Imprimer: Computational Notebooks for CNC Milling

Jasper Tran O'Leary
jaspero@cs.washington.edu
University of Washington
Seattle, Washington, USA

Gabrielle Benabdallah
gabben@uw.edu
University of Washington
Seattle, Washington, USA

Nadya Peek
nadya@uw.edu
University of Washington
Seattle, Washington, USA

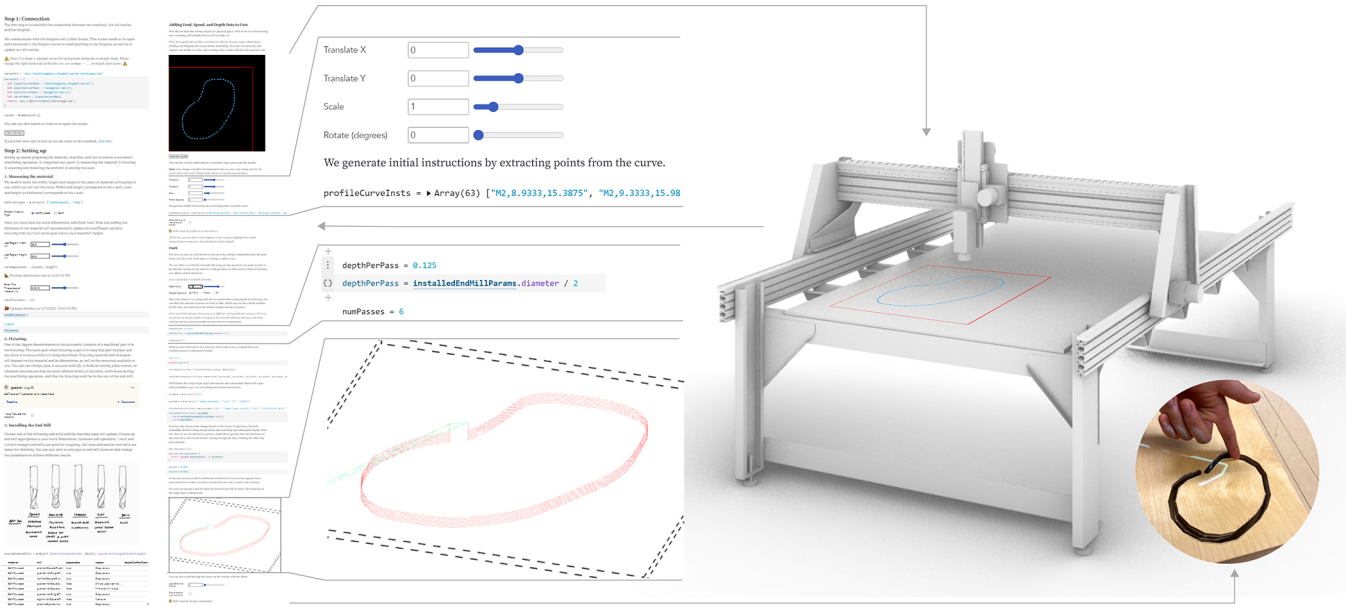


Figure 1: Imprimer Overview. Left: a computational notebook documents a process for setting up and controlling a Shopbot CNC mill. Middle: by writing code, makers create real-time interactive visualizations and inputs to explore machine and control settings. Right: Imprimer projects augmented reality previews onto the Shopbot, enabling testing and refinement of existing cuts.

ABSTRACT

Digital fabrication in industrial contexts involves standardized procedures that prioritize precision and repeatability. However, fabrication machines are now available for makers who focus instead on experimentation. In this paper, we reframe hobbyist CNC milling as writing literate programs which interleave documentation, interactive graphics, and source code for machine control. To test this approach, we present Imprimer, a machine infrastructure for a Shopbot CNC mill and an associated library for the Observable computational notebook. Imprimer lets makers learn experimentally, prototype new interactions for making, and understand physical processes by writing and debugging code. We demonstrate three experimental milling workflows as computational notebooks and conduct a user study with practitioners with a range of backgrounds. We discuss implications for using notebook programming to teach and innovate with CNC milling and to provide a future vision for digital fabrication altogether.

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

ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581334

ACM Reference Format:
Jasper Tran O'Leary, Gabrielle Benabdallah, and Nadya Peek. 2023. Imprimer: Computational Notebooks for CNC Milling. 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, 15 pages. https://doi.org/10.1145/3544548.3581334

1 INTRODUCTION

"[The] learner always gets the experience of interactively controlling the lower-level details, understanding them, developing trust in them, before handing off that control to an abstraction and moving to a higher level of control."
— Bret Victor, *Learnable Programming* [65]

In the last few decades, digital fabrication tools that previously existed only in professional machine shops have become increasingly available to a wider and more varied group of makers [3, 19].

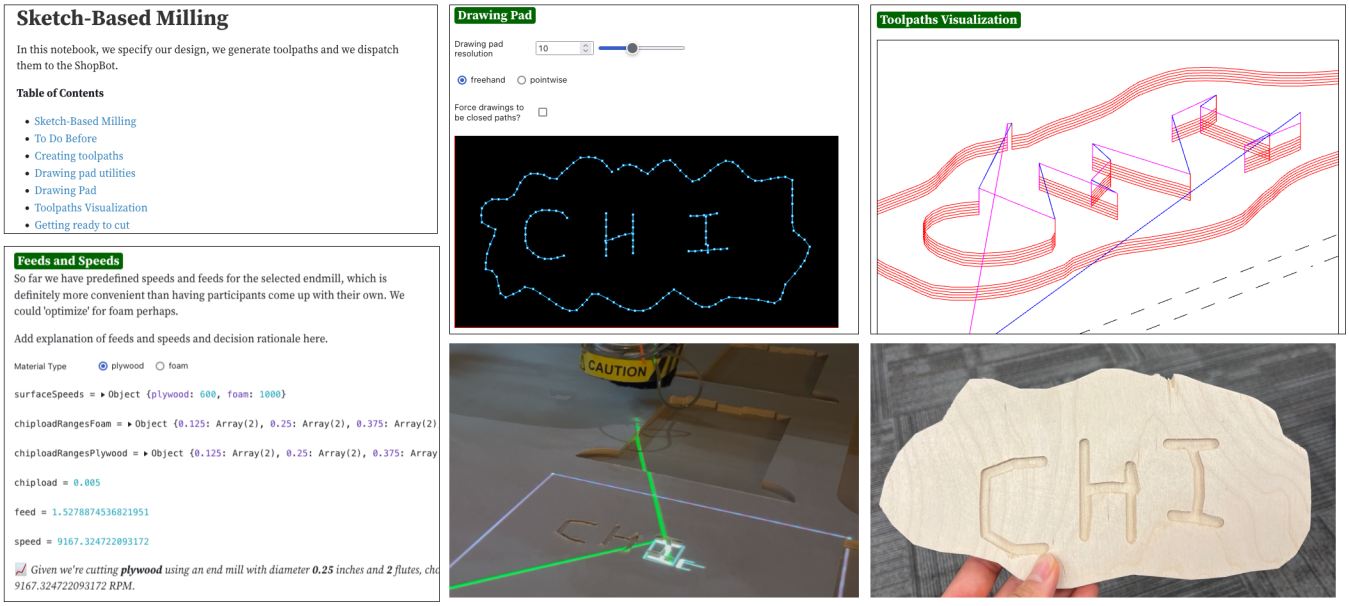


Figure 2: Snippets from the D1: QuickDraw Notebook. Left: the notebook is divided up into separate parts (top), for example, a step for calculating feed and speed values based on the user's tooling and material setup (bottom). Middle: the user can sketch a geometry freehand, by defining control points, or by importing geometry (top) and then visualize the toolpath in-situ using the AR overlay. Right: users can experiment with the cutting depth of the toolpath (top) and dispatch the job to obtain a quickly milled prototype (bottom).

As computer systems moved out of the work place to become increasingly prevalent in domestic, educational, ludic, and aesthetic contexts [6], so did digital fabrication systems reach new sites and groups, creating new priorities and challenges for digital fabrication research. In particular, subtractive manufacturing machines such as CNC (computer numerically controlled) mills—as opposed to additive manufacturing machines like 3D printers—present additional barriers to makers without experience in traditional machine shops, such as safety and tool knowledge. Yet, the existence of digital fabrication systems outside of exclusively industrial contexts indicate a new opportunity: to rethink digital fabrication systems in the light of new priorities, namely expressivity and customization rather than replication and mass production. Artists, designers, entrepreneurs, hobbyists, and other makers prioritize the exploration of new physical forms, the development of original workflows, and the production of bespoke or customized artifacts, rather than the high volume reproduction of a single geometry [26, 64, 73].

We consider the example of a fictional user, Talia. Talia is an artist and product designer who makes ceramic home decorations and objects: drinkware, candle holders, plant pots and accessories. The designs are unique and rely on custom molds that Talia makes herself with a CNC mill. She generates the mold designs in CAD software and then creates toolpaths using a CAM tool, which she then sends to her studio CNC mill. She uses the milled positives to create plaster molds which she uses to create the ceramic pieces.

While her small local business is thriving, Talia is often frustrated with the process of developing new products and designs because of the labor- and resource-intensive nature of experimenting with new designs. For example, one of her best-selling pieces is a mug with a gradient glaze and pronounced ridges generated algorithmically. Talia has spent much time sketching and creating other patterns in CAD and testing them on a mill, but every time she wants to adjust her toolpath or tweak an aspect of the design, she either has to regenerate an entire G-code file in CAM or go back to the model in CAD and go through the CAD-CAM-CNC pipeline again. It is also difficult for Talia to document and share her process with other fabricators at her business because they frequently invent new techniques or improvement to algorithms which require documentation to be constantly rewritten.

In this example, we highlight that the software tools used for CNC milling are locked into a rigid interaction paradigm that focuses on faithful replication, rather than exploration. Even though CNC mills have now become accessible in non-professional spaces (such as fab labs, makerspaces, professional studios and homes), the software tools to control them are still built after the model of this replication-based interaction.

