon behind the dancer, and others. For example, the (quite advanced) danceON program below produces an animation of painting to the screen, captured in Figure 4. It uses `filter` to isolate 60 recent body positions and `map` to access coordinates from the body part `leftWrist`. By using two lists at an offset, (indices `0--59` and `1--60`), the program draws line segments between each successive point.

```
(pose, poseHistory) => [
  {
    what: 'line',
    where: {
      x1: poseHistory.filter((_, i) => i < 60)
                     .map(p => p.leftWrist.x),
      y1: poseHistory.filter((_, i) => i < 60)
                     .map(p => p.leftWrist.y),
      x2: poseHistory.filter((_, i) => i < 61 && i > 0)
                     .map(p => p.leftWrist.x),
      y2: poseHistory.filter((_, i) => i < 61 && i > 0)
                     .map(p => p.leftWrist.y),
    },
    how: {
      stroke: 'cyan',
      strokeWeight: 4,
    },
  }
];
```

5 CONTEXT DRIVEN ITERATIVE DEVELOPMENT

We deployed and studied danceON within a remote SFD summer camp using methods drawn from Design Based Research (DBR) practice in education [6]. We partnered with the organizational staff and instructors, iterated on danceON and its curriculum in response to emergent needs and learner interests (§5.3), and captured rich accounts of the intervention and its context (§6). While a messy, real-world environment lacks some of the control that a laboratory study would offer, proponents of DBR in the Learning Sciences express concerns about the ecological validity of evaluations that occur outside an authentic context [83]. The method provides an understanding of the complexities of the educational innovation in practice. In essence, work with youth and instructors was essential to our formation of danceON, and we consider insights gained from deploying early-stage tools both as part of our design process and as a way to assess the design. As demonstrated in prior DBR research, this work feeds into the iterative development of danceON ensuring that it becomes increasingly aligned with learning theory, design, measurement, and practice over time [38].

5.1 Overview of the Camp

During the remote summer camp, two separate cohorts engaged with danceON across eight days of instruction over a two-week span with a STEM instructor and dance instructor. Both cohorts

Property	Description	Example(s)
what	Type of virtual object to draw.	"circle", "text"
where	Static or dynamic position binding. Varies across virtual object types. Random if not declared.	{x: 50, y: pose.leftWrist.y}
when	Condition for whether to draw.	true, pose.leftWrist.y <pose.nose.y
how	Appearance description. Varies across virtual object types.	{d: 30, fill: color(0, 255, 255, 255)}

Table 1: Virtual Object Properties

	Statement	Psuedocode/Code
Instructor Example	When I raise my right wrist, a small purple circle follows my right wrist.	what: circle where: my right wrist when: I raise my right wrist how: small, purple
Learner A	"When someone's left wrist is raised above y:180, a small yellow triangle appears on their left wrist."	what: 'triangle', when: pose.leftWrist.y <180, where:{ x1:pose.leftWrist.x+10, y1:pose.leftWrist.y+10, x2:pose.leftWrist.x-10, y2:pose.leftWrist.y-10, x3:pose.leftWrist.x+20, y3:pose.leftWrist.y+-10,}, how:{fill:color(225,225,28,255)}
Learner B	"When I raise my head's position, small blue circles come out."	what: 'circle', when: pose.rightEye.y <160, where:{ x: pose.rightEye.x, y: pose.rightEye.y, }, how:{fill: color(0,0,255,255)}

Table 2: Superhero Challenge: Learner code excerpts and statements from Day 4 of Week 1.

were entirely young women in high school without prior programming instruction or experience. 11 learners enrolled in the first cohort, though 2 dropped out before submitting their final project contributions. 10 learners enrolled and completed the second camp. During the first four days, learners were introduced to danceON as they worked independently on a "superhero challenge" where they wrote English descriptions and implemented and modified corresponding code in danceON. They submitted their progress each day as they learned new danceON properties (Table 2). In days 5–7, learners formed teams and worked on group projects that were due the morning of day 8, leaving time to ensure individual videos could be produced and sent to a video editor to stitch them together.

While the camp ran from 9am to 3pm each day, only 1 to 2 hours of time was allocated to technical instruction and project work with danceON. The remainder of each day consisted of social time, dance instruction, and presentations by invited speakers. During the first week of each camp, researchers acted primarily as passive observers only occasionally asking questions or helping debug code. We wrote observational notes and retained learner brainstorming and collaboration documents, code snippets, and dance videos.

5.2 danceON Usability Challenges

Observing learner progress remotely was difficult because we rarely had access to screens. Occasionally learners shared their screens to request help and most submitted superhero code progress at the end of the first four days. We witnessed syntactic and semantic mistakes, especially early in each camp. Learners submitted code containing missing or mismatched brackets or commas and occasionally wrote programs misaligned with written goals. For example, at the end of day 1, a learner submitted the code along with the statement "When I move my right arm, a blue triangle follows my right arm."

```
what: 'triangle',
where: {
    x: pose.rightArm.x,
    y: pose.leftArm.y,
```

The above contains multiple errors:

- Missing curly braces.

- Mismatched goal statement: A coordinate is drawn from the left arm.
- Incorrect object properties: A triangle requires coordinates of three points: `x1--x3` and `y1--y3`. The learner may have modified circle code, which can be described with single `x` and `y` coordinates and a diameter.
- Missing body part: An arm property is not included in data outputted by our pose detection algorithm, unlike shoulders, wrists, or elbows.

While most code segments submitted did not include so many errors, and the learner submitted a correct, expanded program three days later adding color and referencing the shoulder, it reflects common mistakes. We did not observe learners using the IDE features to debug their code, e.g. inline syntax error notification, autocomplete, etc. We also did not observe learners referencing or copying from the embedded help examples. Because there was so little technical work time during lectures, the instructor did not model debugging strategies to the learners.

