Understanding Version Control as Material Interaction with *Quickpose*

Eric Rawn
erawn@berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Jingyi Li
jingyili@cs.stanford.edu
Stanford University
Stanford, California, USA

Eric Paulos
paulos@berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Sarah E. Chasins
schasins@berkeley.edu
University of California, Berkeley
Berkeley, California, USA

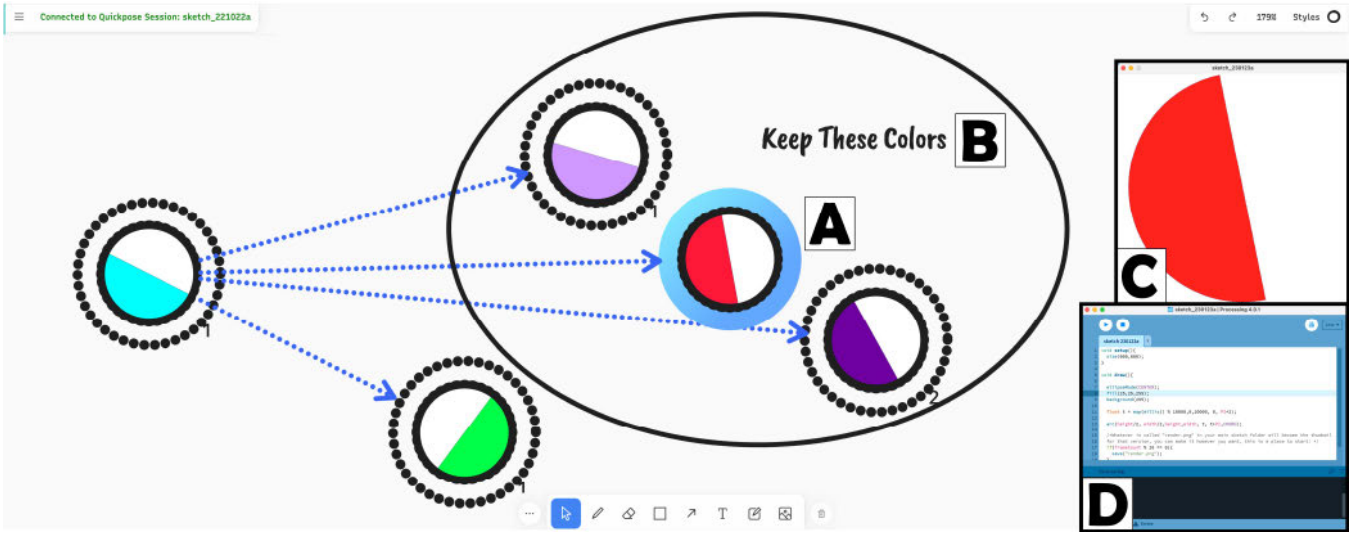

Figure 1: Quickpose is a version control system for creative coding designed to support what we term *material interaction*: how practitioners engage their materials. A. Quickpose represents versions of a program as circular thumbnails on an interactive canvas (current version represented with Blue border). B. In this sample canvas, a user placed a textbox with "Keep These Colors" to annotate the multiple versions nearby. Annotation was one salient aspect of material interaction which we aimed to support with Quickpose. C. The Quickpose render output shows the current version rendered as an animated image. D. The Processing IDE shows the code corresponding to the current version. Selecting a new version on the canvas updates the code to match.

ABSTRACT

Whether a programmer with code or a potter with clay, practitioners engage in an ongoing process of working and reasoning with materials. Existing discussions in HCI have provided rich accounts of these practices and processes, which we synthesize into three themes: (1) reciprocal discovery of goals and materials, (2) local knowledge of materials, and (3) annotation for holistic interpretation. We then apply these design principles *generatively* to the domain of version control to present Quickpose: a version control system for creative coding. In an in-situ, longitudinal study of Quickpose guided by our themes, we collected usage data, version history, and interviews. Our study explored our participants' material interaction behaviors and the initial promise of our proposed measures for recognizing these behaviors. Quickpose is an exploration of *version control as material interaction*, using existing discussions to inform domain-specific concepts, measures, and designs for version control systems.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581394

CCS CONCEPTS

• **Software and its engineering** → **Software configuration management and version control systems**; • **Human-centered**

computing → *Human computer interaction (HCI)*; • **Applied computing** → *Media arts*.

KEYWORDS

End-User Programming, Version Control Systems (VCS), Materiality, Variations, Creative Coding

ACM Reference Format:
Eric Rawn, Jingyi Li, Eric Paulos, and Sarah E. Chasins. 2023. Understanding Version Control as Material Interaction with *Quickpose*. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3544548.3581394

1 INTRODUCTION

Whether practitioners use clay (for ceramicists), mathematical notation (for mathematicians), or code (for programmers), they interact with parts of the world to understand, explore, and create. They take these parts of the world up *as materials*. "Material" in this sense does not describe a certain class of things (like lumps of clay or pieces of source code), but instead describes an ongoing *relationship* between a practitioner and what they work with. For example, the same knife can be activated *as a material* for a blacksmith but become an invisible, unnoticed tool when used by a chef. More nuanced situations might arise with programming; for example, where a programmer may shift between taking up the code, the compiler, or the hardware as a material as their attention and the task ahead shifts. HCI researchers have long studied what it means to engage something *as a material*, highlighting its educational [61], cognitive [43], or creative [75] dimensions. While this relationship between a practitioner and material has taken many names, this paper will use the term *material interaction* to reference this group of ideas.

In this work, we synthesize these rich accounts of material interaction to employ them *generatively* [2]: to inspire domain-specific concepts and novel designs which are built upon broader theoretical accounts. Therefore, we first distill and operationalize this broader HCI discussion of how a practitioner engages their materials into three themes of material interaction: (i) that goals develop alongside the engagement of materials (Section 2.1), (ii) that practitioners build knowledge about materials through *exploring and reasoning* with the materials themselves (Section 2.2), and (iii) that practitioners contextualize, reflect, and organize knowledge about materials through *annotative* practices (Section 2.3). Using these preexisting themes, we propose design principles for material interaction in interfaces (Section 2.4).

We then apply these principles to a specific domain to explore the potential of these themes to suggest novel design insights and measures [2]. A strong thread in these themes of material interaction is in how practitioners move between, compare, or reflect on versions of their work. This made history management practices [81]—the tools, habits, notation systems, or organizational practices used to record, recall, and manage the history of project—an especially interesting site to investigate material interaction in interfaces. For software engineering, version control systems are often the tools that manage this history for programmers, and are a primary way programmers interact with prior versions of their code. Therefore, we take up version control as a domain within which our themes could be *critical* [2] (they could highlight places existing tools failed to accommodate material interaction) and *constructive* (they could suggest novel design principles, measures, and implementations). Guided by these themes, we built Quickpose, a version control system for creative coding. Although material interaction encompasses more than just creativity, artists who code are a group of practitioners who are often reflective and experimental about their process [48], making them a well-suited group to study initially. We discuss the design of Quickpose, including how the themes motivated its design, in detail in Section 4.

Through our in-situ study of Quickpose with expert Processing artists, we used Quickpose as a platform to explore *version control as a material interaction*, connecting the existing conversation on material interaction in HCI to inspire practice-oriented insights for interface design in version control systems. Using our design principles, we proposed concrete measures of material interaction for the domain of version control systems. These measures guided our quantitative analysis of collected data from Quickpose's usage in addition to how we analyzed the qualitative interviews with participants. Rather than validate the themes presented or evaluate Quickpose on a specified task, our study explores (1) how users engage in material interaction behaviors if their tools offer the functionality for doing so, and (2) whether the measures suggested by the collected themes present initial promise for recognizing such practices. Additionally, we used our study to iteratively refine and contextualize the themes and measures we discuss [2, 9] in order to support future development.

We found evidence for each of our three themes through semi-structured interviews with participants and analysis of their usage data, Quickpose canvases, and code versions. For example, navigation history between disparate versions indicated how practitioners broadly utilized their version history even in cases where their forking history showed a more linear behavior. Using the study to reflexively develop our themes and principles, we then propose a refined set of measures for further research. Finally, we discuss how our work might contribute to a broader theory of material interaction for interface design, showing how Quickpose served as a *platform* for investigating the themes and also how they might help explain and reason about previous findings. We present Quickpose as a generative exploration: using a conceptual lens of material interaction to suggest new ways of building version control systems, and then using a system built with those guidelines to investigate material interaction behaviors in practice.

We present the following contributions:

(1) A set of metrics that operationalize the existing conversation around material interaction. These metrics center our three design principles of continual goal reformation, contextual exploration, and holistic, linked annotation.
(2) Quickpose, a version control tool for creative coding, which supports and measures material interaction in line with our themes.
(3) A quantitative and qualitative study of Quickpose users to understand of material interaction behaviors in Quickpose, with implications for future researchers studying material interaction in version control systems.

2 THREE THEMES OF MATERIAL INTERACTION

The HCI Community has long been interested in how people take up, reason with, and subsequently transform the material reality around them. This is especially true in the case of the practitioner or artisan, someone who directly manipulates a material as the core part of their work. Whether the material is code or clay, HCI researchers have drawn from fields like anthropology [19, 38, 76], design studies [16, 75], or philosophy [55] to better understand how practitioners engage their materials. In this section we present three aspects of this discussion of how HCI researchers have discussed material interaction. In particular, we argue HCI researchers have claimed that (i) goals develop in tandem with material engagement (Section 2.1), (ii) practitioners build knowledge about materials through *exploring and reasoning* with the materials themselves (Section 2.2), and (iii) practitioners contextualize, reflect, and organize knowledge about materials through *annotative* practices (Section 2.3). Finally, we distill these themes into design principles for interfaces (Section 2.4), which structure our construction of Quickpose both as a tool for facilitating material interaction and measuring it through the tool itself.

2.1 Reciprocal Discovery of Goals and Materials

In response to working with a material, practitioners' goals change [17, 87]. Whether it is the desire to express an idea, a design brief to be fulfilled, or a curiosity to be investigated, practitioners approach materials with a *goal*, an impulse or motivation which spurs the interaction.

For example, a programmer is building a user interface (UI) for an application. In the course of making the UI, the programmer realizes that all of the buttons planned for the UI make it look cluttered and difficult to understand. The programmer refines the goal in this moment: the goal is now to make a UI where the user can access all of the functionality, but not necessarily via buttons. The programmer continues to iterate, hiding some buttons behind menus and testing the UI until they find the right balance between visual clarity and ease of access—goals that were latent but have now become explicit priorities. As painter James Elkins writes, "the work and its maker exchange ideas and change one another" [23, p. 78].

The example above illustrates how practitioners engage in a *conversational* [38] relationship with a material. Practitioners work out [43, 44], reframe [25, 69], and learn about [61] their goals as they work with materials—the material "talks back" [75, p. 135].

Working with materials is a process of discovery and exploration in two ways: first, because it reveals properties about the materials at hand; second, because those materials change the goal in surprising ways—working with materials "remakes the idea" [23]. Additionally, working with materials may also cause us to reframe our goals or unsettle the entire frame by which the goal made sense. For example, our earlier example programmer might discover in the course of working out the UI that instead of hiding greater functionality behind menus, the total functionality of the interface should have been reduced. On the other hand, perhaps they realize the UI would be more effective in a different paradigm, such as context menus, at which point the goal would become delivering the right functionality *in the right moment* rather than with ease of access. In either case, the programmer learns something about the material and their goals which cause the goals to change.

Theme 1: Goals and materials are reciprocally discovered. Working with materials is a process of continual articulation instead of dictation.