We argue that conventional and hobbyist CAM software tends to discourage makers from "straying from the path" or exploring new designs, as in Talia's example, because they based in a paradigm of *fabrication as executing programs*. In these tools, machining operations are packaged into programs which are relatively easy for makers to execute, but sacrifice the low-level machine control needed to pioneer novel techniques with machines and materials. Low-level machine control allows makers to quickly dispatch commands to the machine, verify operations in real time, and adjust their designs accordingly. As such, it supports improvisational and exploratory fabrication processes. However, writing machine code for direct control of mills and lathes is largely restricted to specific

scenarios in professional machine shops [10, 54]. It is difficult to abstract to more complex interactions; its low-level nature also obscures readability and portability for those besides the author. Moreover, programs written in machine code (e.g., G-code) afford very little documentation and explanation which becomes crucial for less experienced makers.

Altogether, it is difficult to interact with CNC milling machines in an exploratory way. We define exploratory fabrication as an approach to fabrication that focuses on discovering new material behaviors and developing novel workflows rather than the faithful translation of digital models. To achieve exploratory fabrication with CNC mills, makers require control software that allows for low-level machine control that can be quickly iterated on, is richly documented, preserves safety checks, and discourages risky operations.

How can we better support exploratory fabrication with CNC milling machines? To address this question, we argue for a vision that treats *fabrication as writing programs*—not just executing them. The knowledge required for writing programs for fabrication needs to be properly scaffolded. To extend direct machine programming in a supported way, we turn towards *literate programming* where programs are human-readable documents that interleave prose, graphics, and source code [32].

Computational notebooks, a common example of literate programming, aid with digital-only programming problems in data science, machine learning, and related domains [30, 48]. We argue that a literate programming paradigm, if modified properly, could catalyze exploratory fabrication for CNC milling by allowing makers to quickly send commands to the machine, adjust and fine tune their design iteratively, and acquire the necessary programming and manufacturing knowledge to develop their own workflows. Further, literate programming could democratize exploratory fabrication by letting users replicate existing workflows simply by reading and deploying code from a computational notebook.

To test this hypothesis, we present Imprimer, a machine infrastructure for structured, direct control of a CNC mill from a computational notebook—in our case, the Shopbot CNC router [53] and the Observable computational notebook [47]. In Imprimer, makers can prototype new fabrication workflows by writing and modifying code. Our library provides custom visualizations both in the notebook and projected onto the machine in-situ. Makers document their making process by interleaving text and visual documentation. Imprimer blends traditionally separate parts of the fabrication process into a single, live environment where makers can make changes to code, visualize the results, and deploy cutting jobs immediately to the machine. Makers can also view or hide the underlying code for each cell; by hiding all code, makers use the notebook as they would with any other graphical user interface or tutorial document.

We stress that Imprimer is the beginning of a new paradigm: sharing ideas about ways to fabricate, not just running code to fabricate. Thus, it does not yet cover all the functionalities currently available in conventional CNC software tools. Rather, it is a step towards a different interaction paradigm which lets makers discover machine behavior and possibilities gradually, and over time implement their own functionalities by writing code.

To guide the development of this paradigm, we developed three demonstration notebooks that represent exploratory workflows: **D1: QuickDraw**, AR-assisted sketch-based milling (subsection 5.1), **D2: FunctionTile**, surface milling by sampling mathematical functions of two variables (subsection 5.2), and **D3: MiniShelf**, parametric generation of bookshelves (subsection 5.3), alongside two tutorial notebooks that introduce Imprimer's connection and material setup functionalities. We conducted an in-shop user study with participants holding a range of backgrounds in CNC milling and computational notebook programming (Section 6). From these demonstrations and evaluations, we discuss the challenges, rewards, and future possibilities of literate programming as an interaction model for CNC milling.

2 END-TO-END EXAMPLE

To provide a concrete example of using Imprimer, we consider an improvisational milling workflow with Talia, who wants to explore different patterns for ceramic molds. She intends to mill shallow surfaces with various patterns and cast these test pieces in plaster to assess which would work best for cups. This process, which would traditionally take place over several CAD and CAM tools, all happens within a single notebook, enabling rich exploration and quick iteration.

Step 1: Connection. To begin, Talia connects all the different parts of the system—the Observable notebook, the augmented reality overlay and the ShopBot CNC milling machine—over the provided WebSocket. She sends a test command to the ShopBot, "`MX,5`", to make sure the notebook effectively communicates with the milling machine. The ShopBot moves accordingly and the connection is successful.

Step 2: Material setup. Once Talia has established a connection, she goes through a series of steps to prepare the machine and the material for milling: she measures her material (in this case, insulating foam, a cheap and forgiving material for pattern exploration) and fixes it to the machine bed using screws and clamps. Given the thickness of the piece of foam she uses (2 inches), she chooses a long square 1/8th inch end mill and installs it. Finally, she zeroes her axes, and moves back to the Observable notebook to start designing her toolpaths.

Step 3: Overlay calibration. Next, using the AR overlay notebook (subsubsection 5.1.1), she runs through the overlay's calibration procedure to match the projected toolpath to where the machine will actually move. Her material already has some test cuts made in it, but using the AR overlay she is able to position the toolpaths into the spaces on the stock with material remaining.

Steps 4 and 5: Designing toolpaths and milling. After this, Talia uses code from the **D2: FunctionTile** notebook which lets her create 3D surface milling patterns using mathematical functions of two variables. She decides to explore three in particular: sinc, sine/cosine, and the Goldstein-Price functions. With the notebook, she designs her patterns *by designing her toolpaths* and iteratively explores the effect of various machining parameters on her design, such as the maximum cutting depth, the stepover, the frequency of the patterns, the feeds and speeds, among others. With each new swatch, she tweaks a few parameters by making simple edits to the code in the respective cells that calculate these parameters. She documents each variation both in prose and in a dictionary of parameter values in a notebook code cell. Within two hours, she

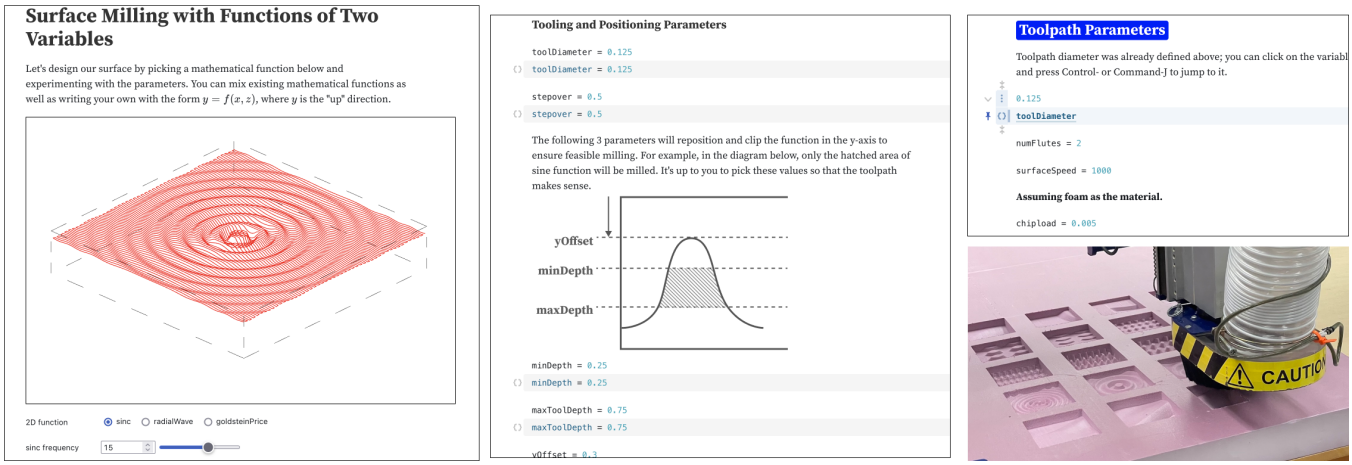


Figure 3: Snippets from the D2: FunctionTile Notebook. Left: users choose a mathematical function of two variables and use graphical elements to experiment with function parameters and see the toolpath visualization update in real time. Middle: to ensure feasible milling, users define variables to offset and clip the function within a desirable bounds as described in a hand-drawn diagram. Right: users specify tooling parameters to calculate feeds and speeds (top) before dispatching the job to the CNC mill (bottom).

has created over 20 swatches with three different functions that she is ready to cast to create her test pieces. Once she is done, she adds notes and comments in the notebook for future reference, and to eventually share the notebook with her community.

2.1 Target Audience: Novices and Experts Alike

Generalizing from Talia's case, Imprimer is a paradigm shift towards using code, visualization, and documentation to pioneer and share knowledge about exploratory fabrication workflows. Users of varying skills in both programming and in CNC milling can engage with Imprimer. Because Observable notebooks can operate with the code completely hidden, those without programming experience can still use the notebooks with full functionality apart from code-level customization. In parallel, while there are increasingly sophisticated tools for learning programming, there are relatively few innovations in digital fabrication education. Programmers who lack CNC milling experience can follow Imprimer notebooks that explain the process where documentation is interleaved with the controls; this contrasts with conventional CAD/CAM tools where documentation is separate.

On the other hand, users who are experienced in both programming and CNC milling can remix notebook code and develop entirely new functionality. Unlike conventional CAD/CAM tools, the source code that drives an interaction can be readily tailored, from simple customization of toolpath algorithms, to custom forks of notebooks, to entirely new notebooks representing new workflows [45]. Imprimer's library and direct machine communication helps experienced developers focus on and test their high level goals. Finally, users of all backgrounds can benefit from Imprimer's free and open source nature, whereas common CAM software can cost up to thousands of dollars per year.

2.2 Sustainability

Exploratory fabrication unavoidably involves creating many iterations of an artifact. Such iteration can be wasteful, especially when running the machine would consume a larger amount of material than strictly needed to test concerns such as tolerances or experimenting with feeds and speeds. Through Imprimer, users can write code and documentation to explicitly produce less wasteful test pieces through code; the advantage is that test pieces can evolve alongside code that produces the final artifact. In addition, by enabling visual debugging, custom visualizations can help reduce the amount of iterations that require machine time at all.