5.3 Important Features Implemented During the Camp

As part of our ongoing Design-Based Research process [83], we made four substantial changes to danceON based on needs that arose during project work so that learners could achieve their artistic ideas in code (G3: Art motivates code), and could implement high quality performances to full songs with multiple animations (G2: Wide room for expression).

5.3.1 Access to Video Time. Initially, we withheld access to the playback time of an uploaded video to ensure that learners could only trigger animations through body position and movement. We were concerned, based on past performance examples gathered during instructor interviews, that learners would ignore opportunities to define choreography and code in direct relation to each other (G3, G4), and instead use time as the sole mediator between animation and code. However, the decision to withhold access to time impeded learner progress on projects where they isolated specific sections and lyrics to divvy up work and code discrete animations. We provided learners access to time within an object's `when` property, (e.g. `when: video.time() < 10`). This enabled learner plans, but resulted in "psuedo-imperative" programs in which previously stateless objects were listed in order as if to be executed in time.

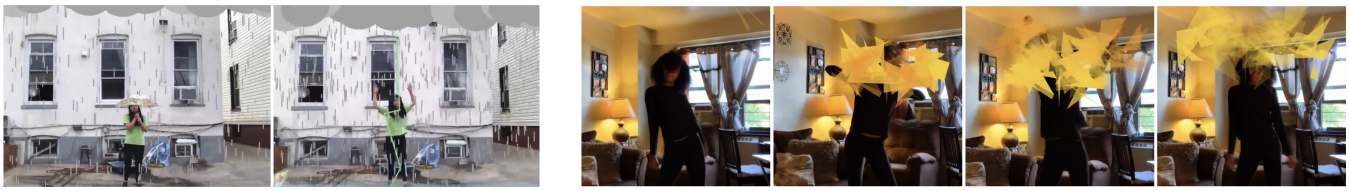
5.3.2 Recipes. Instructors and researchers created new code segments intended to be modified, or "recipes," based on learner discussion and keywords written in brainstorming documents. We made this decision in response to two limitations. First, by using an in-progress DSL and not a widely used creative coding environment, learners lacked existing examples they could access online and use in their projects. Second, while virtually everything learners proposed was possible in danceON, instructors and researchers expressed concerns that learners would not be able to implement everything from scratch in three-and-a-half days. While their superhero statements (Table 2) referenced single shapes, they were not given instruction on combining or repurposing shape primitives to simulate more realistic phenomena. Thus we built and dispersed recipes including "rain", "fire", and "ribbons." At times we built recipes in advance and used structured time to encourage learners to play with and discuss them, while at other times we built recipes live using structured time to describe their inner-workings. When possible, we conveyed to learners that recipes should be thought of as starting points and encouraged learners to not only integrate code in their environment, but to adapt it for their specific video and use it as an example to aid in crafting choreography that takes the animation into account.

5.3.3 Generic and Specific Shapes. To close gaps between what danceON initially supported and observations of learners' expressive desires, we added additional shapes beyond the built-in primitives. We added generic `shape` and `curve` objects that could use any number of vertices allowing for irregular, complex contours and animations over multiple points of the body (e.g. to create lightning that travels vertically down a body or a wave that curves from one wrist across the chest to the other). We also added a heart shape, accessible via the `what` property, to support a team whose theme was "Love Yourself." While a heart could certainly be made using the primitives, we felt it was more important for learners to plan how to use the heart in their videos rather than figure out how to create one.

5.3.4 Data Cleaning. Learners recorded videos under many conditions: indoors, outdoors, dark lighting, etc. In many videos, partial regions of their bodies were temporarily or entirely out of frame. As a result, pose detection algorithms performed with mixed success, occasionally mispredicting body part positions or leaving out body parts entirely. For example, an ear would not be detected if a learner tilted her head sideways to the camera. Because videos were due near the end of the camp, there was no time for retakes or for teaching learners to write code that takes into account incorrect or missing data. Given SFD's main objective of supporting confidence development, we were concerned that learners would feel embarrassed if their videos showed obvious glitches in the public showcase. As a result we made two changes. First after both camps, instructors and researchers made slight modifications to learner projects to catch some blatant instances of disfigured shapes resulting from missing data. Second, before the second camp, we implemented a hidden heuristic condition that replaces a missing value with its last known position. We did not notify learners of this feature.

6 OVERVIEW OF LEARNER ARTIFACTS

Four teams of four-to-six learners created performance videos. Each video consisted of a unique theme and song selection, original choreography, animations coded in danceON, and dance videos. Learners recorded the videos individually, and the danceON-embellished videos were stitched together by an editor following learner instructions. To guide learners, instructors followed a similar process and timeline for both cohorts. On Day 3, learners worked independently to generate thematic, aesthetic, and musical ideas on brainstorming sheets. Instructors synthesized learner documents into four project ideas in the first cohort–Rainfall, Black Lives Matter, LGBTQ, and Female Empowerment–and six project ideas in the second cohort–Smoky Day, Colorful Escape, Love Yourself, Stronger Together, African Identity, and Hip Hop in Space. They provided



(a) Dripping with Power

(b) Black Lives Matter

Figure 5: (a) Virtual clouds fall, water levels rise, and lightning flashes as learners dance to the song "Rain" with an umbrella. (b) Animation and choreography representing the song lyric, "burning into flames"

learners with brief descriptions and possible song choices and asked learners to rank their favorites. On Day 4 of each camp, the last day of structured danceON learning, instructors determined the projects and assigned learners to teams: Dripping with Power/Black Lives Matter in the first camp and Colorful Escape/Love Yourself in the second camp. Below, we include a thick description of one team Dripping with Power's process to detail how danceON was used in its remote, collaborative context, and then we outline the artifacts produced by the other three teams.