2.2 Local Knowledge of Materials

In the same way that goals are formulated through working with a material, practitioners build knowledge about materials locally through exploration and comparison in context [25, 40, 43, 56, 61]. For example, a graphic designer could explore ten different options for the color on a webpage before settling on one option. Treated as a material by the designer, the other nine are not 'failed' options but were what provided the context for the designer's choice: the variations not only gave the designer insight about the webpage and color scheme, but they also provided the axes along which a choice could be made at all (for example, a choice of saturation, or hue). The *material knowledge* of "what color fits best in the webpage" became accessible to the designer through variation of the material itself (the webpage), and remained largely intuitive or tacit (the designer might say the color "feels right" or "works well with the other colors"). Design theorist Donald Schön would describe the web developer's exploration as "knowing-in action, revealed in and by actual designing" [74, p. 131].

This kind of intuitive or tacit knowledge is opposed to abstract or systematic knowledge, which can be utilized independent of context [32]. Exploratory behaviors, like the web developer's here, are dependent on their context because each variation or experiment is only meaningful for material knowledge in the context of surrounding variations [47, 58].

Rather than two separate categories then, material knowledge and systematic knowledge can be seen as two ends of a spectrum, where practitioners utilize many different kinds at once and can, with effort, systematize previously tacit knowledge. For example, Moradi et al. [58] describe how ceramicists work with glazes (a coating for a ceramic piece) through free-form tacit exploration, incremental trial and error, unstructured annotation of pieces, and rigorous analysis and variation of a glaze recipe. At each step, the ceramicist moves away from engaging the clay as a material and towards a systematic, more scientific [64] knowledge of it.

Two meaningful dimensions to describe this exploration are *degree* (how many variations) and *depth* (how effortful or multiply-iterative are the variations) [16, p. 88]. For example, a ceramicist might tweak the composition of a ceramic glaze across ten otherwise identical pieces and compare them [58], which would comprise a low depth but high degree exploration. On the other hand, they might develop two separate glazes, making changes on each iteration and comparing after ten refinements of each glaze. This could be described as high depth but low degree exploration. This language helps contextualize both practices as different expressions of a singular underlying phenomena.

Theme 2: Knowledge about materials is built locally through exploration and comparison. Knowledge about materials begins as tacit, intuitive, or embodied and must be transformed

if it is to become systematic knowledge. Exploration can vary in depth and degree.

2.3 Annotation for Holistic Interpretation

In the course of working with a material, practitioners' *knowledge practices*–the activities which surround building and cultivating knowledge [31]—focus on contextualizing the work rather than replacing or generalizing it [35, 51, 87]. For example, the ceramicist who tries ten different variations of a glaze might record the particular ratio on paper attached to each piece. Seeing the ten variations together allows for comparison, as we discussed earlier, but the notes attached to each pot help the ceramicist understand how the variations in the glaze result in different appearances. In other words, the practitioner used annotation to aid their understanding of an individual *state* of the material exploration (a single pot) and also the connections and relationships between the states. Designer Es Devlin phrases it this way: "...leave traces of your train of thought ... otherwise [it's difficult] to ... remember what the joints and the junctions were between one thought and the next, and to me that's the really interesting part, find the common denominators, find the underlying patterns"[24].

Additionally, the annotations are meaningful specifically when linked to the material states they describe; for example, appending a paper note to a pot, handwriting annotation over a paper essay draft, or typing notes in a textbox on an file in an image editor. These annotations help the practitioner better understand the material state at hand and would lose their utility if separated from the state.

While not all practitioners use formal annotation systems, knowledge practices around material interaction seem to be *annotative*, even with mental notes like a dancer's practice of *marking* [43]. By building the context for reflection, annotation also supports the two previous themes, furthering goal development and reframing alongside knowledge building through comparison.

Theme 3: Knowledge practices in the course of material interaction are focused on annotation rather than generalization; annotation aids interpretation and analysis of both the individual states of a material exploration and the relationships between states, but does not replace them.

2.4 Material Interaction: Principles for Design

These three themes—reciprocal discovery, local knowledge of materials, and holistic annotation—are not meant to be comprehensive or conclusive accounts of what a material interaction entails. Rather, they are an attempt to synthesize insights from across the HCI literature into three practical claims which can support further work, including the remainder of this paper. In this way, we use existing accounts of material interaction not as a general theory of material interaction, but to collect a set of motivating, actionable concepts which structure and guide our research question and study.

While we do not claim these themes of material interaction to be complete, these themes are nonetheless tightly woven together: how we engage materials and act upon them with our intentions, how we build knowledge about them, and how we transform, record, and reflect on that knowledge are three ways of understanding a single phenomena, not three distinct practices summed together.

With this in mind, we extend our themes of material interaction into design principles:

(1) **Continual Goal Reformation** If goals are discovered alongside materials, *interfaces should support the reformation, bifurcation, and demarcation of goals* throughout the material interaction, allowing practitioners to follow their exploration in however many directions it travels.
(2) **Contextual Exploration** If knowledge of materials is built tacitly, *interfaces should support exploration and comparison in the context of other states.* Because exploration can be both low depth, high degree and high depth, low degree, interfaces should allow practitioners to flexibly compare and explore across depth and degree.
(3) **Holistic, Linked Annotation** If knowledge practices with materials do not replace the state they describe but rather aid in its interpretation and the reflection of the entire material process, *interfaces should support flexible annotation which accompanies and directly links to states and groups of states.*

3 RELATED WORK

In our usage, "material" describes a relationship between a thing and a practitioner. A material is thus defined by its context of use by a practitioner. As Sterman et al. [81] discuss in their work on creative strategies, practitioners' engagement with materials over time often manifests through reasoning, working, and reflecting between multiple *versions* or *states*, whether wood (different iterations of a project or piece), code (versions of a program), or dance (iterations of a movement or sections of choreography). They further define *version control systems* as a specific subset of history management tools which organize iterative changes to artifacts themselves. We use these definitions throughout our related work to clarify how existing tools, prior studies, and theoretical contributions have discussed material interaction, history management, and version control.

3.1 Theories of Materiality and Cognition

Theories of materiality have great diversity, but on the whole they often center the agency, influence, or impact of the world outside of a person's thoughts or perceptions [3]. Many of these theories in HCI have focused on *digital* materials – that just as potters mould clay (and the properties of the clay dictates what can and cannot be made), so do interface designers with the digital materials of computer interaction. Wiberg [91] and Jung and Stolterman [40] emphasize how many interface designers already take up computer interaction *as a material*, which is a central premise to Quickpose: that, by extension, creative coders treat (or could possibly treat) Processing code (and the output and interaction it generates) *as a material*. Wiberg clarifies how digital materials manifest in similar ways to physical ones, analyzing their properties, textures, and holistic compositions [90]. Jung and Stolterman similarly address computer interaction as a material, analyzing how the properties and cultural meanings which attend computer interactions manifest in the world of materials and culture (what they term "material ecology") [40]. Where these authors argue for interaction *as a material*, this paper investigates what it means to take something up as a material at all: how do we *interact* with materials?

Other theories emphasize the importance of *tools* (simply put, objects we interact with for practical ends) in our ability to perceive, understand, reflect, and act. Centering the thinking of John Dewey, Dalsgaard presents a pragmatist conception of tools and design practice [17], discussing the importance of tools for how we generate ideas, clarify situations, reason about ideas, and synthesize design perspectives together. Other theorists echo this claim that cognition is a reflexive, practical interplay between mind and world, such as Malafouris' Material Engagement Theory [53], or Alshanetsky's conception of "articulation" [1]. Chalmers and Clarke present an "active external[ist]" view of cognition, highlighting how objects external to the human body or brain play a crucial role in human thinking and acting [14]. Hutchins [36] also supports these sentiments with a theory of *distributed cognition*. This paper certainly relies on the claim that artifacts can play crucial roles in our cognition: we could describe material interaction as not thinking *about* materials, but thinking *through* materials. This sentiment is foundational in Section 2.2. In this paper we build from these theories of how people think and act to try to propose themes for *interface design* – the theories discussed above concern the nature of the mind, cognition, and practical action, but do not provide immediate guidance for understanding how interfaces can support an engagement with materials. For example, if external objects can aid (or are fundamental to) reflection of a design process (which these authors would claim), what does reflection mean in the context of computer interfaces? How can we recognize it and explicitly support it in tool design? These are questions we hope to work towards in this paper.

In a related vein, other scholars discuss *externalization* [20, 74] – the manifestation of an idea outside of a mind, in the form of diagrams, models, notes, or artifacts, as a critical design practice which enables practitioners to reflect on, reason about, and share their design ideas. These theories are foundational for the design principles we propose for interfaces. For example, Dix et al. [20] discuss the importance of summarization, annotation, and tracing of a design process for meaningful reflection. Gedenryd [27] points out the importance of sketches and "low-fidelity" prototypes, which they argue ignores the valueable cognitive function of such prototypes in *"working out"* design ideas. In this paper, we rely on this work to investigate how we can explicitly support behaviors like annotation and sketching in version control systems, in addition to how we can recognize such behaviors in practice.

3.2 Material Interaction but not Version Control

3.2.1 Understanding and Supporting the Design Process. Design ideation and creative practice are important domains where practitioners engage materials. Previous research in these areas offers critical touchstones which we rely on for our discussion of material interaction. Prior empirical studies offer evidence that broad exploration [21], variation [86], iteration [11], and reflection [13] are valuable parts of a design process. These studies show that these activities help people achieve better design outcomes, but they do not propose explanations of *why* these activities occur when practitioners engage materials, nor how to reason about these activities in interfaces. For example, in Section 2.1 we argue that iteration, in part, occurs during a material interaction because practitioners' goals change in response to working with a material. We then propose that supporting iteration within material interaction means supporting the complete reframing, continual refining, and the splitting and fracturing of goals. This prior empirical work thus lays the foundation for our design principles and measures.

In addition to these empirical studies, researchers also recommend design principles for supporting creativity with interfaces. While Quickpose aligns with these guidelines, such as exploration [67], history-keeping [80], and margin-keeping [26], our goal in this work is to build practice-oriented recommendations and principles which help designers reason about *why* these recommendations are useful and how we might extend them.

3.2.2 Exploring Alternatives and History Management Tools. Tools for investigating alternatives, from interactive systems to programming constructs, support our theme of *Local Knowledge of Materials*. This category of research investigates supporting users in managing and exploring variations through parallel authoring for image manipulation [85] and interface design [34]. Other approaches to variation have been explored in programming constructs for multiverse analysis [49, 72]. These projects do not track versions, but allow users to explore local variations effectively. In a similar vein, novel tools allowing users to explore similar examples in web design [68] led to better design outcomes [46]. These parallel authoring and example search interfaces strongly resonate with our principle of local knowledge building through variation and comparison, which we discuss in Section 2.2.

Tools to support the organization and communication of the design process have found success through contextual annotation and flexible arrangement, which support our theme of *Annotation for Holistic Interpretation*. These projects have explored how annotations and virtual canvas editors help communicate the design process [15, 60] and help manage ideas and support reflection [39, 50, 51, 77]. These works offer support for our principle in Section 2.3 that knowledge practices center on contextualization and interpretation of versions. Quickpose builds on this work as a version control canvas that primarily supports a practitioner in engaging the material at hand, including the history of versions and annotations, rather than communicating or reflecting afterwards on a design process.