3 RELATED WORK

Imprimer expands on a body of work on programming in fabrication emerging in academic and hobbyist settings. Imprimer fosters direct machine control and lets makers develop their own interactions in a supported way via literate programming. In this section, we review contributions in digital fabrication research that propose dynamic interaction paradigms alongside research on computational notebooks for data science applications.

3.1 Conventional CAM in Maker Settings and the Alternative

Traditional CAD and CAM tools proliferate in makerspaces within academic and hobbyist contexts. For CNC milling, popular CAD-CAM software choices include Autodesk Fusion 360 [4] and Solidworks [11]. While these tools have features that make them more readily usable by makers, they operate under a CAD-to-CAM paradigm which treats CAM as a translation step of digital designs, rather than as a space for creative decisions. This presents a large barrier to exploratory fabrication and to newer makers in general, as suggested by Hudson et al. [24]. This is especially true for CNC

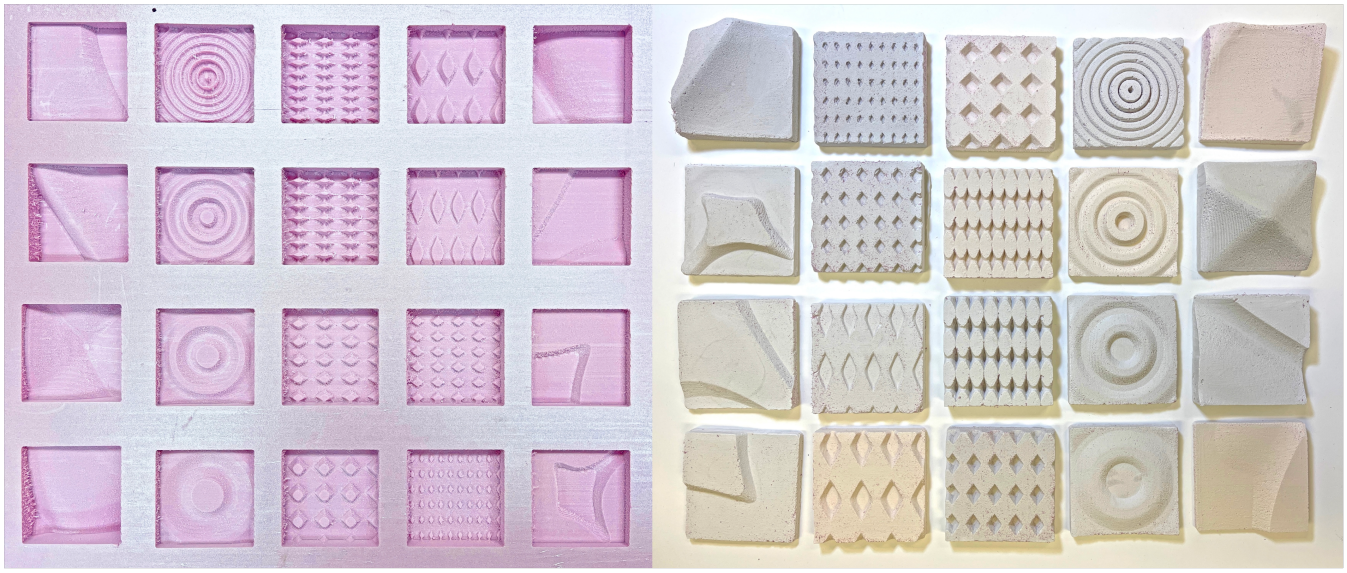


Figure 4: Fabricated Tiles Modeled with the D2: FunctionTile Notebook. Left: a block of molds generated from several mathematical functions of two variables (sinc, sine/cosine, Goldstein-Price) milled in insulation foam. Right: the corresponding tiles cast in plaster.

milling, which requires extensive knowledge of materials, tooling, and toolpathing to use CAM software at all.

Our goal is to facilitate the use of CNC as a medium for creative exploration, besides its current use as a tool for translating digital designs to physical form. In a study of makers' current desires for fabrication tools, Yildirim et al. [71] reported that "participants envisioned future [tools] that could improve their current workflow by leveraging new machine capabilities. They desired systems that have personal and collective awareness for additional support throughout the whole process." We can draw inspiration from key tenets of Bret Victor's notion of learnable programming [65] to imagine *learnable fabrication*: explaining in context, following the [program's] flow, seeing the state, creating by reacting, and creating by abstracting. Rather than constrain makers into using abstractions developed for industrial contexts, we seek to give makers control over low-level movements along with the means to build and document their own abstractions.

3.1.1 Existing Practice for Directly Writing Machine Instructions. In the case of exploratory fabrication, directly writing machine instructions is constrained by the lack of tools that support this practice and is therefore limited to *ad hoc* solutions. For instance, Desjardins and Tihanyi [13] wrote G-code in an Excel sheet to create the ceramic cups toolpaths, which the authors then copy-pasted in a text editor to be sent to the Potterbot7 ceramic 3D printer. In the case of CNC milling, we can look at the example of the AESTUS vases by ODK.design, which feature milled grooves made by a robotic arm that the designer Oliver David Krieg created by developing custom CAM software to generate the toolpaths [33]. In both these cases, the designers needed better control over the generation of toolpaths in order to realize their design intent. They developed their own approach and tool, whereas a system like Imprimer would have enabled them to directly author the toolpaths while retaining generative capabilities and enabling documentation.

Given that Imprimer builds on existing practice, we would expect machinists to appreciate Imprimer's functionality because it would allow them to still directly visualize and use G-Code with the added benefit of interleaved documentation and the ability to use conditionals and definite functions to encapsulate common user-defined behaviors. Because it tends to focus on high-level usability, research at the intersection of fabrication and HCI has seldom approached low-level authoring of machine instructions. With Imprimer, we are extending machinist *and* inventive practices of low-level machine control in a supported way so that it can benefit both experienced and non-professional makers.

3.2 Interaction Techniques for Digital Fabrication and CNC Milling

Prior research in HCI has explored new paradigms for controlling digital fabrication machines. During machine operation, interactive fabrication [69] and continuous fabrication [43] creatively map real-time input to machine output. Kim et al. [28] presented compositional fabrication, which enables interactions for tuning. Hybrid [74, 75] and lucid [59, 60] fabrication also provide meaningful augmented feedback and user control. While all of these techniques promise increased human agency during a machine-centric process, they all focus on human input during machine operation. In contrast, other parts of the fabrication process, namely, material choice, machine tooling, visualization, and responding to errors, are less explicitly explored by these paradigms. Imprimer's goal is to support the inherent iterative process of making: fabricating an object, examining it for issues, and tracing results back to the control system. We argue the easiest way to do this is by not hiding the

code that determines machine control, rather empowering makers to steer the process through writing code.

In particular, compared to interactive systems for 3D printing and laser cutting, relatively few systems examine CNC milling. Li et al. [37] built a system that provided immediate engraving from drawings. Tian et al. [61] used a GUI and specialized CNC for improvisational carving of joinery, while Larsson et al. [34] provided a pipeline to explore and fabricate a large space of voxel-based joints. Follmer et al. [16] and Weichel et al. [67] used 2D/3D scanning and milling to prototype copy-and-paste interaction and bidirectional fabrication, respectively. Teibrich et al. [58] and Mueller et al. [42] used milling to selectively destroy parts or all of existing objects. Saakes et al. [52] optimized placement of new cuts given existing stock material and Müller et al. [44] used computer vision to help makers take in-situ measurements. As a whole, these systems deal with singular fabrication pipelines or parts of the process; they focus less on making decisions about machine and material setup or toolpath generation. In contrast, Imprimer's goal is to open the space of interactions while not unduly abstracting away interconnected machine factors.

Beyond new interaction paradigms for live machine control, many researchers have explored novel ways of programming machines. Many systems (e.g. LabView [5] and Pure Data [49]) have leveraged a flow-based programming paradigm to control machines. In particular, Grasshopper for Rhinoceros [50] has enabled control for new fabrication applications like bioprinting [66] and 4D printing [57]. Notably, Fossdal et al. [17] extended Grasshopper to provide real-time control of machine from a CAD environment, eliding conventional distinctions between CAD and CAM. With this level of control, other work has explored computer-mediated improvisational making with quilting [35], weaving [1], and multi-machine fabrication [21]. Li et al. [36] and Yu and McCann [72] provided tools for mapping output back to relevant parts of the program in digital drawing and knitting, respectively. Lin and colleagues explored languages for PCB design that negotiate trade-offs between high level goals and low-level circuitry [39, 40] while Tran O'Leary et al. [63] modeled machines and machine interaction in a unified grammar.

Of particular note are tools that have made low-level G-code programming more accessible beyond a machine terminal. Subbaraman and Peek [55] presented p5.fab, a creative coding environment with a Javascript wrapper of G-code that facilitates experimentation with machine parameters. Relatedly, Gleadall [20] built a system for fine-grain authoring and previewing low-level primitives and G-code in Excel. Imprimer is similar to these works in that it provides unconstrained means of generating low-level G-code versus conventional CAM, but seeks to do so in a supported way through the rich documentation afforded by literate programming. This documented programming approach is particularly well suited to CNC milling, as it requires more precision and domain knowledge than additive manufacturing techniques.

3.3 Literate Programming, Computational Notebooks, and the Physical-Digital Divide

To our knowledge, Imprimer is the first to control CNC mills through computational notebooks, which are typically used for drastically different applications like data science. Compared to the myriad of existing programming techniques for digital fabrication that we have reviewed above, why might computational notebooks be uniquely suited for exploratory CNC milling—and for machine control more broadly? Our key insight is that makers can best understand machine behavior by writing code, prose, and visualizations in tandem. Literate programming, as proposed by Donald Knuth [32], puts forth a vision where source code can be flexibly rearranged to complement documentation in natural language, and, more recently, with graphics, visualizations, and other multimedia. We argue that computational notebooks offer a medium to quickly explore not only datasets, but also physical machine behaviors and interaction techniques.