6.1 Dripping With Power

6.1.1 Ideation. Dripping With Power contained five learners who ranked "Girl Power" or "Rainfall" as their top choice. They decided immediately to organize their video into two segments: a monologue, that they hoped could represent female empowerment, and a dance section. In attempting to find a connection between two seemingly disparate themes, one learner remembered an impactful scene from "Hidden Figures," a film that depicts three African-American female engineers who encounter racism and sexism while working in a segregated NASA unit in 1960's Virginia. In the movie scene, one engineer is dripping wet and explains to her white colleagues that she has been absent because she needs to regularly travel half-a-mile outside to reach the nearest bathroom allowable for use to black people. Agreeing that the clip fits the theme, Dripping With Power then selected a song: a slowed-down remix of the classic R&B love song "Rain" by the vocal trio SWV (Sisters With Voices).

6.1.2 Planning. Learners set out to work on transcribing the speech from the movie and copying the song lyrics found online in a shared Google Doc. Then, one learner shared her screen with the music playing while others determined the time stamps of section breaks (Intro, Verse - 0:45, Chorus - 1:16), and decided that two teams of three would independently choreograph the first two sections, while everyone would work together on the third section. One learner, after demonstrating her acting prowess, convinced the group that she could handle the monologue. With all roles set, the group worked to identify opportunities for animations. Beginning with the monologue, they highlighted the phrases "simple string of pearls" and "skirt below the knees" as imagery that could be emphasized through animation. Then processing the lyrics, learners recognized that the songwriter connects rising emotion with heavier and heavier rainfall progressing from "misting rain" to "raindrops" to "a dam at capacity." They imagined an increasingly intense storm as the dance progressed assigning themselves to work on clouds, water, lightning, and rain. They decided to use an umbrella as a prop they would dance with and open to shield themselves from the virtual storm.

6.1.3 Implementation. The Dripping With Power team finished planning on Day 1 leaving two days to choreograph, code, record dance videos early enough to use the offline OpenPose algorithm to capture higher quality position data, and produce the final animated takes. It became clear that instructors and researchers would need to take a larger role in writing code examples and directing learner progress. The researcher working with Dripping with Power required learners to begin, end, and customize each code segment, but live-coded some of the objects when communicating syntax over Zoom became too tedious. With some guidance and suggestions, the learners coded individual shapes - a cloud as a group of overlapping grey ellipses, water as a semi-transparent rectangle, lightning as a transparent shape with bright strokes that connect through different body parts during a specific pose, and a raindrop as a small vertical line flashing at a random position (Figure 5a). The researcher went beyond the scope of the lessons and demonstrated/live coded how to make each of these shapes move. Adding randomness caused the lighting to quiver. Mapping song time to the y-value of some shapes of caused the clouds to descend and water level to rise. Rain expanded from a single drop to a full storm through passing a list of random raindrop positions whose length was determined by playback time. Ultimately, the researcher combined each of the individual storm elements uploaded by the learners into one file and asked learners to get it working on their devices and to "remix" it by changing the behavior and color of the lightning to match their shirts. The team ran out of time and was unable to add any animations to the monologue, and unfortunately, one learner was unable to attend the final day of camp and did not submit a video for inclusion in the final cut.

6.2 Black Lives Matter

Black Lives Matter consisted of five young women, though one dropped out. They selected the song "Freedom" by Beyonce and used animations to represent specific lyrics. For example, during the lyrics "I'm telling these tears, go and fall away, fall away, May the last one burn into flames," two learners sway their arms over and around their heads while "flames," red and yellow, semi-transparent triangles, emanate from their faces travelling upwards and disappearing (Figure 5b). Other short animations include text displayed to the screen, chains connecting their wrists, and a wave represented by a semi-transparent blue quadrilateral.

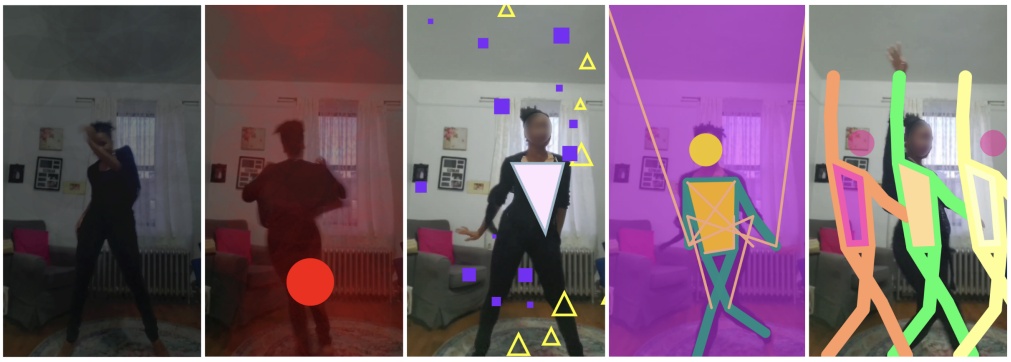


Figure 6: Colorful Escape: From smoke and darkness to dancing clones. (Images use an instructor video with learner code.)

6.3 Colorful Escape

Colorful Escape consisted of five young women, and in contrast to the other three teams, did not identify a current social issue at the start. They used hip-hop and contemporary dance styles to choreograph "A Sweeter Place" by Selena Gomez which describes a desire to escape from pain reflecting a thematic idea from their brainstorming document, "light from darkness." The team's animations were organized by song section rather than specific lyrics and include a dark overlay and nearly transparent shapes to resemble smoke, colorful circles changing in size, animated triangles covering their torsos, abstract silhouettes surrounding their body, and duplicate silhouettes (Figure 6). The team chose not to feature all dancers at all times, but instead feature groups of two and then three dancers in middle portions of the performance.

6.4 Love Yourself

Love Yourself contained five young women who identified with core themes of self-love, perseverance, and living in the moment. They chose to choreograph "Fight Song" by Rachel Platten in a contemporary dance style. The learners integrated five discrete animations: Two represent lyrics literally (waves and explosions), one symbolizes the theme (a heart passed between screens), and two are aesthetically driven featuring traces drawn with `poseHistory` with the team's predefined color palette (Figure 9). The explosion animation (Figure 7) demonstrates an especially clear alignment between music, choreography and animation: As the singer recites, "I might only have one match, but I can make an explosion," the learners forcefully project their hands up and out, while circles appear to explode. A code description is as follows:

- `what:` circle.
- `when:` portion of the song when the lyric occurs.
- `where:` multiple positions, in between the wrists.
- `how:` red and yellow, size determined by wrist distance.