3.2.3 Creative Programming Environments and Constructs. Interfaces for creative coding have prioritized rapid exploration and iteration, and have focused on connecting many small programs together to form directed acyclic graphs (DAGs) which lend themselves to variation [7, 22, 78, 98]. While these implicitly support variation of individual parts of the program, they do not version the program itself. This prevents users from explicitly combining changes across components and using their program history beyond what they manually duplicate. Similarly, p5.fab [84] supports rapid experimentation by allowing users to directly control machine execution, but does not help scaffold this experimentation over time. While all of these tools seem to support material interaction with programs (and, with p5.fab, manufacturing devices), they do not manage versions of their programs. In contrast, Quickpose replicates the entire program in each version, allowing users to easily explore and manage entirely different programs which originated from the same starting point. Quickpose is designed to

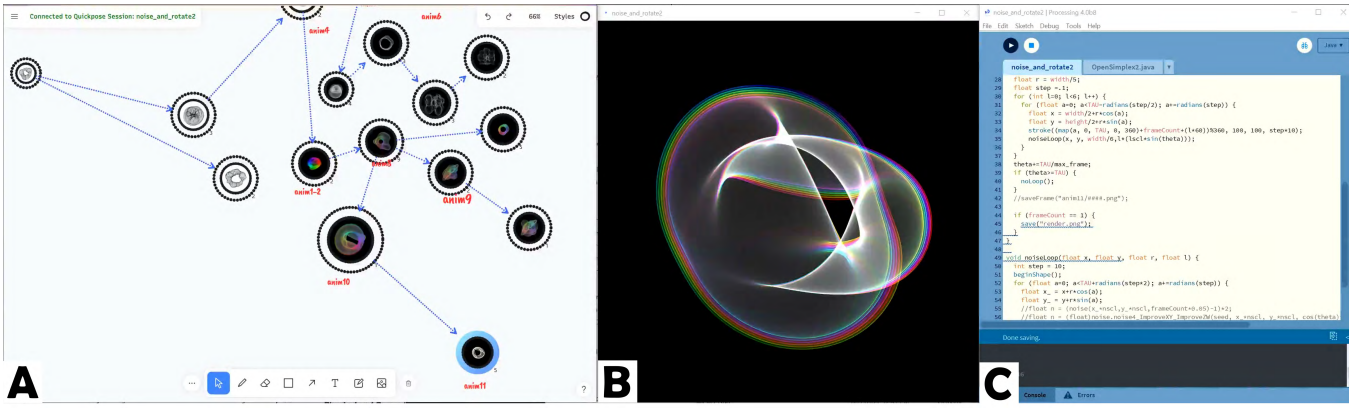

Figure 2: The Quickpose interface alongside the Processing IDE and render output. A. Quickpose integrates a version control history into an interactive canvas, allowing users to flexibly navigate, arrange, and annotate their program versions. Versions are shown as circular thumbnails of their render output. Users can update the render (B) and program state (C) by clicking on a thumbnail corresponding to that version. The current version is indicated with a blue border. B. The output window renders the current Processing program as a still image or animation. C. The Processing IDE is where users edit and execute their Processing programs, or *sketches*.

support more involved exploration and discovery in line with our discussion in Section 2.1.

3.3 Version Control but not Material Interaction

3.3.1 Studying Versioning Strategies for Programmers. Previous research has investigated how programmers take up code *as a material*, often without the explicit support of their programming tools. First, they highlight how programmers already explore, experiment, reason with, and build tacit knowledge about programs and their executions [5, 6]. Second, they emphasize how existing programming environments poorly support programmers in this process [5, 6, 73]. Yoon et al. [94, 96] study programmers' *backtracking*—returning to an earlier state of the program—behaviors to show how current version control tools fail to support these behaviors. In this paper we explore how version control tools can better support programmers in material interaction: reasoning and reflecting on the entire process of programming across versions.

3.3.2 Novel Version Control Tools. Besides supporting experimentation, iteration, and history navigation, version control tools have also focused on annotation of version histories for reflection [88], learning [10, 28, 30], and sharing [52, 62]. These projects highlight the importance of annotation for bringing context to versions and aiding interpretation (Section 2.3).

De Rosso et al. [18] discuss their redesign of Git, a widespread version control system. They were able to address common issues with the system by identifying conceptual errors from users and redesigning Git's abstractions to address those errors. While this line of research is valuable, our investigation in this paper is not meant to deliver design guidance for a system like Git. Git is primarily intended for managing changes in the context of software engineering [18]: communicating and combining changes, tagging versions for institutional purposes (marking releases of software, demarcating experimental or abandoned features, etc), and maintaining a single, common history to restore in case of failure. While these are important aims of software engineering, this paper explores how version control can support an entirely different set of priorities, which we discuss in Section 2.

3.4 Version Control and Material Interaction

Previous version control systems have resonated with our focus of material interaction. Although these works do not discuss material interaction directly, the success of the tools in supporting engagement with materials provides a strong foundation to propose our themes in Section 2. Previous work has included design tools for exploring undo/redo history [59] and managing non-linear histories for image manipulation [12], vector graphics [45, 83], parametric design [97], and interface design [33]. Research in version control tools for programs has explored new ways to use version history, including interactive timelines of program edits [54, 92, 93, 95], and working with and retrieving changes of Jupyter notebook cells [4, 42, 89] or individual lines of code [57].

These projects collectively rely on the principle that exploratory and iterative uses of version history leads to better design outcomes, which strongly aligns with our themes in Sections 2.1 and 2.2. While these works are not presented as such, we would describe them as supporting material interaction as we have described. Not only do they support material interaction with the artifact itself (whether the code, image, or vector graphic), but they also support users in engaging the version history of that artifact as a material in its own right: helping users reflect on, understand, and manipulate their practices of programming. In this paper we build on this work to propose preliminary concepts to explain why these interfaces are successful, alongside design concepts which propose how we might measure and recognize material interaction behaviors in the use of such interfaces.

4 SYSTEM DESCRIPTION

Following our three design principles, Quickpose is a version control system designed to support programmers in engaging version control as a material interaction and subsequently measure that engagement. Built as a companion interface and version control system for the creative coding platform Processing [65], Quickpose integrates versioning capabilities for a Processing program into a direct-manipulation canvas editor, offering a flexible and convenient platform for exploration, comparison, and annotation. Processing was a well-matched IDE within which to build Quickpose, given its active use among expert creative coders, its extensive libraries and community support, and its ethos of building an environment for *sketching* in code [82]. Processing users often generate visual images and animations with their code (in Processing, programs are referred to as *sketches*), iterating between the IDE and the visual output (the render) very quickly.

Quickpose scaffolds this iteration and exploration as a version control tool, saving the render outputs and sketches for each iteration and allowing users to quickly navigate between them. A user would begin a Quickpose session by opening a new Processing Sketch (shown in our system diagram in Fig 2.C) and running Quickpose from within the editor's menus. They would then open their browser to the given URL, which serves the static front end canvas in a web browser. Clicking between versions (shown as circles with images in the center in Fig 2.A) updates both the Processing Code in the IDE (Fig 2.C) and also the render output, which runs as a Java Applet as an additional window (Fig 2.B).

Users can create a copy of any version by shift+clicking on that version (shown in our diagram of features on Fig 3), and the editor will automatically update to that state. The current version is shown on the canvas with a blue highlight (as illustrated in Fig 3). Because Quickpose functions as an interactive canvas, users can flexibly arrange, draw on, style, or label versions (Fig 4). Additionally, users can generate an export folder with the selected code and render outputs from versions depending on the versions' color styling on the canvas, as shown in Fig 5.

4.1 Instantiating Design Principles into Features

In this section we discuss how our three design principles informed the design of Quickpose, highlighting how each feature supports each theme.

4.1.1 Direct Manipulation [79], One-Click Navigation and Forking. Quickpose represents versions as canvas elements with thumbnails of those versions' render outputs. Clicking on a version in the Quickpose interface updates the Processing IDE and output window to that version, while shift-clicking a version creates a new version (a fork), and updates the IDE to that version (Fig 3). To ensure no history is lost, states with child nodes are locked for editing, but are always able to fork.

- **Continual Goal Reformation**: As goals develop and change, users can easily backtrack from any previous iteration to explore a new path, up to and including the very first iteration.
- **Contextual Exploration**: The version history graph embedded directly into the canvas provides effective visualization and comparison of groups of versions. Because users can interact with the graph directly, users can navigate and fork versions just by clicking through the visualization.
- **Holistic, Linked Annotation**: Seeing the thumbnails of all states at once allows reflection of the entire process. Because clicking on each version thumbnail in the Quickpose canvas updates the IDE and render state, the program state and canvas annotations are directly linked.

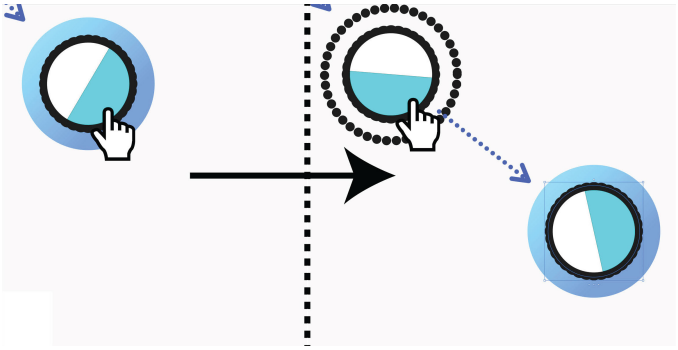

Figure 3: Shift+Clicking on a version creates a new copy (a fork) and updates the program state to edit this new version

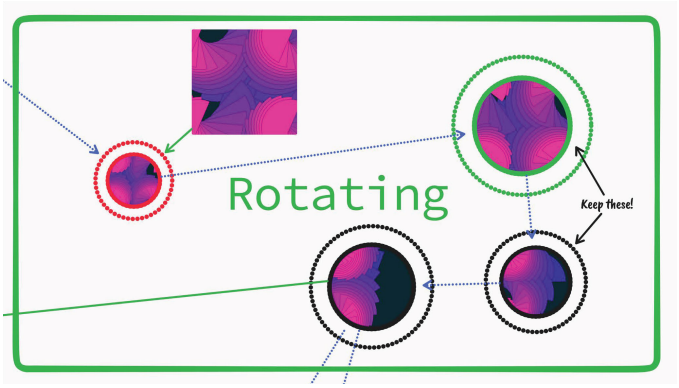

Figure 4: Users can annotate specific versions with text labels and images, or use labeled shapes or arrows to refer to groups of versions.

4.1.2 Grouping, Anchoring, Annotating, Arranging. Because the version history graph is represented as interactive canvas elements, versions can be grouped, anchored and parented via arrows, styled, and scaled just as users interact with any other canvas element. This also allows versions to be easily integrated into existing patterns of working with canvas editors to organize versions alongside notes, images, and other graphics.

- **Continual Goal Reformation**: Goals, outcomes, or versions for export and further development can be annotated or demarcated on the canvas, either describing a single version, many versions, or the entire process.
- **Contextual Exploration**: Versions can be spatially arranged to informally situate versions among others.

- **Holistic, Linked Annotation**: Annotations can reference many versions through positioning, grouping, or parenting via arrows. Additionally, annotations and versions can be arranged and styled flexibly and can comprise multiple kinds of media (text, images, illustrations, etc).

4.1.3 Checkpointing, Autosaving. While forking and navigation between versions are manual operations, Quickpose also saves "checkpoints": edits to a version which are saved internally when a user executes a save or when they navigate away from a version having made edits. This ensures that edits are not lost or accidentally overwritten even if a user has not dedicated a new version to hold those changes yet.

- **Continual Goal Reformation**: Users can retrieve overwritten checkpoints and create versions from them if their goals or framing changes and they become newly valuable or interesting.
- **Contextual Exploration**: Users can switch between many versions quickly without fear that changes might be lost or overwritten accidentally.

4.1.4 Export by Color. Because versions in Quickpose behave like other canvas elements, style elements like color can be used to not only informally demarcate certain versions, but can also be used as a handle for technical features, such as the "export by color" menu which exports all the thumbnails and code iterations of a single color into a folder in the user's local project folder.