Yet, even within data science, computational notebooks are still evolving programming environments. Chattopadhyay et al. [9] discuss the limitations of computational notebooks by identifying nine pain points data scientists face when using popular computational notebooks such as Azure or Jupyter [31]. Among these pain points is "Reproduce and Reuse," where the authors discuss the challenges data scientists run into when trying to reproduce or adapt existing notebooks. Rule et al. [51] analyzed 1 million notebooks on Github and found that a quarter contained no explanation, revealing a tension between exploration and explanation. In response to this Head and colleagues proposed systems to organize existing notebooks into "cleaned up" slices of analysis [22] or flexibly organized tutorials [23]. The Observable notebook in particular addresses some of these problems by providing a live, topological runtime environment that permits notebook cells to be arranged in any order [7]. We argue that these techniques for organizing notebook code could also help makers experiment and iterate quickly on novel milling workflows.

Other research on computational notebooks in HCI has examined more powerful interaction techniques. Kery et al. [27] and Wu et al. [70] developed direct manipulation techniques to generate code by manipulating visualizations. Weinman et al. introduced forking and backtracking to let notebook users explore alternative approaches [68] and Drosos et al. have presented a Jupyter notebook extension that supports data wrangling [14]. DeLine et al. developed Glinda, a domain-specific language for data science workflows that supports live programming with interactive results [12]. Overall, these works have explored and addressed issues with notebooks for data science; in contrast, Imprimer extends notebook programming to the physical application of CNC milling.

4 INTEGRATING MACHINES AND COMPUTATIONAL NOTEBOOKS

To provide the foundation for Imprimer, we built a network infrastructure to afford direct control of a CNC mill from a computational notebook environment. To support the creation of workflows, we contribute a library of functions for a live computational notebook that facilitate: connecting to and controlling the mill, navigating tooling options, visualizing geometries and toolpaths, and more.

4.1 Machine Network Infrastructure

To support this workflow, Imprimer comprises a web server that coordinates communication between desktop application that relays

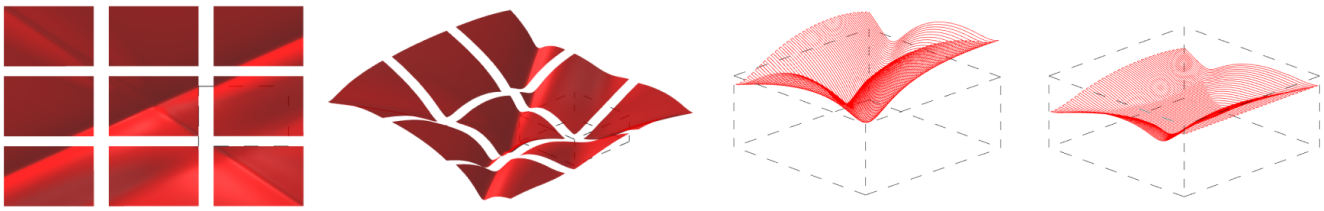
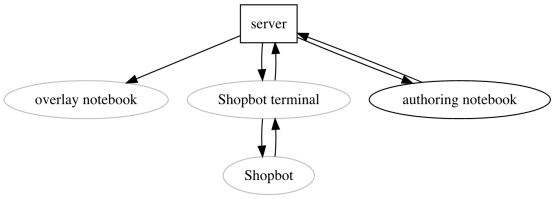


Figure 5: Milling Subsets of the Goldstein-Price Function in the D2: FunctionTile notebook. 1) Top-down view of a surface rendering of nine subsets surrounding the global minimum of the Goldstein-Price function. 2) An isomorphic view of the same rendering. 3) Toolpath rendering of the single subset containing the global minimum (center of the nine), currently infeasible to mill due to the toolpath lying almost entirely above the material. 4) After interactively processing the toolpath by dividing the function output, adding a vertical offset, and clamping the function to a desired bounds, the toolpath becomes feasible to mill into a mold.

information to a full size Shopbot PRSalpha with a 96x60" cutting bed[1] and a projector-based augmented reality interface called the *AR overlay*. The Shopbot accepts both its native SBP instruction set alongside the more commonplace G-code. Figure 6 illustrates the relationship between the user's current notebook ("authoring notebook"), the server, and a desktop computer that communicates over a wired connection to the Shopbot ("Shopbot terminal"). As is done in our tutorial notebook on connecting to the machine, we can generate Figure 6 representing the network architecture using the code that follows the figure.




```
dot`digraph "networkGraph" {
    server [color=${connectedNodes.server
        ? "black" : "gray"}, shape=box]
    // More styling info omitted

    "authoring notebook" -> server
    server -> "overlay notebook"
    "Shopbot terminal" -> server
    // ...
}
```

Figure 6: Visualizing the status of the machine-notebook network. `connectedNodes` is a mutable cell that is updated with packets from the server whenever a client connects or disconnects, recoloring the diagram appropriately

To connect the notebooks, we implemented the network using a WebSocket interface building on the code used by Li et al. [37], which is publicly available. In particular, we built the Shopbot terminal and server to facilitate direct interaction between multiple notebooks and with the machine. We installed the terminal on the computer connected to the Shopbot and deployed the server code in the cloud. This direct notebook-to-machine job dispatch is not a typical functionality for the Shopbot, which requires exporting and importing files, even though similar functionality exists off-the-shelf for hobbyist 3D printers [15].

[1]The Shopbot's default unit of length is inches rather than meters.

4.2 Implementation in Observable Notebooks

To control Imprimer's machine infrastructure, we implemented a library for Observable, a browser-based computational notebook popular for data science applications. Observable runs a modified version of Javascript, letting it interoperate with many existing technologies. The notebook differs from other computational notebooks like Jupyter Notebook (Python) or RMarkdown (R) in that code cells run in *topological order*, meaning that cells are automatically recomputed whenever any cells they depend on (*parent cells*) are themselves recomputed. This produces a live programming [56] environment where changes in one cell propagate throughout the notebook; graphical input, visualizations, and underlying data are always up-to-date, affording quick and interactive iteration. This contrasts with other notebooks whose code is run in a linear order or must be manually re-run; rather, live topological allows us to arrange code in the best order for fabrication workflows and teaching—a key tenet of literate programming [32].

4.2.1 Live Cells and Cell Names. All parts of an Observable notebook are contained in cells which written Markdown, HTML, or Javascript. For example, we represent the following cell that computes a value and associates it with the cell name `stepdownPercent`, representing the proportion of the end mill's diameter that the mill advances downward per cutting pass.

```
stepdownPercent = 0.5
```

```
stepdownPercent = 0.5
```

Figure 7: Declaring a cell variable.

Whenever the the cell `stepdownPercent` is recomputed, for example, if the maker sets a new percentage, the notebook recomputes following cell which calculates the number of cutting passes required to cut through through the material.

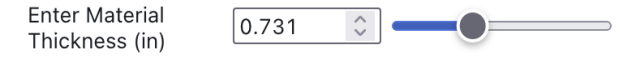
```
numPasses = 6
```

```
numPasses = {
    let stepdown = stepdownPercent *
        installedEndMill.diameter;
    return Math.ceil(thickness / stepdown);
}
```

Figure 8: A child cell reactively computes results when a parent is recomputed. `installedEndMill` and `thickness` are themselves cell values defined elsewhere in the notebook.

Cells can also be Markdown or HTML, so notebooks contain diagrams, videos, and prose that instruct the maker about various considerations while programming for CNC machines, as shown in Figure 2, Figure 3, and Figure 11.

4.2.2 HTML Templating and Views. Cells can also return HTML that is generated by Javascript, affording custom visualizations called *views*. A view has two parts: the view itself, which is typically an interactive DOM element; and the value, which is any JavaScript value. For example, consider the following view from the authoring notebook:




```
viewof thickness = Inputs.range([0.0625, 2], {
  label: "Enter Material Thickness (in)",
  step: 0.0001
})
```

Figure 9: A slider view for setting material thickness.

The result of the right-hand side of the cell evaluates to DOM element which is rendered above the cell. But, by using the `viewof` keyword, the value of `thickness` is instead the *current value* of the DOM element—that is, whatever the notebook user has selected. Views can be arbitratily complex because their appearance and interactivity can be programmed in HTML and/or Javascript. Imprimer builds extensively on this functionality to provide intuitive interfaces for different stages of CNC control.

4.2.3 Bringing Machine State into the Notebooks. Imprimer's library provides a streamlined interface to synchronize notebook state with physical machine state. We configure the Shopbot terminal to periodically emit packets containing the Shopbot's current state to the authoring notebook through the server. For example, whenever the end mill changes position, the Shopbot terminal detects this change and sends the new position in a packet to the server. On the notebook end, we keep track of the machine state for library functions to query.

5 DEMONSTRATIONS

To explore the versatility of Imprimer, we implemented three demonstration notebooks: sketch-based milling with augmented reality (**D1: QuickDraw**), surface milling molds by sampling functions of two variables (**D2: FunctionTile**), and a parametric shelf generator with associated debugging tools (**D3: MiniShelf**).

5.1 QuickDraw: Sketch-Based Milling with Augmented Reality

Following the thread of work on direct drawing with CNC machines introduced by Li et al. [38], we built QuickDraw, a notebook that lets users quickly sketch geometries to engrave or cut through, visualize the resulting toolpaths in-situ on the physical material using an augmented reality overlay, make adjustments, and directly mill the job from the notebook. The goal of QuickDraw is for users to quickly prototype low-fidelity versions of more complicated form they might want to test later, or test out the effects of different end mills on material finish. Assuming the machine's tooling and material are set up correctly, and that the AR overlay has been calibrated, users can walk up to the notebook with no prior design file, sketch a quick concept as in Figure 2, and mill it out completely in around ten minutes. Once satisfied, users can also import and process existing 2D geometries, create 3D toolpaths while experimenting with scale and cutting depth, and check the toolpaths using the AR overlay. They can then move into other notebooks to work at a higher fidelity. We defer importing 3D geometries for future work.