Furthermore, Love Yourself attempted to pass a virtual heart between videos to the lyrics, "Can make a heart open" (Figure 8). One learner sketched out an arrangement of videos for the editor to use and for the heart to travel along. With instructor support, the learners coded the heart to move in and out of view based on the approximate timing of specific lyrics. Yet, they did not see the collage of videos until the final performance, and timing was not adequate to convey the effect they intended.

7 OVERVIEW OF FINDINGS

We draw from observation notes, learner project videos, code, and working documents to identify successes and limitations of the current danceON prototype and deployment with respect to our initial four design goals.

7.1 Personally and Culturally Relevant: Low barrier of entry

danceON enabled two cohorts of novice programmers to code and successfully complete original artistic projects in under two weeks with assistance from the instructor and researchers. Its web implementation and lack of additional hardware requirements enabled remote learning even for a few learners who only had partial computer access. Furthermore, the system's support for any video featuring a single dancer allowed learners to select their own music and dance in any style. We see early evidence from the camp, including the superhero statements (§5) coded independently, that danceON's syntax and reactivity encouraged learners to draw connections between their body and its representation in code. The webcam mode and video overlays (§4.1) were used heavily by the instructor while lecturing and asking questions.

Yet, instructors and researchers provided ample support to learners coding examples in advance to use as starting points and helping them finish their projects to better realize their ideas (§5.3.2). While

Figure 7: Love Yourself: Explosion animation building on relevant lyrical content and choreography.

Figure 8: Love Yourself: Frames captured across time as learners planned to move a heart between their screens, (e.g. entering from the left and exiting out the bottom of the top row) relying on proper arrangement and accurate timing.

the lack of time combined with the large scope of projects necessitated additional support, certain features of danceON contributed to a lack of scaffolding and complexity. The template examples provided by our team only showed basic use of shape primitives but not how to combine primitives or manipulate their properties to simulate phenomena like fire or water, and it wasn't clear whether learners understood how to reference and copy the provided examples (§4.4). Additionally, danceON's syntax is complex – especially its punctuation. During instruction, learners experienced some difficulties with comma placement and matching brackets (§5.2), while projects became especially hard to manage as individuals worked separately on discrete animations or song regions (§5.3.1).

7.2 Personally and Culturally Relevant: Wide room for expression

danceON clearly supports personally and culturally relevant engagement in a range of forms. We saw no shortage of ideas as instructors synthesized ten potential themes worthy of further exploration. While three teams, Dripping With Power, Black Lives Matter, and Love Yourself responded to social issues, including women empowerment, racial justice, and self care respectively, the Colorful Escape team was motivated by aesthetic alone. danceON enabled the groups to explore narrative on their own terms. Dripping with Power featured a soloist and a spoken monologue. Love Yourself attempted to come together despite their physical distance through passing a heart between screens. Colorful Escape used color as a storytelling vehicle, beginning their performance covered in dark grey hues and concluding with brightly colored virtual outfits and clones dancing side by side. The Black Lives Matter Team displayed moments of evocative imagery including becoming bound and later breaking free of virtual chains. As further illustrated below, danceON's environment supported countless animation methods and mappings for learners to concretize their ideas.

7.3 Creative Coding: Art motivates code

There are numerous instances of learners fulfilling artistic ideas in code. Learners programmed animations that represent specific song lyrics (e.g. Black Lives Matter's flames) or captured a wider lyrical theme (e.g. Dripping With Power's storm). They wrote animations to augment single gestures (e.g. Love Yourself's explosions) and entire sections (e.g. Colorful Escape's clones). Learners chose shapes and shape properties to adhere to their team's theme, e.g. the Love Yourself Team requested a heart shape while the Colorful Escape team transitioned from dark grey to bright vivid colors.

7.4 Creative Coding: Code motivates art

In contrast, we observed a small number of instances where a learner's programming process led to new artistic ideas. After electing to animate a virtual storm, the Dripping With Power team chose to use an umbrella as a key prop in their choreography. Furthermore, unlike her teammates in Love Yourself who wrote specific animation ideas tied to lyrics and theme, one learner wrote `poseHistory` under the Animation column of her team's planning document indicating an interest in experimenting with a data structure referenced by the instructor but not given instruction time. Ultimately her animation did not deviate far from the `poseHistory` example included, but she independently copied and remixed it by accessing multiple body parts and changing the colors to match her team's theme (Figure 9). Finally, learners across teams altered the appearance of virtual objects to correspond with their final performance videos, e.g. to match the color of their shirts, but we note that this was often prompted by instructors.

8 DISCUSSION

Our Findings show that danceON's systems are compatible with the first four design goals we listed. In our Discussion, we reflect further on the extent to which danceON achieved those goals and address the implications for designing creative computing environments.