- **Continual Goal Reformation**: Exporting by color allows users to export multiple versions on the same branch or across branches as a single outcome.

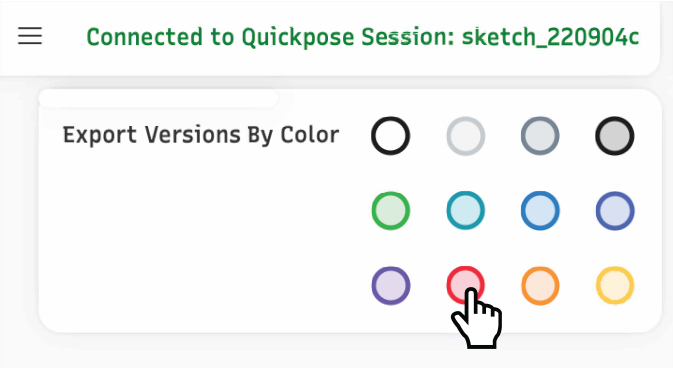

Figure 5: Export by Color allows users to export versions by their canvas color

4.2 Implementation

Quickpose is implemented in two parts: a background daemon in the Processing IDE environment which listens for changes, stores local data, and updates the IDE, and a static front-end web interface which connects to the daemon via a local network connection (Fig 4). The front-end is built onto the Tldraw canvas library [70] to display live updates of versions alongside canvas annotations. Like other version control systems, we represent versions as a *directed acyclic graph* of nodes and links between nodes, but allow the user to interact directly with this representation by displaying them as generated, interactive canvas elements.

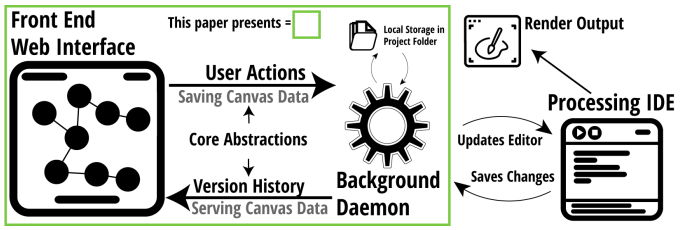

Figure 6: Quickpose is implemented as a front-end interface and a back-end daemon for Processing. These two components communicate through our "Core Abstractions", which define a basic API for reimplementing the Quickpose front-end for another domain.

4.2.1 Local Data and Logging. In order to study how practitioners use Quickpose, regular backups of the canvas are made in addition to logs of user activity. These logs and archives are stored in the user's project folder locally. These logs gave us a detailed picture of user interaction with the tool, which we analyze in our user study.

4.2.2 Bringing Quickpose to Other Domains. While Processing was a convenient place to implement Quickpose given its user base of artists and its focus on programs which generate visual outputs, the Quickpose interface can be used in other domains. As a front-end system for managing versions, it could handle versions in a variety of other places, such as an image editor, programming interface, or design tool. We describe the API structure ("Core Abstractions" in Fig 6) below. These are the only abstractions needed to implement Quickpose for a new programming environment. For example, an image editor could interface with the core of Quickpose as long as it was able to change between versions, make copies of versions, and serve thumbnail images. Due to the small and well-defined nature of these abstractions, which make few assumptions about programming environment or language, we argue Quickpose can be straightforwardly adapted to a variety of contexts.

5 STUDY DESIGN

Our study of how practitioners use Quickpose is designed to understand (1) how users engage in material interaction behaviors if their tools offer the functionality for doing so, and (2) whether the measures suggested by the collected themes present initial promise for recognizing such practices. We do not focus on evaluating the tool's usability, its success on a predefined task, nor trying to validate the themes presented. Instead, we explore how we can recognize behavior associated with material interaction. Therefore, we study how indicators of material interaction appear in participants' use of version control when that version control tool (Quickpose) is designed to permit such behaviors. As a generative [2] exploration of supporting and measuring material interaction, this study is not designed to provide *predictive* power (that is, it could suggest, in a falsifiable way, how certain design changes will influence user

Actions	Parameters (Inputs)	Server Responses (Outputs)
Select	Version ID	New Versions file
Fork	Version ID	New Versions file
SaveCanvas	Canvas version	Nothing
GetCanvas	Nothing	Canvas File
Image	ID of Version	Thumbnail Image
Export	List of Version IDs	Export to Local Files

Table 1: The Quickpose API was designed to be simple to implement and not geared specifically for Processing or Creative Coding. Any domain which has the basic concepts of versions, forking, navigating, and exporting could reimplement these abstractions.

behavior), but rather seeks to understand what potential our collective themes (and subsequent measures and design principles) have for supporting and recognizing the behaviors they suggest in the specific context of Quickpose. We discuss how our findings might generalize and support future work in other domains in Section 7.2.2.

Although Quickpose is not first and foremost a creativity support tool due to its focus on material interaction instead of creativity, our study nonetheless follows many of Remy et al.'s recommendations for CST evaluation: we deployed Quickpose in an in-situ, longitudinal study among four expert users of Processing for 3-5 weeks [66]. Recruited from Processing and Creative arts communities and related email lists, these expert participants used Quickpose in the course of their regular Processing usage and were not given specific goals or tasks, nor were they asked to use the tool for a certain number of hours. Participants were screened for their experience with and planned regular usage of Processing for the duration of the study. Partway and at the conclusion of the study, researchers discussed in semi-structured interviews with each participant their experience of using Quickpose, walking through specific projects and reflecting on their artistic process. Compensated at $40 per hour (for interview time, not time spent using the software) in the form of a gift card, total interview time for each participant ranged from 2 hours to 4 hours. Participants were also asked to (optionally) upload project data, including usage logs, Quickpose canvases, and program versions.

In total, we collected interview recordings, usage logs, canvas data, and project files, including code and renderings. To analyze our data, we conducted provisional coding [71] (done by the first author) on interviews and canvas annotations using measures, listed below, which we created from our themes. Codes were not iteratively defined during the analysis, although we present an updated set of measures in Section 7.2.1. We coded canvas data for annotations which contextualize states, annotations which interpret states, and indications of "final" or "finished" versions. We also coded our interviews with participants for the presented themes more broadly.

We use these themes of material interaction to motivate our study of practitioners' use of Quickpose and propose practical measures for studying material interaction in version control systems. Following our discussion in Section 2, our measures are as follows:

(1) **Measuring Continual Goal Reformation:** Do practitioners significantly reframe their goals over the course of a Quickpose session? Do they make many significant backtracks to pursue a different direction? Do they work on multiple branches at once? Do they export or mark as final multiple versions, either at the ends of multiple branches or sequential points along a single branch?
(2) **Measuring Contextual Exploration:** Do practitioners exhibit both low depth, high degree and high depth, low degree patterns in the course of their explorations? Do they shift between them at different moments of a session? Do practitioners repeatedly backtrack with low depth on the same state or nearby states to capture a single variation?
(3) **Measuring Holistic, Linked Annotation:** Do practitioners use annotations in Quickpose (color, text, positions of versions, images) to contextualize rather than generalize the versions? Do practitioners use annotative practices which support interpretation (emotional or metaphorical language)? Do practitioners use annotation to describe not just individual states, but groups of states or movements between many states?

Theme	Measures
Continual Goal Reformation	Number and depth of backtracks Evidence of simultaneous development on multiple branches Number of versions exported or marked "Final" or "To export"
Contextual Exploration	Navigation patterns between versions Depth and degree of nodes
Holistic, Linked Annotation	Annotations which contextualize states Annotations which support interpretation Annotations which describe movements between states or multiple states

Table 2: Proposed material interaction measures for Quickpose. These measures structure our study below, where we investigate how each measure might reveal or clarify material interaction behaviors.

6 MEASURING AND IDENTIFYING MATERIAL INTERACTION

Three participants submitted ten projects over a study period of five weeks. In total, we captured usage data for 192 hours of usage of Quickpose, 115 versions created, 871 navigations to a different version, 178 annotations created, and 230 checkpoints saved. Because many projects were worked on across multiple periods and with different timescales, timeline views below have been normalized for the time spent with each project running.

6.1 Measuring Continual Goal Reformation

6.1.1 Number and depth of backtracks. We recorded each instance of a user forking a version. For each project, we then calculated the depth (the number of nodes from the new forked node to the *root node* of the version tree) and the distance (the number of nodes on the shortest path from the new forked node to the *previous forked node*). We show the results for three projects in Figure 7. In this plot we see multiple backtracks, represented as sharp slopes up, or in other words, a decrease in depth from the previous fork. These jumps indicate users went back significantly in their history to start again in a new direction.

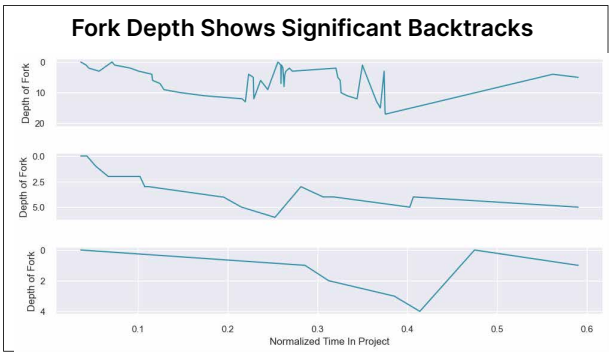

Figure 7: Plotting forks from three projects by normalized project time (x-axis) against depth (y-axis), which is the distance from the root node of the version tree. Steep jumps show significant backtracks.

6.1.2 Development on multiple branches in parallel. Simultaneous development on multiple branches, in contrast to backtracking, is identified by a high distance from the previous fork but a small change in depth from the previous fork. For example, a participant may have two long branches, b1 and b2, fork from the end of branch b1, then fork from the end of b2, between these forks is a small difference in depth (the two forks are similar distances away from the root node), but a high change in distance(the two forks have a large traversal distance between them). Figure 8 shows all instances of forking in our dataset, arranged according to both distance and change in depth; the area marked "Parallel Development" represents the high distance, low difference-in-depth area.

6.1.3 Versions marked as "final" or "complete". From user canvases, we recorded instances of multiple "final" versions, marked either by a text label indicating finality, or, as in Figure 9, a filename indicating an export. In the Figure 9 canvas, the participant was generating video files from desired versions and annotating the version with the filename of the generated video. We therefore interpreted two kinds of "marking final": (1) either referencing an export of the artifact, like the example above, or (2) indicating that a version was to be kept for future development. For example, P2 generated multiple variations of the same sketch but with different aesthetic qualities for different situations. P2 described in an interview with researchers how these versions were intended to form a "palette" for future development. In the four projects where these annotations were present, we recorded 53 annotations of this kind, although one

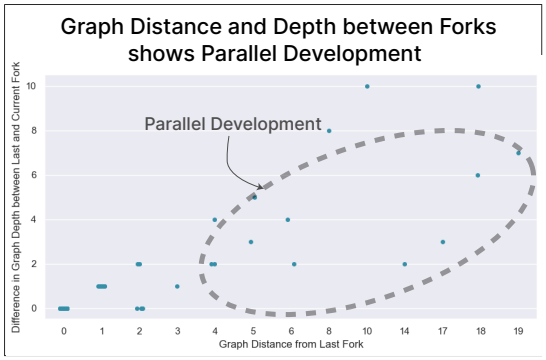

Figure 8: Plotting forks from all projects by distance from last fork (x-axis) against the *difference* in depth from the last fork (y-axis). High distance and low difference in depth (indicated in the dotted ellipse) shows a parallel development between different branches, but not a backtrack.

project contained 36 of these. Many of these annotations (all but 9) were not on leaf nodes and many were on multiple parts of a branch. Marking many successive iterations or intermediate versions each as "final" or "to export" indicates an evolving goal, where marking versions across disparate branches as "final" indicates bifurcation of goals. We did not record any instances of participants using the "Export by Color" feature. We will note, however, that this feature only exported still images, and all of the participants in our study worked on video animation projects.