5.1.1 Augmented Reality Previews. We sought to extend frequent visualization of data beyond the notebook environment into physical space. To this end, we installed a projector above the bed of the Shopbot that projects toolpath visualizations directly onto the machine's bed. This provides an *AR overlay* which can project 2D views onto the material in-situ; users must flatten 3D views into 2D before rendering the with the overlay by calculating a 2D projection or "slice" onto a desired cutting plane. The visualizations are rendered by a separate notebook which is connected to the network and listens for updates from the authoring notebook, for example, any changes in the toolpath's instructions or in machine parameters that affect the toolpath. We include a calibration routine that the maker performs when setting up their stock material to ensure that the visualized path matches the end mill's physical location as proposed by Tran O'Leary et al. [62]. This routine involves the maker dragging four projected points to match the ground truth corners of their stock material; the notebook then computes a homography from the mapping of points and uses the homography to transform all elements of the visualization to match the toolpath's true physical location.

5.2 FunctionTile: Surface Milling Tile Molds by Sampling Functions of Two Variables

One of Imprimer's strengths is that notebook authors can combine in a single, documented programming environment parts of a fabrication pipeline that would otherwise be split across applications. In this demonstration, we created a means of surface milling molds for plaster tiles based on forms derived from mathematical functions. **D2: FunctionTile** includes a visualization for both the surface of a function of two variables alongside a derived toolpath that can be dispatched to a CNC mill (Figure 3). For example, we implemented the two-variable unnormalized sinc function

$$z = \frac{\sin(f \cdot \sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$$

where f is the frequency or "waviness" of the function, shown in the cells in Figure 10, which results in the plot in Figure 3 and a slider which recomputes the plot in real time.

```
sinc = f(x, y)
```

```
function sinc(x, y) {
  let r = Math.sqrt(x ** 2 + y ** 2);
  let z = Math.sin(sincFreq * r) / r;
  return THREE.Vector3(x, y, z);
}
```

sinc frequency 12

```
viewof sincFreq = Inputs.range([0, 24])
```

Figure 10: Defining the sinc function with a slider for experimenting with frequency.

We provided three example functions of two variables: the sinc function, a sine/cosine function, and the Goldstein-Price function [41]. Users can modify these functions or write their own functions in code as needed.

We then sample points across the surface of the plot and connect them to generate a toolpath. Several decisions greatly affect how this toolpath is generated, for example: whether the path passes row-by-row or column-by-column (e.g. with anisotropic materials like wood with a grain), the stepover which controls how much of the end mill's diameter the mill will remove per pass, and the z-offset at which the surface is milled relative to the top of the material. These toolpathing decisions, alongside the mathematical parameters of the chosen function, are all interconnected and affect the form and quality of the milled mold. For example, very thin parts of the surface could break depending on the material, stepdown, and chosen feed and speed.

With conventional CAD and CAM, it is possible to create similar molds, but a user would have to work across several programs, changing parameters in one program before exporting and importing to another. Even programmatic tools such as Grasshopper plus RhinoCAM [8] which support generating 2D surfaces still require back-and-forth between the Grasshopper window ("programming side"), the Rhino interface ("CAM side"), and the RhinoCAM options interface ("CAM side") where the user has to know in advance how changing one parameter affects the design. In contrast, Imprimer lets the notebook author quickly visualize the effect of each parameter change on the toolpath and document the flow of data. The notebook exposes both input elements for experimentation and the underlying source code, making it easier for users to follow along without invested expertise in experimental CAD/CAM or programming tools.

5.2.1 Translating Abstract Concepts into Millable Forms. Among the three example functions, the Goldstein-Price function proved especially challenging to translate into something that could be physically milled. This is largely because the function was designed to test optimization algorithms and has volatile behavior spanning several orders of magnitude around its global minimum. We were particularly inspired to explore the function as a creative inspiration when one participant in our user study (Section 6) wanted to have additional views that showed how the function could be translated into a millable mold. If we naively map part of the function to the space occupied by the tile mold, we risk "squashing" much of its the sloping topology. To address this, as Figure 5 shows, we sampled several subsets of the function around its global minimum. We then defined a graphical input so that a notebook user can pick one subset at a time and experiment with a division term and vertical offset to fit just a subset into a mold space while preserving as much detail as possible for the given subset. They then repeat the process with other subsets to produce a set of tiles that together show the topology of the function.

5.3 MiniShelf: a Parametric Shelf Generator with Associated Debugging Tools

MiniShelf is a lightweight parametric toolpath generator for fabricating shelves that includes tools for troubleshooting the fabrication process. MiniShelf accepts as parameters: the height, width, and depth of the unit, the number and spacing of shelf spans, and the thickness of the stock material. For now, the selection of joints are fixed, though this could be easily extended in the future; we use assembly-friendly rabbet and groove joints to respectively connect the sides of the shelf case and to connect the shelf spans to the case. In the MiniShelf notebook, we implemented from scratch each part of the fabrication process—from machine setup, to material selection, to parameter selection, to toolpath generation, to visualization, to job dispatch.

Upon first implementing this workflow, fabricated shelves did not yet fit correctly, and we relied heavily on Imprimer's features to debug our own process. While multiple things can go wrong while exploring a new fabrication workflow, by expressing the workflow in code using Imprimer, we could debug our process by debugging code. As a result, we not only contribute a parametric pipeline, but also an associated set of features that we used to debug the process which we have incorporated into Imprimer's library which we describe below.

5.3.1 Debugging Incorrect Logic in Toolpath Algorithms. To create rabbets and grooves in the shelf pieces, and to cut out all the pieces to size, we wrote all the necessary toolpath generation algorithms by hand. Writing bespoke toolpath algorithms allowed us and future users of the notebook to readily generate just the right toolpaths for fabricating the shelves, rather than having to adapt output from a conventional CAM interface. Naturally, while writing the algorithms, we implemented buggy functionality, for example, generating the rabbets in the wrong location. We were able to catch many of these issues using the top-down view (Figure 11, right). Though, some issues could not reasonably be identified until runtime; for example, by not "overshooting" the rabbet toolpath over the designated region, we were left with artifacts where the mill could not reach (Figure 12, left). We could have addressed these issues by implemented dog bone fillets [46], though, in our case,

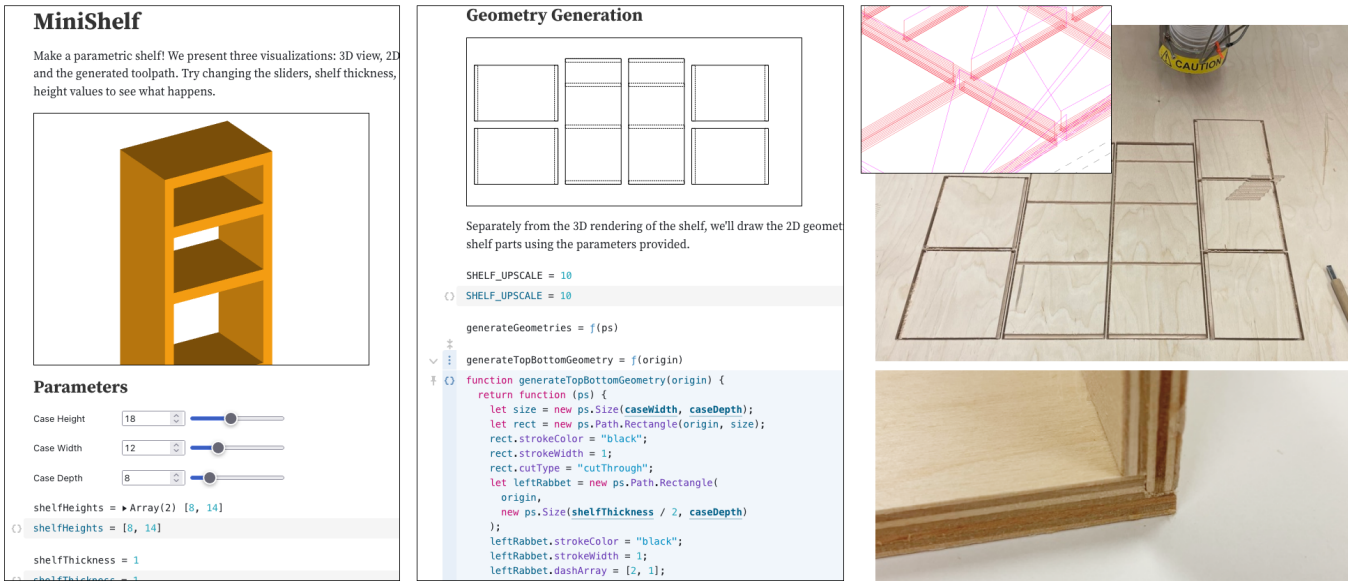


Figure 11: Snippets from the D3: MiniShelf notebook. Left: notebook users can experiment with different shelf dimensions and see a 3D rendering of the assembled shelf update immediately. Middle: when any shelf parameters are changed, the notebook generates new planar geometries of the shelf parts as defined in notebook cells which users can edit if desired. Right: A generated 3D toolpath (top) which can then be dispatched to a CNC mill (middle); rabbet joint parameters can be adjusted to achieve a tight fit (bottom).

we chose to just overshoot the rabbet toolpaths by editing the algorithm and left a note about this design decision in an adjacent cell.

5.3.2 Tolerancing with Joints. An inevitable issue with getting joints to fit together is fabricating both parts within an acceptable tolerance. While conventional machining practice uses rigorous techniques like geometric dimensioning and tolerancing to systematically address these issues [2], these techniques are generally too heavyweight for most hobbyist applications. In our case, to debug issues of getting joints to fit together, we wrote a function that takes in machine and maker-set parameters in the notebook and generates small testing pieces (Figure 12, middle). Once we found the optimal milling depth for the joints, we set it as a cell variable in the MiniShelf notebook.

6 USER STUDY

To better understand how users conceptualize literate programming as a technique for CNC milling, we conducted a user study with 6 participants with varying levels of experience in CNC milling and in programming. We recruited participants from professional connections as well as from a makerspace email list. Participants' CNC backgrounds ranged from no prior experience to those experienced with advanced CAD, CAM, and manual woodworking practice. Similarly, participants' programming skills ranged from passing knowledge to high expertise in scientific computing and computational notebooks.