8.1 Connecting the Virtual and Physical World

Within a curriculum built upon reflective ideation in which young women generated themes and ideas to use as the basis of their dance, danceON succeeded at developing strong connections between the computational thought process and the artistic depiction of their ideas. First, our deployment demonstrates how simplicity provides room for artistic freedom and contextual sense-making. danceON

Figure 9: Love Yourself: A learner remixed an example to use both wrists and ankles, and create fading trails in her group's color palette.

enabled learners to create metaphorical and symbolic representations related to the lyrical and thematic content of the songs and their ideas. While available objects and properties were simple (i.e. lines, circles, triangles, etc.), learners composed and repurposed them to create understandable phenomena, such as multiple vertical lines representing "rain"; and yellow and red circle "explosions." These representations were authentic in that they enabled learners' choreography to respond to them in recognizable ways, e.g. integrating props like the umbrella used by Dripping with Power as the animations rained (Figure 5a). Second, danceON enabled learners to create animations that they simply found aesthetically pleasing, rather than metaphorically significant. Crucially, aesthetic ideas held power because they were attached to real, physical body movement. For example, learners created a fading trail of circles tied to each of their wrists and ankles (Figure 9). While circles in isolation may be uninteresting, when matched to dancer movements, they augment choreography connecting data to the physical world, e.g. visualizing movement speed, and interacting with real objects like learner shirts. These physical-to-virtual connections were only made possible because of the underlying design centered on learner-uploaded dance videos. In prior work in which learners programmed avatar dancers, Daily et al. [19] noted learners' desire to close the gap between the avatar and their identity. In danceON, learners work directly with their own videos and their own bodies potentially opening opportunities for the art piece to more easily align with their identity.

8.2 Design for Learning Progress

In our initial deployment of danceON, we observed a clear trajectory of learners working with increasingly long and sophisticated media and data. Learners first explored individual danceON object properties in isolation, then constructed single-gesture, reactive behaviors in their superhero statements, and finally worked on large performances bringing together diverse, sophisticated animations. However, learners struggled to independently realize their complex ideas under time constraints (§5.3.2), and experienced difficulties understanding the textual-language and debugging error messages (§5.2). These difficulties point to specific opportunities to scaffold the learning progression through the design of the language and IDE. danceON, and other creative coding environments, could benefit from implementing adaptive design paradigms such as Ability-Based Design [88], in which an interface is responsive to a user's ability and malleable when that ability changes. Similarly, the language levels approach [41] implements progressive disclosure to programming languages and interpreters/compilers in which the programmatic interface of a system matches the complexity of a user's current understanding. For example, if danceON applied language levels to support understanding through clearer error messages, an early level might prevent variable definitions (or other bindings). A learner who forgets to place quotation marks around the word *circle*, would receive a message hinting that they forgot quotation marks along with reasoning and a suggested fix, e.g. "did you forget quotation marks around 'circle'?" rather than the message "circle is not defined" which requires more knowledge of programming to understand and resolve. This kind of error message would be feasible within the constraints of the language level precisely because it does not allow definitions – circle would be a member of a pre-defined set of names within the program's scope, permitting the use of the clearer error message.

8.2.1 Improvements to the danceON Editor. The editing environment is inextricably linked to the programming language, and we see further design opportunities to support the collaboration and work habits we observed, as well as expand and adapt to changing skill levels. For example, several media-focused novice programming tools, including Scratch [47], JES [35], and EarSketch [30], combine a programming editor with a direct manipulation media viewing/editing environment. Currently, all of danceON's editing is carried out through code. Once learners began their group projects, danceON code morphed from unordered sets of relationships and behaviors as we intended, to long, ordered lists essentially containing timed events (§5.3.1). This became challenging for learners to read, edit, and collaborate remotely on with their peers. A solution could include a timeline below the video, into which learners drag-and-drop animation "scenarios," defined by danceON scripts that become operative within visually selected segments of the timeline. Such a feature, which introduces complexities like multiple scripts and media-based state, may be hidden at first to guide learners toward learning to primarily trigger animations through movement rather than micro-timing.

8.2.2 Future Learning Opportunities. The above design proposals derive from observations of relatively brief learner experiences with danceON, and thus pertain more to the beginning of a learning progression. However, danceON is intended to support complex projects and use cases, and further research is necessary to understand how to promote creativity and learning at later stages. During the camp we observed how learners independently translated English statements into declarative code chunks (Table 2) that were clearer and more succinct than in general creative coding environments (§4.5). Yet, we did not witness learners gaining

Figure 10: Black Lives Matter: Key point locations of the elbows are out of view in the second frame causing the 'rope' animation to disconnect from the body and visibly 'glitch.'

enough mastery over the danceON syntax to treat objects as a low-level grammar with which to realize broader creative goals. Learners, such as one inspired by the flames described in a Beyonce song (Figure 5b), certainly expressed relevant and meaningful goals. Realizing those ideas however was much more complex than the challenges they overcame in which there was always a one-to-one mapping between stated goal and code implementation. Continuation of the design-based research with learners and instructors over a longer period is necessary to understand how to scaffold more advanced problem decomposition in a dance computing context.

8.3 Messy, Glitchy Data Art

While danceON successfully represented and exposed body position data, it could better facilitate interaction with and guide learner understanding of missing and incorrect data. danceON's declarative, reactive programming system is centered around the output of computer vision algorithms run on homemade dance videos. As described in Section 5.3.4, these algorithms are imperfect and occasionally generate data with missing or inaccurate body position information, especially if body parts are occluded (e.g. Figure 10). In the context of this deployment, we and SFD leads viewed inaccurate and missing data unfavorably since they produced apparent "mistakes" threatening the success of final projects.

Future work might investigate how errors and noise in data could offer organic opportunities for discovery, learning, and inspiration. In promoting data empowerment for non-technical learners, "[c]ultivating skepticism of 'raw' data" should be seen as an important goal of any data literacy program [21]. Focusing attention on errors in data can lead to better understanding of the limitations of data collection systems. System constraints can in turn lead to thoughtful workarounds or, in the current context, may be harnessed for creative effect. For example, some game designers embraced technical challenges in the "seamful design" of early location-based mobile games [12].