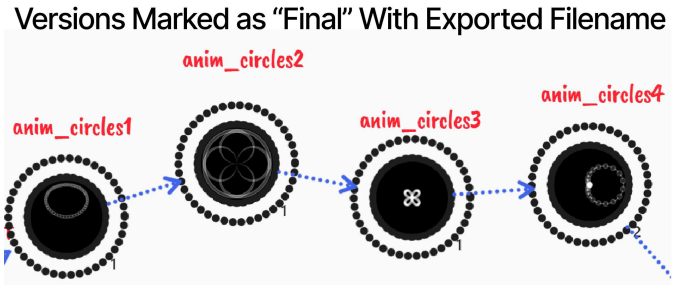

Figure 9: Versions made by P2 show successive versions. Each version is annotated with the video file name which was generated by that version (indicated by dashed circles).

6.2 Measuring Contextual Exploration

6.2.1 Navigation Among Versions. In addition to forking behavior, we also recorded when participants navigated between versions by clicking on them on the Quickpose canvas to update the editor and the render. In Figure 10, we show two Quickpose projects as directed graphs with both the forks (shown in grey) and navigations (shown in blue) between nodes (shown in green). These visualizations show how participants navigated broadly across their version history even in cases in which their forking history shows a more linear behavior; at each version, this navigation history shows evidence

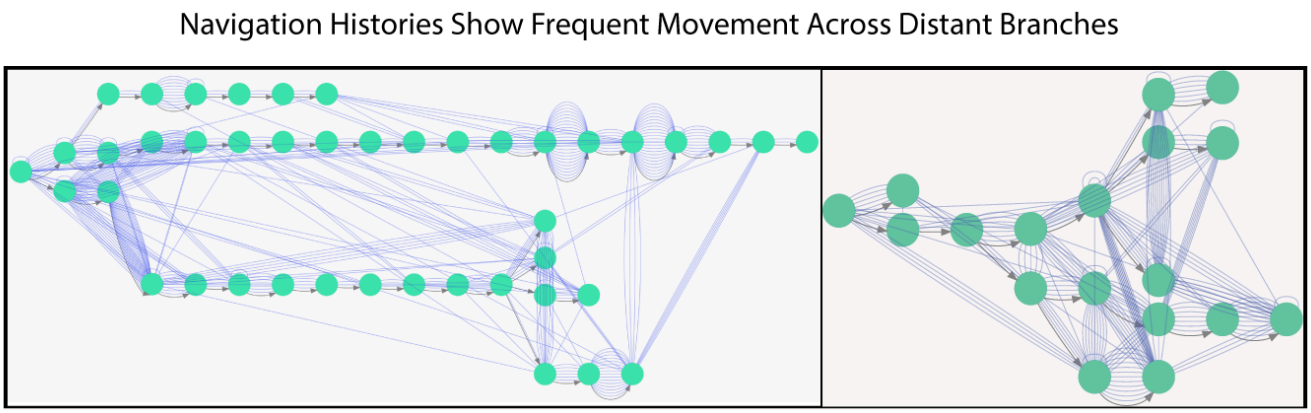

Figure 10: Two Quickpose version graphs visualized. Grey arrows indicate a fork between versions, and blue lines indicate a navigation from one version to the other.

of how participants reflected on and compared versions across their history.

6.2.2 Depth and Degree of Nodes. In order to investigate broad (low depth and high degree) versus deep (high depth and low degree) exploration, we chart the nodes of four projects by degree (how many edges a node has) and depth (its distance from the root node) in Fig 12. We theorized that many high depth, low degree nodes would indicate deep exploration, while low depth, high degree nodes would indicate broad exploration. Our results visualized here show that while participants were able to explore *deeply*, there were relatively few nodes that had more than three edges (i.e., forked more than twice). (See discussion in Section 7, including how future work may support broad exploration.)

6.3 Measuring Holistic, Linked Annotation

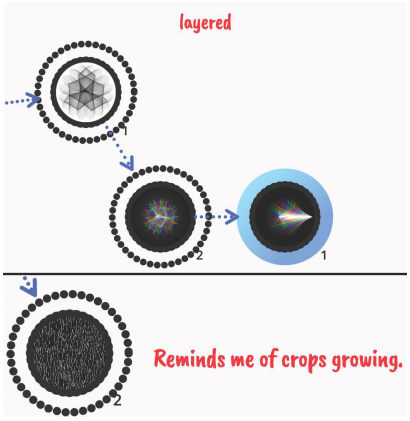

Figure 11: Top: An annotation of "layered" describes a group of three versions. The canvas element group includes both the annotation and the versions. Bottom: A participant uses interpretive language to describe a version rather than summarize it.

6.3.1 Annotations that describe movements between states or multiple states. We saw participants use a single annotation to describe multiple versions in four of the projects. Participants either referenced multiple versions *explicitly*, by "grouping" them together in the Quickpose interface (Fig 11, left), or *implicitly*, through positioning an annotation near multiple versions on the canvas and describing the content of the versions (for example, annotating "these versions rotate instead of translate"). However, we found that these annotations that described multiple versions appeared only once per project. (See discussion in Section 7 for possible reasons why.)

6.3.2 Annotations that support interpretation. In six of the projects, participants used interpretive language to annotate a version. This included comparisons, such as "Reminds me of crops growing" from P3 as shown in Figure 11, or included emotional descriptors like "magical" or "interesting." In the data collected, participants used 13 annotations in this way out of a total of 178 canvas annotations.

6.3.3 Annotations that contextualize states. We found 25 contextualizing annotations in eight of the projects. These annotations added information about the version, why it was created, or why it was or was not interesting to the participant. For example, participants contextualized versions with annotations like "Failed rotation experiment" and "The previous effect but animating forward then reversing in a loop".

6.4 Qualitative Support for Material Interaction

At the mid-way point and end of the study, participants were interviewed for approximately 60 minutes on their experiences using Quickpose and their artistic process. Researchers did not explicitly mention material interaction in these interviews, instead asking participants to narrate their experience of one of their Quickpose canvases and highlight improvements and suggestions for the tool. For example, participants were asked if they learned anything about their artistic process while using Quickpose, or if they noticed that Quickpose was changing their artistic process. In this section we present excerpts from these interviews which relate to our themes

of material interaction and help contextualize our participants' experiences using Quickpose.

6.4.1 Evidence for Continual Goal Reformation. All of our participants mentioned how Quickpose supported exploration of multiple goals by backtracking in order to explore new possibilities. P3 highlighted how Quickpose helped support goal formation through supporting exploration of many different directions: *P3: "Quickpose is really good for when you don't really know where you're going."* P2 discussed how Quickpose supported iteratively backtracking as their goals refined and changed, allowing them to collect all their explorations in a single place for reflection and re-use: *P2: "I would start a branch and explore it all the way down...you keep some of your changes and incorporate that into a different direction, and other times you...start fresh and...do something totally different...in which case you go back to the start and branch off."* Our participants all indicated that saving and storing many versions quickly allowed them to more easily explore, especially if their goals were not well defined. P3 highlighted how the low barrier to make a new version allowed saving versions which might only become meaningful later: *P3: "If you don't know where you're going, then you've got to record everything."* P2 additionally highlighted how quickly saving many versions enabled a sustained focus on exploration, rather than thinking about version control as a separate step in the programming process: *P2: "What's really great about [Quickpose] is that it frees you up...I can always go back to the root and go in a completely different direction after working for three hours in one way and it's all still contained and you don't break that flow."*

6.4.2 Evidence for Contextual Exploration. P3 and P2 expressed that Quickpose supported context building, idea generation, and comparison. P2 draws attention to how seeing all the versions at a glace supported more exploration: *P2: "There's more immediate inspiration...Having [all the versions] there gives you more ideas...I think that's why [the project] branched out so quickly."* All participants also mentioned that Quickpose lowered the barrier for exploring variations. P1 described how Quickpose supported them iterating locally on a single "base file:" *P1: "You can have a base file from where you can try different things...tweaking it a little or tweaking it a lot."* P1 and P2 indicated that exploring local variations aided building local knowledge about programs. P2 highlighted how easily forking aided their ability to form tacit knowledge about their code, discussing how seeing local variations together on the Quickpose canvas helped them understand the parameter space of their program: *P2:"...just playing with numbers, I wouldn't even really understand [the code]—I really do have to save this, because I don't really understand what it's doing."* In all of these excerpts, participants found that seeing all variations at a glance and creating and navigating versions quickly helped how they form ideas and reason about their programs, features which we designed Quickpose to enable specifically.

6.4.3 Evidence for Holistic, Linked Annotation. P3 mentioned that Quickpose supported annotating and organizing in an idiosyncratic way: *P3: "That's what Quickpose is for, whatever kind of organization you want."* P2 referred to the Quickpose canvas as a "mindmap," allowing them to map out their ideas and explorations through successive versions. P3 requested that web pages be integrated into the Quickpose canvas itself to store reference material within the version history, analogous to *"leaving a comment in the source code."* P3 cited how they would like to embed articles and resources that were not just about a single version, but were aiding the exploration more holistically.

6.4.4 How This Evidence Supports Our Themes. Through the interviews, participants expressed their perception of their programming interactions while working with Quickpose. These excerpts present evidence that participants' perceptions of their experiences with Quickpose aligned with our theories of material interaction. While testimony from participants is not sufficient on its own, it further supports that our measures of material interaction were in fact measuring salient aspects of the user experience.

7 DISCUSSION

In this section we first show how Quickpose scaffolds research on material interaction, using themes from our interviews. We use this discussion to outline future directions for version control that support the formation and development of goals. We then review our measures in light of our study results to recommend refined measures. Finally, we discuss how the themes we distilled for material interaction might help clarify existing conversations in HCI and how this work could support the development of general theories of material interaction for interfaces. We also reflect on what remains to (i) produce a *predictive* theory of material interaction, (ii) test (and attempt to falsify) that theory, and (iii) design with a theory of material interaction in mind.

7.1 Giving Structure to Process

From our study and analysis, we see that Quickpose served as a platform for studying material interaction because it allowed practitioners to express how they were working and reasoning with materials in a format which researchers could analyze and track. From a user perspective, Quickpose served as a language for expressing process because it was the environment in which practitioners actively reasoned about their artifacts (in this case, programs and renders) in tandem with goals. For example, P2 and P3 said that Quickpose helped them think and explore in a more "structured" way. We interpret "structured" here as meaning that Quickpose gave them a way to instantiate and represent in a computational tool how they were working through their process. P2 drew explicit attention to Quickpose as a platform for externalizing and recording their cognition [36], describing it as a "mindmap." Likewise, we recall P3's earlier comment on how the flexibility of the annotations supported externalizing an idiosyncratic workflow for "whatever kind of organization you want." Therefore, we argue that Quickpose succeeded as a tool for studying material interaction because it was also a language for practitioners to articulate their process. This framing of tools—as simultaneously a platform for research and a language for practitioners—helps us not only study material interaction, but also evaluate the success of tools that aim to facilitate it.