Specifically, P1 is a researcher who studies computational notebooks, P2 is a materials scientist who has used a Shopbot for a digital fabrication course, P3 is a professor with background in CAD and computational notebooks, P4 is a user experience designer with background in CAD and CAM for mechanical engineering, P5 is a teaching professor who uses computational notebooks for teaching control theory, and P6 is a student who uses CAD and 3D printing to design custom camera components. P3 and P5 also each have over five years of experience with manual woodworking.

Rather than focus on pure usability, our goal for this user study was to better understand how users experienced Imprimer as a novel paradigm. During a 2-hour in-shop session, we asked each participant to walk through two tutorial notebooks to connect to the Shopbot and set up the machine and the material. Next, participants used either the **D1: QuickDraw** (subsection 5.1) or **D2: FunctionTile** (subsection 5.2) notebooks to mill a sketch or tile[2]. We paid particular attention to moments of learning, pain points expressed, and code reading and tailoring. After completing the study, we conducted brief semi-structured interview where we asked participants on their experience using the notebooks, how they learned to use the system, and improvements they would like to see implemented

6.1 Participants Brought Diverse Fabrication Goals to the Study

Participants came from varied backgrounds and brought their own desires for CNC milling to the study. These included creating custom furniture from precisely specified 2D geometries, rastering and engraving images, and teaching others. Because Imprimer is a prototype for a new paradigm for controlling CNC mills, participants

[2]We did not ask any participants to use the MiniShelf notebook (subsection 5.3) due to the relatively long milling time required.

understood that it did not yet have the all of the functionality of more established tools, but were willing to work with the limitations. For example, one participant enjoyed using the FunctionTile notebook's sliders to experiment with tile forms, but wanted additional functionality beyond defining mathematical functions for modifying the surface.

> **P6** I think for many applications, you wouldn't want to use math at all; there are only so many 2D functions that look interesting. I would imagine having [direct manipulation] like with Rhino where the user could have control points and edit the surface that way.

We plan on adding control points to 3D surface views as future work. Another participant wanted to be able to specify geometric dimensions in a programmatic way, which we had only implemented as low level functionality and not yet as a full-fledged API in **D1: QuickDraw**. Nonetheless, P3 praised the fact that notebooks could be extended later on and that functionality could be imported or swapped out, remarking:

> **P3** Overall, besides [not having the API], the concept is neat and I see a lot of applications. It seems if you can swap out the toolpath generation part to match an application—as long as you could write [code for] toolpath generation—then there's a lot of real world use. Using a notebook feels like a more natural way to interact with a machine.

6.2 Code Became Crucial in Understanding Exploratory Milling Processes

All participants, even with those with less experience programming, navigated code cells in the notebooks, often exploring, reading, and experimenting with code from different parts of a given notebook. In some instances, participants stated that they wanted to do things differently than we had already written them in the notebook and proposed their changes by speaking about them in terms of code.

For example, as a matter of preference, P2 wanted to cut a little deeper than the thickness of the stock material into the machine bed to make sure that there were not any artifacts left over. To do this, they created a new cell, `epsilon = 0.01`, to define an additional value to add to the last cut-through pass. P2 then used calipers to visualize one thousandth of an inch physically and then realized that it was too small to be perceptible practically speaking. They then manually edited the `epsilon` cell to be one sixteenth of an inch, i.e., `epsilon = 0.0625`. When asked why they picked this value, P2 reported that they had regularly worked with such increments in the past.

Introducing code to the act of CNC milling also prompted to participants to think about how they might learn either basic CNC functionality through the notebooks, or how they might navigate more experimental workflows. Some participants preferred *writing code* in a scaffolded manner to understand what they were implementing.

> **P4** [For learning] if I started from scratch, I'd think about the different things I could make, the different materials. I would have liked to have written the notebook I'm using myself.

Other participants preferred *reading code*, particularly when exploring novel forms in the **D2: FunctionTile** notebook

> **P5** I like having text in front of me. I like to start with having many examples, I like to remix what's already there ... Seeing what's already written helps me understand what's possible at all, versus starting with a blank slate.

6.3 Negotiating Learning and Making within Notebooks

We noticed that presenting a fabrication workflow in a computational notebook format often uncovered two competing goals for new users: the need to understand how Imprimer and the CNC machine itself work versus the need to actually dispatch jobs and try things out. P1 in particular had a great deal of background in computational notebooks but less in CNC milling, and commented about how CNC-specific practice would need to be introduced and discussed before getting to the "making" parts of the notebook. For example, while physically installing an end mill, P1 remarked

> **P1** Are you recording what we're doing now? Because that could be helpful for somebody installing the end mill. ... So much information like 80%, 20% of the shaft in the collet—I'm not sure how bad it is if one thing is done wrong versus another.

We discussed with P1 this tension between walking through steps specific to CNC milling versus, in their words, "just getting things done." They suggested that we might structure a series of notebooks in a tutorial-and-reference style akin to programming language tutorials. The tutorial notebooks would then link to reference notebooks on topics for makers might who more information—for example, an entire dedicated notebook to understanding what end mills are, how to install them, and how to program in a notebook to account for their effects. To this end, we extracted the "how to" parts from the **D1: QuickDraw** and **D2: FunctionTile** notebooks and placed them in dedicated tutorial notebooks.

6.4 Utility of Custom Views versus Graphical Input Elements

In both the **D1: QuickDraw** and **D2: FunctionTile** notebooks, participants frequently used visualizations to understand how their input would map to toolpaths. We found that, in general, participants preferred to debug their toolpaths by making small adjustments in parameters such as stepover, tool choice, or their choice of sketch and subsequently inspecting the results in a visualization. Some participants found that being able to bridge mathematical renderings with CNC-specific toolpath visualizations in the same visual space unlocked new forms of interaction.

> **P3** This is like when I first got Mathematica, I wanted to see all sorts of things that it would plot. I would just edit the code—is that something I can do here? Okay then! I would divide the denominator with another factor of r ... Is that two times symbols in Javascript? Also I might add a phase shift right in the sine function to get more activity in the middle of the material.

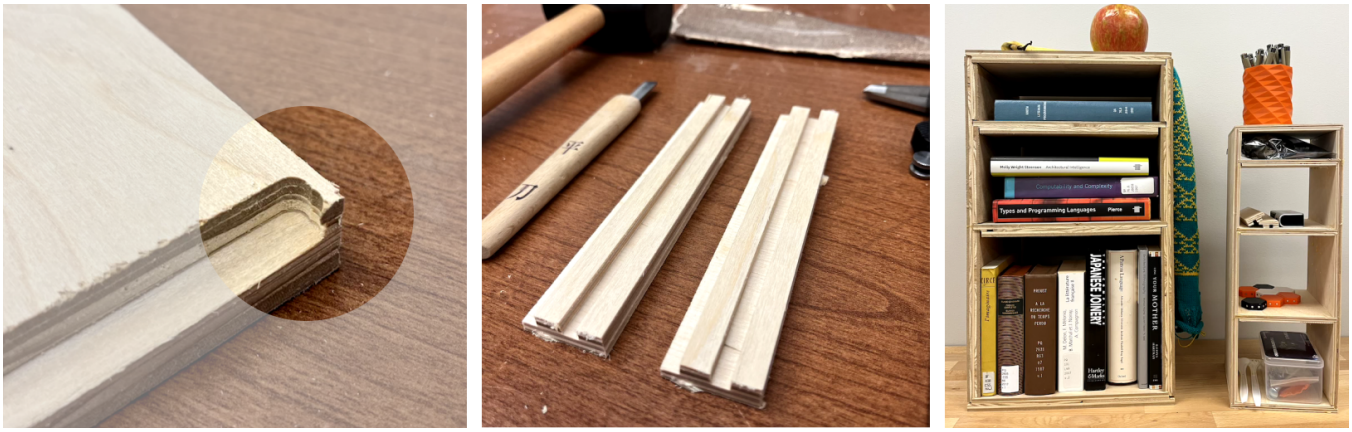


Figure 12: Moments from Developing MiniShelf. Left: a miscalculation in generating the rabbet toolpath left artifacts in the corners. Middle: parametrically generated test pieces to check the cut depth of the joints. Right: two shelves each fabricated with MiniShelf.

Conversely, visual *input* elements often confused participants, who saw code-generated input elements as conceptually separate from other code. This differs from other notebooks like Jupyter Notebook which rarely feature code-defined input elements; in the context of CNC milling, such elements could present more of a cognitive gap than they do with purely digital tasks. Instead, while participants used output visual elements without question, they generally preferred to write raw cell variable values that did not obscure underlying code with graphical controls.

> **P4** How do you even create visual elements, and how do they work? Sliders don't make sense to me. It's difficult to understand the code that generates them. I would want to key in important values.

P3 further remarked that anything that took their attention away from raw code, even though the input elements were themselves defined in code, detracted from their experience of CNC milling through programming.

> **P3** To me, input elements defeat the purpose of the notebook. I want to see the code, and I want to edit things myself. When it comes to widgets, that's something that you might want your boss to play around with, but the scientist needs to be working with just the code.

6.5 Scaffolding and Sharing Experimental Milling with Others

However, in contrast to other participants, P5 saw great value in graphical input elements because of their potential value for scaffolding and sharing knowledge about novel production processes with others. P5 cited their background as an electrical engineer and engineering educator in how they interacted with the Imprimer notebooks. In particular, they mentioned how computational notebooks such as Jupyter Notebook and Google Colab helped their students collaborate around projects in control theory, and how such code needed to be curated and documented.

> **P5** I really like that you can write code that creates buttons and sliders. This way you could limit the notebook to specific functionalities, like you can adjust and play with these numbers, but not these other ones that might be more dangerous to mill. Like you'd say "I only want you to change this number from 1 to 10." Like, well you could [edit more] if you went into the code, but most students would stick to the controls you give them.

P3 echoed the value of interleaved documentation, saying "I use Jupyter notebook to teach my classes—having tons of lines of prose for just one line of code is very helpful." In addition, P5 noticed which code cells were and were not pinned open to always show their code implementations rather than collapsing the code after editing. While we had originally considered pinning to be a superfluous detail, P5 insisted that proper choice of what code to show and hide was crucial for allowing others to experiment with notebooks without being misled.