A structured inquiry process into data imperfections might invite the learner to seek out the boundaries of motion capture technology and to differentiate between systematic and random errors in data (G5: Data literacy). Guiding questions include: Under what conditions do body keypoints disappear? What are the limitations of a two-dimensional (depth-less) system when the dancer's arms are pointing towards the camera? How still is standing still, when tiny fluctuations in alignment (or lighting) can manifest as visible jitter in the location of a dancer's knee? How fast is too fast for the computer to track a hand? More broadly, to what extent is the computer's "perception" different from a human observer's?

danceON's design may also be further extended to support user interactions with incorrect data. For example, an additional layer to the Skeleton overlay (§4.3) could allow users to manually drag body data coordinates that appear incorrect and add body part data that is missing. Or, an additional tab may allow users to toggle rules/heuristics, (e.g. if a left or right body part is missing, reflect the position of its known partner over the center of the body, if a body part is missing use its last known position, etc.), and observe whether such choices improve results and/or yield unintended consequences. A feature for advanced use could provide further algorithmic control over missing data based on hypothesized causes in a particular video. Still, accurate data is a lesser priority than scaffolding an authentic learner experience of data wrangling.

9 CONCLUSION

In this paper, we presented the first iteration of danceON, a system for integrating dance and animation in culturally relevant, creative computational learning environments. Our ongoing design-based research investigation with STEM From Dance (SFD) led us to create a tool to engage young women of color in computing and provide novices ease of entry into explorations at the boundaries of data and dance. The four learner projects we describe highlight the affordances of danceON: its support for a range of opportunities for personal expression and its potential for creating mutually informing relationships between computation, data, and dance. We identified opportunities to embed scaffolding within danceON's design to account for learners' growing expertise. Additionally, we addressed the benefit of real-world, messy data to critically engage learners working at the seams of technology. Our work sets a path for future developments to explore the connections between dance and data-driven systems and to push the bounds of how learners can engage in meaningful embodied learning experiences.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support for this project by the National Science Foundation (STEM+C 1933961). We thank the administrators and instructors who guided us and worked tirelessly to provide a supportive and rewarding summer learning environment– Yamilée Toussaint Beach, Elena Hartoonian, Stacie Cannon, and Shannon Peng. Finally, we thank Shriram Krishnamurthi whose feedback and insights inspired portions of our Discussion.

REFERENCES

[1] Samuel Aaron, Alan F. Blackwell, and Pamela Burnard. 2016. The development of Sonic Pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music, Technology and Education* 9, 1 (5 2016), 75–94. https://doi.org/10.1386/jmte.9.1.75_1

[2] Brittany Aronson and Judson Laughter. 2016. The Theory and Practice of Culturally Relevant Education: A Synthesis of Research Across Content Areas. *Review of Educational Research* 86, 1 (2016), 163–206. https://doi.org/10.3102/0034654315582066

[3] Computer Science Teachers Association. 2017. CSTA K-12 Computer Science Standards, Revised 2017. http://www.csteachers.org/standards

[4] Mary M Atwater, Melody Russell, and Malcolm B Butler. 2013. *Multicultural science education: Preparing teachers for equity and social justice.* Springer Science & Business Media.

[5] Engineer Bainomugisha, Andoni Lombide Carreton, Tom van Cutsem, Stijn Mostinckx, and Wolfgang de Meuter. 2013. A survey on reactive programming. *ACM Computing Surveys (CSUR)* 45, 4 (2013), 1–34.

[6] Arthur Bakker. 2018. *Design research in education: A practical guide for early career researchers.* Routledge.

[7] Jeanne Shapiro Bamberger and Armando Hernandez. 2000. *Developing musical intuitions: A project-based introduction to making and understanding music.* Oxford University Press, USA.

[8] Yoav Bergner, Shiri Mund, Ofer Chen, and Willie Payne. 2020. Leveraging interest-driven embodied practices to build quantitative literacies: A case study using motion and audio capture from dance. *Educational Technology Research and Development* (2020), 1–24.

[9] Rahul Bhargava, Ricardo Kadouaki, Emily Bhargava, Guilherme Castro, and Catherine D'Ignazio. 2016. Data murals: Using the arts to build data literacy. *The Journal of Community Informatics* 12, 3 (2016).

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (1 2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[11] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, Vol. 1. 25.

[12] Gregor Broll and Steve Benford. 2005. Seamful design for location-based mobile games. In *International Conference on Entertainment Computing*. Springer, 155–166.

[13] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. (2018). arXiv:1812.08008

[14] Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3334480.3382839

[15] Dionne N Champion. 2018. *The STEAM dance makerspace: A context for integration: An investigation of learning at the intersections of STEM, art, making and embodiment.* Ph.D. Dissertation. Northwestern University.

[16] John Clements, Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2004. Fostering Little Languages. *Dr. Dobb's Journal* 29, 3 (2004), 16–24.

[17] Kathryn Cunningham. 2020. Purpose-first Programming: A Programming Learning Approach for Learners who Care Most About What Code Achieves. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 348–349.

[18] Shaundra B Daily, Alison E Leonard, Sophie Jörg, Sabarish Babu, and Kara Gundersen. 2014. Dancing alice: Exploring embodied pedagogical strategies for learning computational thinking. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 91–96.

[19] Shaundra B Daily, Alison E Leonard, Sophie Jörg, Sabarish Babu, Kara Gundersen, and Dhaval Parmar. 2015. Embodying computational thinking: Initial design of an emerging technological learning tool. *Technology, Knowledge and Learning* 20, 1 (2015), 79–84.

[20] Kayla DesPortes, Monet Spells, and Betsy DiSalvo. 2016. The MoveLab: Developing Congruence Between Students' Self-Concepts and Computing. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16* (2016), 267–272. https://doi.org/10.1145/2839509.2844586

[21] Catherine D'Ignazio. 2017. Creative data literacy. *Information Design Journal* 23, 1 (7 2017), 6–18. https://doi.org/10.1075/idj.23.1.03dig

[22] Catherine D'Ignazio and Lauren F Klein. 2020. *Data feminism.* MIT Press.

[23] Betsy DiSalvo. 2014. Graphical qualities of educational technology: Using drag-and-drop and text-based programs for introductory computer science. *IEEE computer graphics and applications* 34, 6 (2014), 12–15.

[24] Benedict du Boulay, Tim O'Shea, and John Monk. 1981. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies* 14, 3 (1981), 237–249.