7.1.1 Feedback : Feedforward :: Backtracking : Forwardtracking. In Section 6, we discussed how practitioners reason about their work

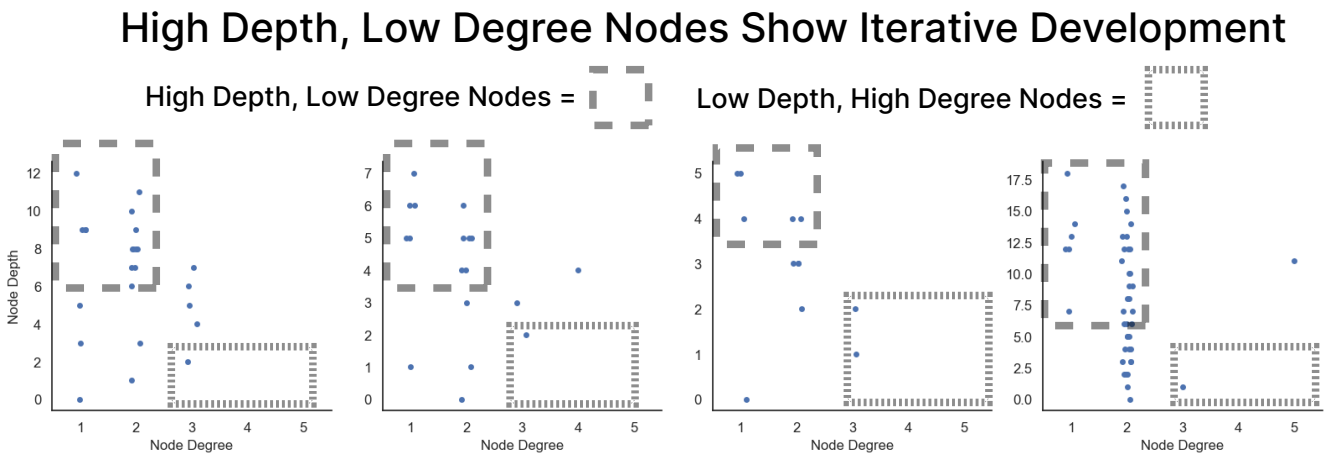

Figure 12: The nodes from four projects graphed by depth and degree (a node's number of edges). High depth, low degree nodes indicate deep exploration (indicated with the dashed box on the top left of each plot), while low depth, high degree nodes indicate broad exploration(indicated with dashed box on the bottom right of each plot).

with Quickpose, which includes not only understanding and reflecting on past versions, but also *future* directions, intentions, and goals. We found evidence that once version control (Quickpose) became the platform for reasoning about the materials at hand (and taken up as a material itself), participants also indicated future directions, latent possibilities, and unarticulated goals. For example, P1 annotated versions with what possibilities or variations had yet to be explored. Analogous to feedback (understanding what a computer has just done) and feedforward (understanding what a computer is about to do), we propose a similar counterpart to backtracking (navigating to an earlier version of the code) as *forwardtracking*: outlining future directions, intentions, and unexplored space in a user's "mindmap." Our discussion of continuous goal reformation highlighted how working with materials is a continuous process of reflecting on prior versions, engaging actively with the material at hand, and using both to project into the future. With our distilled themes of material interaction in hand, we argue that version control should then support "forwardtracking" as a practice of setting and refining goals, which includes outlining unexplored or latent possibilities in materials, as an important part of supporting material interaction.

7.2 Towards Practical Concepts for Material Interaction in Interfaces

Beyond the considerations for Quickpose specifically, we envision a theory of material interaction for interfaces which would propose (1) how to support material interaction through tool design, (2) how to recognize and measure material interaction when it occurs, and (3) how to explain and reason about material interaction behaviors in computer interfaces. While this work does not propose such a theory, it uses existing literature on material interaction to propose initial themes and preliminary measures which could scaffold future theoretical work and system building. As Beaudouin-Lafon et al suggest [2], generative construction of theories in HCI most productively happens in domain-specific, well-bounded contexts – in-tandem with iterative implementation and testing. With this in mind, a general theory of material interaction for interfaces might only emerge from a mosaic of many explorations, systems, and applied concepts – what Höök and Löwgren call *vertical grounding* [37].

This study was not designed to offer *predictive* power – the results could not be used to predict how future users would change their behavior in response to specific design interventions. However, the value of a predictive theory of material interaction for interfaces is still an open question. While a predictive theory may be helpful to the requirements we have outlined above, it also does not seem required to generate useful design principles and outcomes. Höök and Löwgren [37] emphasize "strong concepts" in HCI which offer a design idiom or language to interface designers. These can be flexibly applied to new domains, require skill to fruitfully employ, and do not require predictive power for their ability to generate new designs, ideas, and conceptual lenses. Future development of a predictive theory of material interaction will need to articulate why it needs such predictive power, and for what ends. While much work remains to build theories of material interaction for interface design, we argue that the themes we have drawn together from existing literature already present *explanatory* power to clarify existing recommendations and findings about material interaction– presenting a meaningful place to start on such work. We show this below in a case study. Additionally, we present our refined, post-study measures to aid further development and exploration of material interaction in interfaces.

7.2.1 Measures for Material Interaction. Using the results of our study, we present an updated set of measures for material interaction for future research (Table 3).

Measuring continual goal reformation. We found that significant backtracks could be characterized by a sharp decrease in depth from a previous fork, simultaneous development could be indicated by forks with a small change in depth but a high distance between them, and that versions marked as exports or to be used later were

valuable ways to capture how goals were iteratively refined. While we did not record any instances of our "Export by Color" feature, we saw through canvas annotations, later supported by interviews, that participants had exported animations as videos, which our tool didn't support. We also observed that participants left "final" versions within Quickpose canvases to draw on within future projects. Therefore, instead of looking for "exported" versions, we instead propose measuring versions which are either linked to files outside of the Quickpose project, versions marked for future use, or versions which have been drawn into other projects.

Measuring contextual exploration. We found frequent navigation between disparate parts of the version history indicative that participants were reflecting on and comparing across the version history. Additionally, we saw that high depth and low degree nodes could indicate deep exploration while low depth, high degree nodes could indicate broad exploration. Although we did not measure many low depth, high degree nodes, we argue that this is not because the measure is faulty, but because Quickpose did not adequately support that interaction. All of our participants requested greater support for procedurally making versions by varying one or a few variables—such as Quickpose generating a design gallery across the variables—indicating they felt unsupported in broad variation.

Measuring holistic, linked annotation. While contextualizing versions were our most common category of annotations, they were also the most vague categorization, and frequently overlapped with interpretive annotations. We therefore refined our measures for contextualizing and interpretive annotations, combining them to annotations which support interpretation and understanding of why a version is meaningful.

These measures were developed within the context of Quickpose (and version control systems more broadly), and therefore we cannot predict how they will generalize to other tools and domains. However, the diverse fields, methods, and practices from which we drew our discussion of material interaction might indicate that they have the potential to be helpful elsewhere. As we discuss below, this can only be answered by putting our themes, principles, and measures into practice in other domains.

7.2.2 Material Interaction in HCI. While this paper does not propose a comprehensive theory of material interaction, in this section we outline how our themes might already be helpful for making sense of previous recommendations for design tools in HCI and help reconcile them with new findings. For example, in his seminal creativity support tools paper [80], Shneiderman recommends keeping a detailed history to record which actions or alternatives have already been tried by the user. This record is intended to support iteration and exploration of the material at hand. However, this recommendation also raises questions: Why is keeping a record of previous actions beneficial? How should interactions with this history be measured? Sterman et al. find evidence of practitioners *preferring* a less detailed, lower fidelity history in some cases [81]—how can this evidence be reconciled with Shneiderman's design recommendation? In this example, we see our themes and design principles of material interaction working to bolster the original recommendation, offering meaningful dimensions to study, and also offering an explanation of Sterman et al.'s finding. First, our themes can help explain *why* keeping records might be beneficial. For example, record-keeping aids in building local knowledge through enabling users to reflect on how their changes have impacted their artifact, or it helps users backtrack if their goals change in response to the material and they need to pursue another direction. Second, our measures suggest that we could study history-keeping by tracking both how users backtrack to earlier versions but also how they use earlier versions to inform current work by navigating across the version history. Finally, these themes of material interaction offer an explanation of why some practitioners prefer lower-fidelity versioning methods [81]. For example, the authors cite a performer who prefers keeping only the audio to their performances so that they can have a record of previous work but also feel able to gradually modify or spontaneously develop the performance. With the themes of material interaction in hand, one reason for this practice could be that versions which do not contain enough information to fully recreate the artifact require exploration and variation on every backtrack. Thus, low-fidelity capture is a process constraint that forces a practitioner to reengage with their material at every step. Where before this practice might have seemed counter-intuitive in light of Shneiderman's recommendation, when viewed with a lens of material interaction, the goal of maintaining a low-fidelity capture to spurn exploration is aligned with Shneiderman's call of engendering innovation through history keeping.

Additionally, our themes of material interaction generate new research directions beyond the scope of Quickpose. For example, it suggests exploring a more expansive definition of version control systems, which go beyond tools for backing up, collaborating, and managing code to also include tools for supporting reflection, comparison, and goal formation. While we propose material interaction as *a lens through which* to study and design version control systems, material interaction within other areas of HCI research remain to be explored. We hope these themes, design principles, and measures will scaffold generative work with material interaction in other domains, which might require them to be reworked and appended.

8 LIMITATIONS AND FUTURE WORK

Scale was the major limitation of our study. While we were able to uncover rich insights by closely investigating three practitioners and their processes across 3-5 weeks, our small number of participants limited the extensibility of our study results. With this in mind, we look forward to the open-source release of Quickpose, where a larger usage study of the tool would become possible. Future methodological work might require more robust validations of measures than presented here–testing the inter-rater reliability of qualitative codes in addition to building codes which can apply to multiple systems. Both of these will be required for authoritative comparative studies of material interaction across interfaces.

To better support practitioners in broad explorations in Quickpose, we draw inspiration from prior work in parametric design galleries [97] as a starting point.

There are many opportunities to extend Quickpose to ask further questions about material interaction: How might Quickpose enable greater opportunities for sharing and collaboration? How does sharing this kind of history affect how people collaborate on

Theme	Initially Proposed Measures	Refined Measures
Continual Goal Reformation	Number and depth of backtracks	Sharp decreases in depth from previous forks
	Evidence of simultaneous development on multiple branches	Forks with small changes in depth but high distance between
	Number of versions exported or marked "Final" or "To export"	Versions linked to file exports Versions marked for future use or used by other projects
Contextual Exploration	Navigation patterns between versions	Frequent navigations between disparate branches
	Depth and degree of nodes	High depth, low degree nodes indicate deep exploration Low depth, high degree nodes indicate broad exploration
Holistic, Linked Annotation	Annotations which contextualize states	Explicit groups of annotations with multiple versions
	Annotations which support interpretation	Annotations implicitly referencing, by content or position, many versions
	Annotations which describe movements between states or multiple states	Annotations supporting interpretation of why a version is meaningful

Table 3: Initial and updated material interaction measures.

Processing projects? How might Quickpose be used in an educational environment to support learning outcomes and reflection, as research has already shown version histories are promising in this domain [29]? We are also interested in investigating the ideas behind Quickpose in non-visual, creative domains in addition to domains not seen as "creative": for instance, how might we support and measure material interaction in more traditional software engineering environments?

Additionally, previous work in exploratory version control systems have engaged scale (the navigation and retrieval of many different versions) as a major challenge in the domain. Kery [41] discusses the difficulty of the Verdant-1 system in scaling beyond a few versions, which were surfaced as inline alternatives to Jupyter notebook cells. In later work, they emphasize the importance of visual search and diverse ways of retrieving information, which appear, from our initial study, to be strengths of Quickpose. Quickpose's design principles may suggest further work in this area: having a version control system serve as a "mindmap" seemed to allow participants to organize and navigate between tens of versions at once, which we discuss in Section 6.2. Future work could test to what extent Quickpose's flexible visual layout and open-ended annotation aids search and retrieval of versions, in conversation with theories of information foraging [63] and context reinstatement [8]. At the same time, future research systems for notebook versioning like Verdant could explore the continuous annotation and curation which allowed Quickpose to feel more like a "mindmap" than a list of commit messages, integrating design features of Quickpose into versioning systems for data science and computational notebooks.