7 DISCUSSION

With Imprimer we explore the opportunities and requirements of moving from a paradigm of *fabrication-as-executing-programs* to *fabrication-as-writing-programs* in fabrication systems research. As mentioned before, writing code to control CNC machines is a rich practice in professional machine shops and an improvised affair in exploratory fabrication but remains underexplored from an interaction perspective. Below we discuss how complexity manifested for users when they used Imprimer and how to support an engagement with this irreducible aspect of fabrication.

Deep engagement with digital fabrication require both programming and manufacturing knowledge. Fabrication systems in HCI research tend to lower the threshold to fabrication by abstracting away complexity. This approach invites more users in, which is extremely valuable to increase and diversify participation and normalize fabrication practices. Yet, rather than asking how we can facilitate the fabrication process, we were motivated to ask how we

can support the acquisition of programming and manufacturing knowledge necessary to transform makers from passive users to active developers of workflows.

Because the computational notebook paradigm offers the ability to weave rich documentation with machine code, it moves the programming practices of machine shops to a wider audience, and with better readability and portability than raw G-code. Instead of espousing a vision of fabrication as more streamlined, with the divide between bits and atoms rendered invisible by "seamless couplings" [25], we suggests a different type of interaction with CNC machines rooted in sustained engagement and an attitude of troubleshooting. We argue that a literate programming environment such as computational notebooks best supports this approach, which exposes all the "wiring" while offering scaffolding, support, and flexibility.

For example, P2 demonstrated that their fabrication process was a series negotiations between physical contingencies, the notebook's code, and its visual debugging tools. When they sought to make a profile cut and discovered that this function was not supported in the notebook, they went ahead and wrote their own code for a profile cut. They then adjusted the cut depth by first checking in the notebook's 3D view, then by measuring the stock with calipers. That P2 did so indicated not only their understanding of machine behavior (the CNC machine cutting deeper through the material with each pass, and the importance of defining each pass' depth) but also how their engagement was sustained by the back and forth between programming, digital, and physical verification. Rather than going through the process of "loading computer-aided design (CAD) files into a fabricator" [29], P2 could directly sketch their design, generate toolpaths, and make adjustments on the fly.

By including source code, notebooks can be easily customized to support various fabrication goals and expertise. We started this project wondering how to write and format the notebooks so that they were legible *and* provided optimal machine control, but this question became less important as more users used them. Each user came with a different background, programming sensibility and sense of possibilities of what they could make with a CNC mill. The principles of universal design [18] paled when faced with the diversity of fabrication experience and contexts, even with such a small sample. For instance, P3 and P4 both mentioned that the graphical elements of the notebooks did not add to the experience of programming the ShopBot. P5, on the other hand, saw great pedagogical value in the ability to constrain the notebooks to specific functionalities and value ranges. While P2 enjoyed the ability to sketch freehand geometry, P3 would have liked to connect the notebook to a drawing API for cutting accurate shapes. The level *and* the locus of complexity varied for each user according to their personal goals and concerns—for some it was at the design level, for others in the machining aspects, for others in the particulars of programming. Complexity was not a fixed variable that one system could address; it was a shifting tension that varied with each maker and with each session.

The entangled challenges and benefits of representing aspects of the machining process through literate programming raise the question of how to appropriately guide makers through a notebook. In particular, we observed a tension between the goals of *getting something done* versus *learning how something works*. Many existing systems will orient makers towards getting things done, hiding complexity and making design decisions that ensure manufacturability and ease of use. Others will offer many functionalities as well as in-app and external support to guide makers through the many features. With Imprimer, we contemplate a third alternative: to show the code underlying each functionality from the start so that makers can develop their own personal "grammar" of machine control.

8 CONCLUSION AND FUTURE WORK

Ultimately, Imprimer is only the beginning of an emerging paradigm: digital fabrication as writing literate programs. Leveraging its network of notebooks, a machine, and an augmented reality overlay, we showed how to extend computational notebooks beyond their envisioned use in data science into the physical world. Through demonstrations and a user study, we examined the effectiveness of starting with low-level machine control and abstracting up to more human-centered machine interactions. Future work will be needed to incorporate the vast breadth of existing CNC milling techniques into open source literate code while also adding entirely novel ones. Nevertheless, it is precisely through programming that we aim to anchor further thought and experimentation with CNC milling across physical and digital worlds.

ACKNOWLEDGEMENTS

We would like to thank the participants of our user study for leading us in new directions as to what could be possible with literate programming for digital fabrication. We are grateful for the help of Ted Hall and Jennifer Jacobs in engineering direct control for the Shopbot. We would also like to thank the anonymous reviewers for their insightful feedback. Finally, we thank the Alfred P. Sloan foundation for financial support in carrying out this work.

REFERENCES

[1] Lea Albaugh, Scott E. Hudson, Lining Yao, and Laura Devendorf. 2020. Investigating Underdetermination Through Interactive Computational Handweaving. In Proceedings of the 2020 ACM Designing Interactive Systems Conference (DIS '20). Association for Computing Machinery, New York, NY, USA, 1033–1046. https://doi.org/10.1145/3357236.3395538

[2] American Society of Mechanical Engineers. 2019. Dimensioning and tolerancing: engineering product definition and related documentation practices. American Society of Mechanical Engineers, New York. OCLC: 1091588533.

[3] Chris Anderson. 2012. Makers: The New Industrial Revolution. Random House.

[4] Autodesk. 2022. Fusion 360. https://www.autodesk.com/products/fusion-360/overview

[5] Rick Bitter, Taqi Mohiuddin, and Matt Nawrocki. 2006. LabVIEW: Advanced programming techniques. Crc Press.

[6] Susanne Bødker. 2006. When Second Wave HCI Meets Third Wave Challenges. In Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles (Oslo, Norway) (NordiCHI '06). Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/1182475.1182476

[7] Mike Bostock. 2022. Observable Inputs. https://github.com/observablehq/inputs original-date: 2021-01-25T16:50:46Z.

[8] Brendan Harmon. 2022. CNC Surface Milling. https://baharmon.github.io/cnc-surface-milling

[9] Souti Chattopadhyay, Ishita Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376729

[10] Grigoris Daskalogrigorakis, Salva Kirakosian, Angelos Marinakis, Vaggelis Nikolidakis, Ioanna Pateraki, Aristomenis Antoniadis, and Katerina Mania. 2021. G-Code Machina: A Serious Game for G-code and CNC Machine Operation Training. In 2021 IEEE Global Engineering Education Conference (EDUCON). 1434–1442. https://doi.org/10.1109/EDUCON46332.2021.9453982

[11] Dassault Systèmes. 2022. Solidworks. https://www.solidworks.com/home-page-2021

[12] Robert A DeLine. 2021. Glinda: Supporting Data Science with Live Programming, GUIs and a Domain-specific Language. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, 1–11. http://doi.org/10.1145/3411764.3445267

[13] Audrey Desjardins and Timea Tihanyi. 2019. ListeningCups: A Case of Data Tactility and Data Stories. In Proceedings of the 2019 on Designing Interactive Systems Conference (San Diego, CA, USA) (DIS '19). Association for Computing Machinery, New York, NY, USA, 147–160. https://doi.org/10.1145/3322276.3323694

[14] Ian Drosos, Titus Barik, Philip J. Guo, Robert DeLine, and Sumit Gulwani. 2020. Wrex: A Unified Programming-by-Example Interaction for Synthesizing Readable Code for Data Scientists. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (Honolulu, HI, USA) (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376442

[15] Emily Peyton. 2021. How to Communicate Between CNC Machinery and a Computer. https://www.itbriefcase.net/how-to-communicate-between-cnc-machinery-and-a-computer-through-usb-ports

[16] Sean Follmer, David Carr, Emily Lovell, and Hiroshi Ishii. 2010. CopyCAD: remixing physical objects with copy and paste from the real world. In Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology (UIST '10). Association for Computing Machinery, New York, New York, USA, 381–382. https://doi.org/10.1145/1866218.1866230

[17] Frikk Fossdal, Rogardt Heldal, and Nadya Peek. 2021. Interactive Digital Fabrication Machine Control Directly Within a CAD Environment. In Symposium on Computational Fabrication (SCF '21). Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3485114.3485120

[18] Christian Fuchs and Marianna Obrist. 2010. HCI and Society: Towards a Typology of Universal Design Principles. International Journal of Human-Computer Interaction 26, 6 (2010), 636–656. https://doi.org/10.1080/10447311003781334

[19] Neil Gershenfeld. 2007. Fab: The Coming Revolution on Your Desktop–from Personal Computers to Personal Fabrication. Basic Books, Inc., USA.

[20] Andrew Gleadall. 2021. FullControl GCode Designer: Open-source software for unconstrained design in additive manufacturing. Additive Manufacturing 46 (Oct. 2021), 102109. https://doi.org/10.1016/j.addma.2021.102109

[21] Bruna Goveia da Rocha, Johannes M. L. van der Kolk, and Kristina Andersen. 2021. Exquisite Fabrication: Exploring Turn-taking between Designers and Digital Fabrication Machines. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3411764.3445236

[22] Andrew Head, Fred Hohman, Titus Barik, Steven M. Drucker, and Robert DeLine. 2019. Managing Messes in Computational Notebooks. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). ACM, New York, NY, USA, 270:1–270:12. https://doi.org/10.1145/3290605.3300500 event-place: Glasgow, Scotland Uk.