[25] Alireza Ebrahimi. 1994. Novice programmer errors: Language constructs and plan composition. *International Journal of Human-Computer Studies* 41, 4 (1994), 457–480.

[26] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2009. A functional I/O system or, fun for freshman kids. *ACM Sigplan Notices* 44, 9 (2009), 47–58.

[27] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. 2018. *How to design programs: an introduction to programming and computing.* MIT Press.

[28] Lila Finch, Celeste Moreno, and R Benjamin Shapiro. 2020. Teacher and student enactments of a transdisciplinary art-science-computing unit. *Instructional Science* (2020), 1–44.

[29] Kathi Fisler. 2014. The recurring rainfall problem. In *Proceedings of the tenth annual conference on International computing education research*. 35–42.

[30] Jason Freeman, Brian Magerko, Tom McKlin, Mike Reilly, Justin Permar, Cameron Summers, and Eric Fruchter. 2014. Engaging underrepresented groups in high school introductory computing through computational remixing with EarSketch. In *Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14*. ACM Press, New York, New York, USA, 85–90. https://doi.org/10.1145/2538862.2538906

[31] Geneva Gay. 2002. Preparing for culturally responsive teaching. *Journal of teacher education* 53, 2 (2002), 106–116.

[32] Berto Gonzalez, Erin Cherry, and Celine Latulipe. 2012. Dance-inspired technology, Technology-inspired dance. *NordiCHI 2012: Making Sense Through Design - Proceedings of the 7th Nordic Conference on Human-Computer Interaction*. https://doi.org/10.1145/2399016.2399078

[33] Norma González, Rosi Andrade, Marta Civil, and Luis Moll. 2004. Bridging funds of distributed knowledge: Creating zones of practices in mathematics. 6, 1-2 (2004), 115–132. https://doi.org/10.1207/s15327671espr0601-2_7

[34] Kris D Gutiérrez and Barbara Rogoff. 2003. Cultural ways of learning: Individual traits or repertoires of practice. *Educational researcher* 32, 5 (2003), 19–25.

[35] Mark Guzdial. 2003. A media computation course for non-majors. In *Proceedings of the 8th annual conference on Innovation and technology in computer science education*. 104–108.

[36] Brian Harvey and Matthew Wright. 1999. *Simply Scheme: introducing computer science.* Mit Press.

[37] Katrina Hazzard-Gordon. 1985. African-American vernacular dance: core culture and meaning operatives. *Journal of Black Studies* 15, 4 (1985), 427–445.

[38] Christopher M Hoadley. 2004. Methodological alignment in design-based research. *Educational psychologist* 39, 4 (2004), 203–212.

[39] Robert Holwerda and Felienne Hermans. 2018. A usability analysis of blocks-based programming editors using cognitive dimensions. In *2018 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE, 217–225.

[40] Liwen Huang and Paul Hudak. 2003. Dance: A declarative language for the control of humanoid robots. *Yale, Department of Computer Science, Yale University New Haven, CT 06520, Tech. Rep.* (2003).

[41] Shriram Krishnamurthi and Kathi Fisler. 2019. Programming paradigms and beyond. *The Cambridge Handbook of Computing Education Research* 37 (2019).

[42] Gloria Ladson-Billings. 2014. Culturally relevant pedagogy 2.0: aka the remix. *Harvard Educational Review* 84, 1 (2014), 74–84.

[43] Alison E Leonard and Shaundra B Daily. 2014. The Dancing Alice Project: Computational and embodied arts research in middle school education. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*.

[44] Alison E Leonard, Shaundra B Daily, Sophie Jörg, and Sabarish V Babu. 2020. Coding moves: Design and research of teaching computational thinking through dance choreography and virtual interactions. *Journal of Research on Technology in Education* (2020), 1–19.

[45] Alison E Leonard, Nikeetha Dsouza, Sabarish V Babu, Shaundra B Daily, Sophie Jörg, Cynthia Waddell, Dhaval Parmar, Kara Gundersen, Jordan Gestring, and Kevin Boggs. 2015. Embodying and programming a constellation of multimodal literacy practices: Computational thinking, creative movement, biology, & virtual environment interactions. *Journal of Language and Literacy Education* 11, 2 (2015), 64–93.

[46] Lindsay Lindberg, Deborah Ann Fields, and Yasmin B Kafai. 2020. STEAM maker education: Conceal/reveal of personal, artistic and computational dimensions in high school student projects. In *Frontiers in Education*, Vol. 5. Frontiers Research Foundation, 1.

[47] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 1–15.

[48] Bill Manaris, Blake Stevens, and Andrew R. Brown. 2016. JythonMusic: An environment for teaching algorithmic music composition, dynamic coding and musical performativity. *Journal of Music, Technology and Education* 9, 1 (5 2016), 33–56. https://doi.org/10.1386/jmte.9.1.33_1

[49] Lauren McCarthy. 2020. p5.js. https://p5js.org

[50] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2011. Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. 168–172.

[51] Leo A. Meyerovich, Arjun Guha, Jacob Baskin, Gregory H. Cooper, Michael Greenberg, Aleks Bromfield, and Shriram Krishnamurthi. 2009. Flapjax: A programming language for Ajax applications. *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA* (2009), 1–20. https://doi.org/10.1145/1640089.1640091

[52] Sten Minör. 1992. Interacting with structure-oriented editors. *International Journal of Man-Machine Studies* 37, 4 (1992), 399–418.

[53] Jessica Morales-Chicas, Mauricio Castillo, Ireri Bernal, Paloma Ramos, and Bianca L Guzman. 2019. Computing with relevance and purpose: A review of culturally relevant education in computing. *International Journal of Multicultural Education* 21, 1 (2019), 125–155.

[54] Marco T Morazán. 2018. Infusing an HtDP-based CS1 with distributed programming using functional video games. *J. Funct. Program.* 28 (2018), e5.