The themes of material interaction are far from complete as discussed here. For example, timescale is a major unexplored part of how practitioners work with materials, but a major contributor to a practitioner's development of knowledge, tools, and experimentation practices. All of our participants mentioned that the 3-5 week study period was too short for them to utilize many of the projects they worked on, with P1 saying that it is not uncommon for them to visit a project more than a year later. Additionally, P2 and P3 mentioned how they have built up, over the course of their entire time working in Processing, personal libraries of code snippets for use in future work. We argue that material practice is a lifelong engagement, one in which practitioners build environments, tools, and knowledge over months and years of work. Studying a long-term material practice echoes similar recommendations for longitudinal studies of creativity support tools [58, 81] and building community support for research tools [48].

9 CONCLUSION

We presented three themes of material interaction, which distilled an existing, high-level discussion of how practitioners engage their materials into actionable claims and measures for interfaces. We used these claims to generate design principles which informed Quickpose, a version control tool for creative coding. Quickpose provided a platform to measure behaviors associated with material interaction and better understand them by giving users a language to express their process. To investigate the initial promise of our proposed measures, we conducted an in-situ, longitudinal study with expert creative coding practitioners. We found some of our proposed measures revealed evidence of behaviors associated with material interaction and contextualized these findings through interviews with participants. We argue that operationalizing an existing discussion of material interaction gave insight into practitioners' processes, clarifies existing recommendations, and suggests future directions for research. In doing so, we hope to scaffold future *generative* exploration of material interaction through the design and study of new systems.

ACKNOWLEDGMENTS

We would like to thank the many colleagues who gave suggestions and critique throughout the project, especially Sarah Sterman, J.D. Zamfirescu-Pereira, Jennifer Jacobs, and Camille Utterback. We also thank our study participants for their time and thoughtful reflections. This work is supported in part by NSF grant CA-HDR 2033558. Sarah E. Chasins is a Chan Zuckerberg Biohub Investigator.

REFERENCES

[1] Eli Alshanetsky. 2019. *Articulating a Thought.* Oxford University Press.

[2] Michel Beaudouin-Lafon, Susanne Bødker, and Wendy E. Mackay. 2021. Generative Theories of Interaction. *ACM Transactions on Computer-Human Interaction* 28, 6 (Dec. 2021), 1–54. https://doi.org/10.1145/3468505

[3] Jane Bennett. 2010. *Vibrant Matter: A Political Ecology of Things.* Duke University Press. https://doi.org/10.2307/j.ctv111jh6w.

[4] Mary Beth Kery. 2017. Tools to support exploratory programming with data. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, Raleigh, NC, USA, 321–322. https://doi.org/10.1109/vlhcc.2017.8103490

[5] Mary Beth Kery and Brad A. Myers. 2017. Exploring exploratory programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, Raleigh, NC, 25–29. https://doi.org/10.1109/vlhcc.2017.8103446

[6] J. Brandt, P.J. Guo, J. Lewenstein, S.R. Klemmer, and M. Dontcheva. 2009. Writing Code to Prototype, Ideate, and Discover. *IEEE Software* 26, 5 (Sept. 2009), 18–24. https://doi.org/10.1109/ms.2009.147

[7] Cameron Burgess, Dan Lockton, Maayan Albert, and Daniel Cardoso Llach. 2020. Stamper: An Artboard-Oriented Creative Coding Environment. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems.* ACM, Honolulu HI USA, 1–9. https://doi.org/10.1145/3334480.3382994

[8] Gaston R. Cangiano. 2011. *Studying episodic access to personal digital activity : activity trails prototype.* Ph. D. Dissertation. UC San Diego. https://escholarship.org/uc/item/7jc2n9zh

[9] J. M. Carroll and W. A. Kellogg. 1989. Artifact as theory-nexus: hermeneutics meets theory-based design. *ACM SIGCHI Bulletin* 20, SI (March 1989), 7–14. https://doi.org/10.1145/67450.67452

[10] Caitlin Cassidy, Max Goldman, and Robert C. Miller. 2018. Glanceable code history: visualizing student code for better instructor feedback. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale (L@S '18).* Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3231644.3231680

[11] Joel Chan and Christian D. Schunn. 2015. The importance of iteration in creative conceptual combination. *Cognition* 145 (Dec. 2015), 104–115. https://doi.org/10.1016/j.cognition.2015.08.008 QID: Q50565694.

[12] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear revision control for images. In *ACM SIGGRAPH 2011 papers on - SIGGRAPH '11.* ACM Press, Vancouver, British Columbia, Canada, 1. https://doi.org/10.1145/1964921.1965000

[13] Ricky Chen, Mychajlo Demko, Daragh Byrne, and Marti Louw. 2021. Probing Documentation Practices: Reflecting on Students' Conceptions, Values, and Experiences with Documentation in Creative Inquiry. In *Creativity and Cognition.* ACM, Virtual Event Italy, 1–1. https://doi.org/10.1145/3450741.3465391

[14] Andy Clark and David Chalmers. 1998. The Extended Mind. *Analysis* 58, 1 (1998), 7–19. https://doi.org/10.1093/analys/58.1.7 Publisher: [Analysis Committee, Oxford University Press].

[15] Verina Cristie and Sam C. Joyce. 2017. Capturing And Visualising Parametric Design Flow Through Interactive Web Versioning Snapshots. In *Proceedings of IASS Annual Symposia*, Vol. 2017. International Association for Shell and Spatial Structures (IASS), 1–8. Issue: 5.

[16] Nigel Cross. 1982. Designerly ways of knowing. *DESIGN STUDIES* 3, 4 (1982), 7.

[17] Peter Dalsgaard. 2017. Understanding the Nature and Role of Tools in Design. *International Journal of Design* 11, 1 (2017), 13.

[18] Santiago Perez De Rosso and Daniel Jackson. 2016. Purposes, concepts, misfits, and a redesign of git. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications.* ACM, Amsterdam Netherlands, 292–310. https://doi.org/10.1145/2983990.2984018

[19] Kristin N. Dew and Daniela K. Rosner. 2018. Lessons from the Woodshop: Cultivating Design with Living Materials. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, Montreal QC Canada, 1–12. https://doi.org/10.1145/3173574.3174159

[20] Alan Dix and Layda Gongora. 2011. Externalisation and design. In *Procedings of the Second Conference on Creativity and Innovation in Design (DESIRE '11).* Association for Computing Machinery, New York, NY, USA, 31–42. https://doi.org/10.1145/2079216.2079220

[21] Steven Dow, Julie Fortuna, Dan Schwartz, Beth Altringer, Daniel Schwartz, and Scott Klemmer. 2011. Prototyping Dynamics: Sharing Multiple Designs Improves Exploration, Group Rapport, and Results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).* Association for Computing Machinery, New York, NY, USA, 2807–2816. https://doi.org/10.1145/1978942.1979359 event-place: Vancouver, BC, Canada.

[22] Marc Downie. 2008. Field—a new environment for making digital art. *Computers in Entertainment* 6, 4 (Dec. 2008), 1–34. https://doi.org/10.1145/1461999.1462006

[23] James Elkins. 1999. *What painting is: how to think about oil painting, using the language of alchemy.* Routledge, New York.

[24] Es Devlin. 2020. Es Devlin Culture in Quarantine Masterclass. https://www.youtube.com/watch?v=58UroGqQ1ls

[25] Raune Frankjær and Peter Dalsgaard. 2018. Understanding Craft-Based Inquiry in HCI. In *Proceedings of the 2018 Designing Interactive Systems Conference.* ACM, Hong Kong China, 473–484. https://doi.org/10.1145/3196709.3196750

[26] Jonas Frich, Michael Mose Biskjaer, Lindsay MacDonald Vermeulen, Christian Remy, and Peter Dalsgaard. 2019. Strategies in Creative Professionals' Use of Digital Tools Across Domains. In *Proceedings of the 2019 on Creativity and Cognition.* ACM, San Diego CA USA, 210–221. https://doi.org/10.1145/3325480.3325494

[27] Henrik Gedenryd. 1998. *How designers work - making sense of authentic cognitive activities.* Doctoral Thesis (monograph). Cognitive Science. ISBN: 9789162832100.

[28] Shiry Ginosar, Luis Fernando De Pombo, Maneesh Agrawala, and Bjorn Hartmann. 2013. Authoring multi-stage code examples with editable code histories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology.* ACM, St. Andrews Scotland, United Kingdom, 485–494. https://doi.org/10.1145/2501988.2502053

[29] Elena L. Glassman, Jeremy Scott, Rishabh Singh, Philip J. Guo, and Robert C. Miller. 2015. OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale. *ACM Transactions on Computer-Human Interaction* 22, 2 (April 2015), 1–35. https://doi.org/10.1145/2699751

[30] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10).* Association for Computing Machinery, New York, NY, USA, 143–152. https://doi.org/10.1145/1866029.1866054

[31] Kai Hakkarainen. 2009. A knowledge-practice perspective on technology-mediated learning. *International Journal of Computer-Supported Collaborative Learning* 4, 2 (June 2009), 213–231. https://doi.org/10.1007/s11412-009-9064-x

[32] Donna Haraway. 1988. Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective. *Feminist Studies* 14, 3 (1988), 575–599. https://doi.org/10.2307/3178066 Publisher: Feminist Studies, Inc..

[33] Björn Hartmann, Sean Follmer, Antonio Ricciardi, Timothy Cardenas, and Scott R. Klemmer. 2010. d.note: revising user interfaces through change tracking, annotations, and alternatives. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10.* ACM Press, Atlanta, Georgia, USA, 493. https://doi.org/10.1145/1753326.1753400

[34] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. 2008. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st annual ACM symposium on User interface software and technology - UIST '08.* ACM Press, Monterey, CA, USA, 91. https://doi.org/10.1145/1449715.1449732

[35] James Hollan, Edwin Hutchins, and David Kirsh. 2000. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction* 7, 2 (June 2000), 174–196. https://doi.org/10.1145/353485.353487

[36] Edwin Hutchins. 2006. *Cognition in the wild* (8. pr ed.). MIT Press, Cambridge, Mass.

[37] Kristina Höök and Jonas Löwgren. 2012. Strong concepts: Intermediate-level knowledge in interaction design research. *ACM Transactions on Computer-Human Interaction* 19, 3 (Oct. 2012), 1–18. https://doi.org/10.1145/2362364.2362371

[38] Tim Ingold. 2013. *Making: anthropology, archaeology, art and architecture.* Routledge, London ; New York.

[39] Nanna Inie, Jonas Frich, and Peter Dalsgaard. 2022. How Researchers Manage Ideas. In *Creativity and Cognition.* ACM, Venice Italy, 83–96. https://doi.org/10.1145/3527927.3532813

[40] Heekyoung Jung and Erik Stolterman. 2012. Digital form and materiality: propositions for a new approach to interaction design research. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction Making Sense Through Design - NordiCHI '12.* ACM Press, Copenhagen, Denmark, 645. https://doi.org/10.1145/2399016.2399115

[41] Mary Beth Kery. 2021. *Designing Effective History Support for Exploratory Programming Data Work.* Ph. D. Dissertation. Carnegie Mellon University, Pittsburgh, Pennsylvania, United States.

[42] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, Denver Colorado USA, 1265–1276. https://doi.org/10.1145/3025453.3025626

[43] David Kirsh. 2013. Embodied cognition and the magical future of interaction design. *ACM Transactions on Computer-Human Interaction* 20, 1 (March 2013), 1–30. https://doi.org/10.1145/2442106.2442109

[44] Scott R. Klemmer, Björn Hartmann, and Leila Takayama. 2006. How bodies matter: five themes for interaction design. In *Proceedings of the 6th ACM conference on Designing Interactive systems - DIS '06.* ACM Press, University Park, PA, USA, 140. https://doi.org/10.1145/1142405.1142429

[45] Siniša Kolarić, Robert Woodbury, and Halil Erhan. 2014. CAMBRIA: a tool for managing multiple design alternatives. In *Proceedings of the 2014 companion publication on Designing interactive systems - DIS Companion '14.* ACM Press, Vancouver, BC, Canada, 81–84. https://doi.org/10.1145/2598784.2602788

[46] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10.* ACM Press, Atlanta, Georgia, USA, 2257. https://doi.org/10.1145/1753326.1753667

[47] Jingyi Li, Joel Brandt, Radomír Mech, Maneesh Agrawala, and Jennifer Jacobs. 2020. Supporting Visual Artists in Programming through Direct Inspection and Control of Program Execution. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–12. https://doi.org/10.1145/3313831.3376765

[48] Jingyi Li, Sonia Hashim, and Jennifer Jacobs. 2021. What We Can Learn From Visual Artists About Software Development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3411764.3445682

[49] Yang Liu, Alex Kale, Tim Althoff, and Jeffrey Heer. 2021. Boba: Authoring and Visualizing Multiverse Analyses. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (Feb. 2021), 1753–1763. https://doi.org/10.1109/tvcg.2020.3028985 Conference Name: IEEE Transactions on Visualization and Computer Graphics QID: Q100482061.

[50] Andrés Lucero. 2012. Framing, aligning, paradoxing, abstracting, and directing: how design mood boards work. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. Association for Computing Machinery, New York, NY, USA, 438–447. https://doi.org/10.1145/2317956.2318021

[51] Nic Lupfer, Andruid Kerne, Rhema Linder, Hannah Fowler, Vijay Rajanna, Matthew Carrasco, and Alyssa Valdez. 2019. Multiscale Design Curation: Supporting Computer Science Students' Iterative and Reflective Creative Processes. In *Proceedings of the 2019 on Creativity and Cognition*. ACM, San Diego CA USA, 233–245. https://doi.org/10.1145/3325480.3325483

[52] Mark Mahoney. 2018. Storyteller: a tool for creating worked examples. *Journal of Computing Sciences in Colleges* 34, 1 (Oct. 2018), 137–144.

[53] Lambros Malafouris. 2013. *How Things Shape the Mind: A Theory of Material Engagement*. https://doi.org/10.7551/mitpress/9476.001.0001

[54] Katsuhisa Maruyama, Takayuki Omori, and Shinpei Hayashi. 2016. Slicing Fine-Grained Code Change History. *IEICE Transactions on Information and Systems* E99.D, 3 (2016), 671–687. https://doi.org/10.1587/transinf.2015edp7282

[55] Malcolm McCullough. 1996. *Abstracting Craft: The Practiced Digital Hand*. MIT Press, Cambridge, MA, USA.

[56] David A. Mellis, Sam Jacoby, Leah Buechley, Hannah Perner-Wilson, and Jie Qi. 2013. Microcontrollers as material: crafting circuits with paper, conductive ink, electronic components, and an "untoolkit". In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. Association for Computing Machinery, New York, NY, USA, 83–90. https://doi.org/10.1145/2460625.2460638

[57] Hiroaki Mikami, Daisuke Sakamoto, and Takeo Igarashi. 2017. Micro-Versioning Tool to Support Experimentation in Exploratory Programming. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver Colorado USA, 6208–6219. https://doi.org/10.1145/3025453.3025597

[58] Hedieh Moradi, Long N Nguyen, Quyen-Anh Valentina Nguyen, and Cesar Torres. 2022. Glaze Epochs: Understanding Lifelong Material Relationships within Ceramics Studios. In *Sixteenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '22)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3490149.3501310

[59] Brad A. Myers, Ashley Lai, Tam Minh Le, YoungSeok Yoon, Andrew Faulring, and Joel Brandt. 2015. Selective Undo Support for Painting Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, Seoul Republic of Korea, 4227–4236. https://doi.org/10.1145/2702123.2702543

[60] Jasper O'Leary, Holger Winnemöller, Wilmot Li, Mira Dontcheva, and Morgan Dixon. 2018. Charrette: Supporting In-Person Discussions around Iterations in User Interface Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–11. https://doi.org/10.1145/3173574.3174109

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

[62] Soya Park, Amy X. Zhang, and David R. Karger. 2018. Post-literate Programming: Linking Discussion and Code in Software Development Teams. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings (UIST '18 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 51–53. https://doi.org/10.1145/3266037.3266098

[63] Peter Pirolli. 2007. *Information foraging theory: adaptive interaction with information*. Oxford University Press, Oxford ; New York. OCLC: ocm70334982.

[64] Michael Polanyi. 2009. *The Tacit Dimension*. University of Chicago Press, Chicago, IL. https://press.uchicago.edu/ucp/books/book/chicago/T/bo6035368.html

[65] Casey Reas and Ben Fry. 2007. *Processing: a programming handbook for visual designers and artists*. MIT Press, Cambridge, Mass. OCLC: ocm73993935.

[66] Christian Remy, Oliver Bates, Jennifer Mankoff, and Adrian Friday. 2018. Evaluating HCI Research beyond Usability. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal QC Canada, 1–4. https://doi.org/10.1145/3170427.3185371

[67] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. 2005. Design Principles for Tools to Support Creative Thinking. (Jan. 2005). https://doi.org/10.1184/R1/6621917.v1 Publisher: Carnegie Mellon University.

[68] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. d.tour: style-based exploration of design example galleries. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, Santa Barbara, California, USA, 165. https://doi.org/10.1145/2047196.2047216

[69] Daniela K. Rosner, Miwa Ikemiya, and Tim Regan. 2015. Resisting Alignment: Code and Clay. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, Stanford California USA, 181–188. https://doi.org/10.1145/2677199.2680587

[70] Steve Ruiz. 2022. Tldraw. https://tldraw.com/

[71] Johnny Saldana. 2015. *The Coding Manual for Qualitative Researchers*. SAGE.

[72] Abhraneel Sarma, Alexander Kale, Michael Jongho Moon, Nathan Taback, Fanny Chevalier, Jessica Hullman, and Matthew Kay. 2021. *multiverse: Multiplexing Alternative Data Analyses in R Notebooks*. Technical Report. OSF Preprints. https://doi.org/10.31219/osf.io/yfbwm type: article.

[73] Toby Schachman. 2012. Alternative programming interfaces for alternative programmers. In *Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software - Onward! '12*. ACM Press, Tucson, Arizona, USA, 1. https://doi.org/10.1145/2384592.2384594

[74] D. A. Schön. 1992. Designing as reflective conversation with the materials of a design situation. *Knowledge-Based Systems* 5, 1 (March 1992), 3–14. https://doi.org/10.1016/0950-7051(92)90020-G

[75] Donald A. Schön. 2017. *The Reflective Practitioner: How Professionals Think in Action*. Routledge, United States.

[76] Richard Sennett. 2008. *The Craftsman*. Yale University Press, United Kingdom.

[77] Moushumi Sharmin and Brian P. Bailey. 2013. ReflectionSpace: an interactive visualization tool for supporting reflection- *on* -action in design. In *Proceedings of the 9th ACM Conference on Creativity & Cognition*. ACM, Sydney Australia, 83–92. https://doi.org/10.1145/2466627.2466645

[78] Paul Shen. 2021. natto.dev. https://natto.dev

[79] Shneiderman. 1983. Direct Manipulation: A Step Beyond Programming Languages. *Computer* 16, 8 (Aug. 1983), 57–69. https://doi.org/10.1109/mc.1983.1654471

[80] Ben Shneiderman. 2007. Creativity support tools: accelerating discovery and innovation. *Commun. ACM* 50, 12 (Dec. 2007), 20–32. https://doi.org/10.1145/1323688.1323689

[81] Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. 2022. Towards Creative Version Control. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2 (Nov. 2022), 25. https://doi.org/10.1145/3555756

[82] Liz Stinson. 2021. Processing: the Software that Shaped Creative Coding. https://eyeondesign.aiga.org/processing-the-software-that-shaped-creative-coding/ Section: Digital.

[83] Sara L. Su, Sylvain Paris, Frederick Aliaga, Craig Scull, Steve Johnson, and Frédo Durand. 2009. *Interactive Visual Histories for Vector Graphics*. Technical Report MIT-CSAIL-TR-2009-031. Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Cambridge, MA.

[84] Blair Subbaraman and Nadya Peek. 2022. p5.fab: Direct Control of Digital Fabrication Machines from a Creative Coding Environment. In *Designing Interactive Systems Conference*. ACM, Virtual Event Australia, 1148–1161. https://doi.org/10.1145/3532106.3533496

[85] Michael Terry, Elizabeth D. Mynatt, Kumiyo Nakakoji, and Yasuhiro Yamamoto. 2004. Variation in element and action: supporting simultaneous development of alternative solutions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. Association for Computing Machinery, New York, NY, USA, 711–718. https://doi.org/10.1145/985692.985782

[86] Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. 2006. Getting the right design and the design right. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. Association for Computing Machinery, New York, NY, USA, 1243–1252. https://doi.org/10.1145/1124772.1124960

[87] Barbara Tversky. 2019. *Mind in Motion: How Action Shapes Thought*. Basic Books, United States.

[88] April Yi Wang, Zihan Wu, Christopher Brooks, and Steve Oney. 2020. Callisto: Capturing the "Why" by Connecting Conversations with Computational Narratives. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376740

[89] Nathaniel Weinman, Steven M. Drucker, Titus Barik, and Robert DeLine. 2021. Fork It: Supporting Stateful Alternatives in Computational Notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–12. https://doi.org/10.1145/3411764.3445527

[90] Mikael Wiberg. 2014. Methodology for materiality: interaction design research through a material lens. *Personal and Ubiquitous Computing* 18, 3 (March 2014), 625–636. https://doi.org/10.1007/s00779-013-0686-7

[91] Mikael Wiberg. 2018. *The materiality of interaction: notes on the materials of interaction design*. The MIT Press, Cambridge, Massachusetts.

[92] Moritz Wittenhagen, Christian Cherek, and Jan Borchers. 2016. Chronicler: Interactive Exploration of Source Code History. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 3522–3532. https://doi.org/10.1145/2858036.2858442

[93] YoungSeok Yoon and Brad A. Myers. 2015. Semantic zooming of code change history. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 95–99. https://doi.org/10.1109/VLHCC.2015.7357203

[94] Young Seok Yoon and Brad A. Myers. 2012. An exploratory study of backtracking strategies used by developers. In *2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE)*. IEEE, Zurich, Switzerland, 138–144. https://doi.org/10.1109/chase.2012.6223012

[95] Young Seok Yoon and Brad A. Myers. 2014. A demonstration of AZURITE: Backtracking tool for programmers. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Melbourne, Australia, 225–226. https://doi.org/10.1109/vlhcc.2014.6883067

[96] Young Seok Yoon and Brad A. Myers. 2014. A longitudinal study of programmers' backtracking. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, Melbourne, Australia, 101–108. https://doi.org/10.1109/vlhcc.2014.6883030

[97] Loutfouz Zaman, Wolfgang Stuerzlinger, Christian Neugebauer, Rob Woodbury, Maher Elkhaldi, Naghmi Shireen, and Michael Terry. 2015. *GEM-NI*: A System for Creating and Managing Alternatives In Generative Design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, Seoul Republic of Korea, 1201–1210. https://doi.org/10.1145/2702123.2702398

[98] Zicarelli.D. 2002. Max/MSP. http://www.cycling74.com/products/maxmsp.html