[55] Bahare Naimipour, Mark Guzdial, and Tamara Shreiner. 2019. Helping social studies teachers to design learning experiences around data: Participatory design for new teacher-centric programming languages. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 313–313.

[56] Bahare Naimipour, Mark Guzdial, and Tamara Shreiner. 2020. Engaging pre-service teachers in front-end design: Developing technology for a social studies classroom. In *Proceedings of Frontiers in Education (FIE) 2020*. IEEE.

[57] Enrico Nardelli. 2019. Do we really need computational thinking? *Commun. ACM* 62, 2 (2019), 32–35.

[58] Chris Nash and Alan Blackwell. 2012. Liveness and flow in notation use. *NIME 2012 Proceedings of the International Conference on New Interfaces for Musical Expression* (2012), 76–81.

[59] National Science Foundation, National Center for Science and Engineering Statistics. 2019. *Degrees awarded to women: Mathematics and statistics, 1997, 2006, 2016.* https://ncses.nsf.gov/pubs/nsf19304/digest/field-of-degree-women#mathematics-and-statistics

[60] Tor Ole B Odden and Rosemary S Russ. 2019. Defining sensemaking: Bringing clarity to a fragmented theoretical construct. *Science Education* 103, 1 (2019), 187–205.

[61] Ihudiya Finda Ogbonnaya-Ogburu, Angela D.R. Smith, Alexandra To, and Kentaro Toyama. 2020. Critical Race Theory for HCI. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–16. https://doi.org/10.1145/3313831.3376392

[62] Cathy O'neil. 2016. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books.

[63] John F Pane, Chotirat Ratanamahatana, and Brad A a Myers. 2001. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies* 54, 2 (2001), 237–264.

[64] Django Paris. 2012. Culturally sustaining pedagogy: A needed change in stance, terminology, and practice. *Educational Researcher* 41, 3 (2012), 93–97. https://doi.org/10.3102/0013189X12441244

[65] D. Parmar, J. Isaac, S. V. Babu, N. D'Souza, A. E. Leonard, S. Jörg, K. Gundersen, and S. B. Daily. 2016. Programming moves: Design and evaluation of applying embodied interaction in virtual environments to enhance computational thinking in middle school students. In *2016 IEEE Virtual Reality (VR)*. 131–140.

[66] Roy D Pea. 1986. Language-independent conceptual "bugs" in novice programming. *Journal of educational computing research* 2, 1 (1986), 25–36.

[67] John Peterson, Gregory D Hager, and Paul Hudak. 1999. A language for declarative robotic programming. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, Vol. 2. IEEE, 1144–1151.

[68] Nichole Pinkard, Caitlin Kennedy Martin, and Sheena Erete. 2020. Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community. *Interactive Learning Environments* 28, 3 (2020), 347–361.

[69] Joe Politz, Benjamin Lerner, Sorawee Porncharoenwase, and Shriram Krishnamurthi. 2019. Event loops as first-class values: A case study in pedagogic language design. *The Art, Science, and Engineering of Programming* (2019). https://doi.org/10.22152/programming-journal.org/2019/3/11

[70] Justin Pombrio, Shriram Krishnamurthi, and Kathi Fisler. 2017. Teaching programming languages by experimental and adversarial thinking. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[71] Thomas W Price and Tiffany Barnes. 2015. Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the eleventh annual international conference on international computing education research*. 91–99.

[72] Charlie Roberts, Jesse Allison, Daniel Holmes, Benjamin Taylor, Matthew Wright, and JoAnn Kuchera-Morin. 2016. Educational design of live coding environments for the browser. *Journal of Music, Technology and Education* 9, 1 (5 2016), 95–116. https://doi.org/10.1386/jmte.9.1.95_1

[73] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization & Computer Graphics (Proc. InfoVis)* (2017). https://doi.org/10.1109/tvcg.2016.2599030

[74] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. 2015. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 659–668.

[75] Emmanuel Schanzer, Shriram Krishnamurthi, and Kathi Fisler. 2018. Creativity, customization, and ownership: Game design in Bootstrap: Algebra. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 161–166.

[76] Kimberly A Scott, Kimberly M Sheridan, and Kevin Clark. 2015. Culturally responsive computing: A theory revisited. *Learning, Media and Technology* 40, 4 (2015), 412–436.

[77] Kimberly A Scott and Mary Aleta White. 2013. COMPUGIRLSâ€™standpoint: Culturally responsive computing and its effect on girls of color. *Urban Education* 48, 5 (2013), 657–681.

[78] Kristin A Searle and Yasmin B Kafai. 2015. Culturally responsive making with American Indian girls: Bridging the identity gap in crafting and computing with electronic textiles. In *Proceedings of the Third Conference on GenderIT*. 9–16.

[79] Otto Seppälä, Petri Ihantola, Essi Isohanni, Juha Sorva, and Arto Vihavainen. 2015. Do we know how difficult the rainfall problem is?. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*. 87–96.

[80] R Benjamin Shapiro and Matthew Ahrens. 2016. Beyond blocks: Syntax and semantics. *Commun. ACM* 59, 5 (2016), 39–41.

[81] Elliot Soloway. 1986. Learning to program= learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[82] Tensorflow. 2020. Pose Detection in the Browser: PoseNet Model.

[83] The Design-Based Research Collective. 2003. Design-Based Research: An Emerging Paradigm for Educational Inquiry. *Educational Researcher* 32, 1 (1 2003), 5–8. https://doi.org/10.3102/0013189X032001005

[84] David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children*. 199–208.

[85] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1–25.

[86] David Weintrop and Uri Wilensky. 2017. How block-based languages support novices. *Journal of Visual Languages and Sentient Systems* 3 (2017), 92–100.

[87] Jeannette M Wing. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1881 (2008), 3717–3725.

[88] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-based design: Concept, principles and examples. *ACM Trans. Access. Comput.* 3, 3, Article 9 (April 2011), 27 pages. https://doi.org/10.1145/1952383.1952384