[23] Andrew Head, Jason Jiang, James Smith, Marti A. Hearst, and Björn Hartmann. 2020. Composing Flexibly-Organized Step-by-Step Tutorials from Linked Source Code, Snippets, and Outputs. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376798

[24] Nathaniel Hudson, Celena Alcock, and Parmit K. Chilana. 2016. Understanding Newcomers to 3D Printing: Motivations, Workflows, and Barriers of Casual Makers. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 384–396. https://doi.org/10.1145/2858036.2858266

[25] Hiroshi Ishii and Brygg Ullmer. 1997. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (Atlanta, Georgia, USA) (CHI '97). Association for Computing Machinery, New York, NY, USA, 234–241. https://doi.org/10.1145/258549.258715

[26] Jennifer Jacobs and Amit Zoran. 2015. Hybrid Practice in the Kalahari: Design Collaboration through Digital Tools and Hunter-Gatherer Craft. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (Seoul, Republic of Korea) (CHI '15). Association for Computing Machinery, New York, NY, USA, 619–628. https://doi.org/10.1145/2702123.2702362

[27] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. 2020. mage: Fluid Moves Between Code and Graphical Work in Computational Notebooks. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20). Association for Computing Machinery, New York, NY, USA, 140–151. https://doi.org/10.1145/3379337.3415842

[28] Jeeeun Kim, Clement Zheng, Haruki Takahashi, Mark D Gross, Daniel Ashbrook, and Tom Yeh. 2018. Compositional 3D Printing: Expanding & Supporting Workflows Towards Continuous Fabrication. In Proceedings of the 2Nd ACM Symposium on Computational Fabrication (SCF '18). ACM, New York, NY, USA, 5:1–5:10. https://doi.org/10.1145/3213512.3213518

[29] Jeeeun Kim, Clement Zheng, Haruki Takahashi, Mark D Gross, Daniel Ashbrook, and Tom Yeh. 2018. Compositional 3D Printing: Expanding and Supporting Workflows towards Continuous Fabrication. In Proceedings of the 2nd ACM Symposium on Computational Fabrication (Cambridge, Massachusetts) (SCF '18). Association for Computing Machinery, New York, NY, USA, Article 5, 10 pages. https://doi.org/10.1145/3213512.3213518

[30] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. 2016. Jupyter Notebooks - a publishing format for reproducible computational workflows. In ELPUB.

[31] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows, Fernando Loizides and Birgit Scmidt (Eds.). IOS Press, 87–90. https://doi.org/10.3233/978-1-61499-649-1-87

[32] Donald E. Knuth. 1984. Literate Programming. Comput. J. 27, 2 (may 1984), 97–111. https://doi.org/10.1093/comjnl/27.2.97

[33] Oliver David Krieg. 2022. Aestus by odk.design. https://oliverdavidkrieg.com/

[34] Maria Larsson, Hironori Yoshida, Nobuyuki Umetani, and Takeo Igarashi. 2020. Tsugite: Interactive Design and Fabrication of Wood Joints. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (Virtual Event, USA) (UIST '20). Association for Computing Machinery, New York, NY, USA, 317–327. https://doi.org/10.1145/3379337.3415899

[35] Mackenzie Leake, Frances Lai, Tovi Grossman, Daniel Wigdor, and Ben Lafreniere. 2021. PatchProv: Supporting Improvisational Design Practices for Modern Quilting. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3411764.3445601

[36] 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 (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376765

[37] Jingyi Li, Jennifer Jacobs, Michelle Chang, and Björn Hartmann. 2017. Direct and Immediate Drawing with CNC Machines. In Proceedings of the 1st Annual ACM Symposium on Computational Fabrication (Cambridge, Massachusetts) (SCF '17). Association for Computing Machinery, New York, NY, USA, Article 11, 2 pages. https://doi.org/10.1145/3083157.3096344

[38] Jingyi Li, Jennifer Jacobs, Michelle Chang, and Björn Hartmann. 2017. Direct and Immediate Drawing with CNC Machines. In Proceedings of the 1st Annual ACM Symposium on Computational Fabrication (SCF '17). ACM, New York, NY, USA, 11:1–11:2. https://doi.org/10.1145/3083157.3096344

[39] Richard Lin, Rohit Ramesh, Connie Chi, Nikhil Jain, Ryan Nuqui, Prabal Dutta, and Björn Hartmann. 2020. Polymorphic Blocks: Unifying High-level Specification and Low-level Control for Circuit Board Design. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20). Association for Computing Machinery, New York, NY, USA, 529–540. https://doi.org/10.1145/3379337.3415860

[40] Richard Lin, Rohit Ramesh, Nikhil Jain, Josephine Koe, Ryan Nuqui, Prabal Dutta, and Bjoern Hartmann. 2021. Weaving Schematics and Code: Interactive Visual Editing for Hardware Description Languages. In The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21). Association for Computing Machinery, New York, NY, USA, 1039–1049. https://doi.org/10.1145/3472749.3474804

[41] Molga, M and Smutnicki, C. 2005. Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf

[42] Stefanie Mueller, Martin Fritzsche, Jan Kossmann, Maximilian Schneider, Jonathan Striebel, and Patrick Baudisch. 2015. Scotty: Relocating Physical Objects Across Distances Using Destructive Scanning, Encryption, and 3D Printing. In Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '15). ACM, New York, NY, USA, 233–240. https://doi.org/10.1145/2677199.2680547

[43] Stefanie Mueller, Anna Seufert, Huaishu Peng, Robert Kovacs, Kevin Reuss, François Guimbretière, and Patrick Baudisch. 2019. FormFab: Continuous Interactive Fabrication. In Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction (Tempe, Arizona, USA) (TEI '19). Association for Computing Machinery, New York, NY, USA, 315–323. https://doi.org/10.1145/3294109.3295620

[44] Florian Müller, Maximilian Barnikol, Markus Funk, Martin Schmitz, and Max Mühlhäuser. 2018. CaMea: Camera-Supported Workpiece Measurement for CNC Milling Machines. In Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference (Corfu, Greece) (PETRA '18). Association for Computing Machinery, New York, NY, USA, 345–350. https://doi.org/10.1145/3197768.3201569

[45] Anders Mørch. 1997. Customization, Integration, and Extension. Computers and design in context (1997), 51. Publisher: MIT press.

[46] Nicolas Padfield. 2017. More elegant CNC dogbones. http://fablab.ruc.dk/more-elegant-cnc-dogbones/

[47] Observable. 2022. Observable HQ. https://observablehq.com/

[48] Jeffrey M. Perkel. 2021. Reactive, reproducible, collaborative: computational notebooks evolve. Nature 593 (2021), 156–157. https://doi.org/10.1038/d41586-021-01174-w

[49] Miller Puckette. 2022. Pure Data. https://puredata.info/

[50] Robert McNeel & Associates. 2022. Rhinoceros 3D. https://www.rhino3d.com/

[51] Adam Rule, Aurélien Tabard, and James D. Hollan. 2018. Exploration and Explanation in Computational Notebooks. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (2018).

[52] Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. 2013. PacCAM: Material Capture and Interactive 2D Packing for Efficient Material Usage on CNC Cutting Machines. In Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (St. Andrews, Scotland, United Kingdom) (UIST '13). Association for Computing Machinery, New York, NY, USA, 441–446. https://doi.org/10.1145/2501988.2501990

[53] ShopBot. 2022. ShopBot Tools. https://www.shopbottools.com/

[54] P. Smid. 2008. CNC Programming Handbook (third ed.). Industrial Press, Incorporated.

[55] Blair Subbaraman and Nadya Peek. 2022. p5.fab: Direct Control of Digital Fabrication Machines from a Creative Coding Environment. arXiv:2205.00323 [cs] (April 2022). http://arxiv.org/abs/2205.00323 arXiv: 2205.00323.

[56] Steven L. Tanimoto. 2013. A perspective on the evolution of live programming. In 2013 1st International Workshop on Live Programming (LIVE). 31–34. https://doi.org/10.1109/LIVE.2013.6617346

[57] Ye Tao, Guanyun Wang, Caowei Zhang, Nannan Lu, Xiaolian Zhang, Cheng Yao, and Fangtian Ying. 2017. WeaveMesh: A Low-Fidelity and Low-Cost Prototyping Approach for 3D Models Created by Flexible Assembly. ACM, 509–518. https://doi.org/10.1145/3025453.3025699

[58] Alexander Teibrich, Stefanie Mueller, François Guimbretière, Robert Kovacs, Stefan Neubert, and Patrick Baudisch. 2015. Patching Physical Objects. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15). Association for Computing Machinery, New York, NY, USA, 83–91. https://doi.org/10.1145/2807442.2807467

[59] Rundong Tian and Eric Paulos. 2021. Adroid: Augmenting Hands-on Making with a Collaborative Robot. In The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21). Association for Computing Machinery, New York, NY, USA, 270–281. https://doi.org/10.1145/3472749.3474749

[60] Rundong Tian, Vedant Saran, Mareike Kritzler, Florian Michahelles, and Eric Paulos. 2019. Turn-by-Wire: Computationally Mediated Physical Fabrication. In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19). Association for Computing Machinery, New Orleans, LA, USA, 713–725. https://doi.org/10.1145/3332165.3347918

[61] Rundong Tian, Sarah Sterman, Ethan Chiou, Jeremy Warner, and Eric Paulos. 2018. MatchSticks: Woodworking through Improvisational Digital Fabrication. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, 1–12. http://doi.org/10.1145/3173574.3173723

[62] Jasper Tran O'Leary, Eunice Jun, and Nadya Peek. 2022. Improving Programming for Exploratory Digital Fabrication with Inline Machine Control and Styled Toolpath Visualizations. In Proceedings of the 7th Annual ACM Symposium on Computational Fabrication (SCF '22). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3559400.3561998

[63] Jasper Tran O'Leary, Chandrakana Nandi, Khang Lee, and Nadya Peek. 2021. Taxon: a Language for Formal Reasoning with Digital Fabrication Machines. In The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21). Association for Computing Machinery, New York, NY, USA, 691–709. https://doi.org/10.1145/3472749.3474779

[64] Hannah Twigg-Smith, Jasper Tran O'Leary, and Nadya Peek. 2021. Tools, Tricks, and Hacks: Exploring Novel Digital Fabrication Workflows on #PlotterTwitter. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 594, 15 pages. https://doi.org/10.1145/3411764.3445653

[65] Bret Victor. 2012. Learnable Programming. http://worrydream.com/#!/LearnableProgramming

[66] Guanyun Wang, Lining Yao, Wen Wang, Jifei Ou, Chin-Yi Cheng, and Hiroshi Ishii. 2015. xPrint: From Design to Fabrication for Shape Changing Interfaces by Printing Solution Materials. In SIGGRAPH Asia 2015 Posters (SA '15). ACM, New York, NY, USA, 7:1–7:1. https://doi.org/10.1145/2820926.2820944

[67] Christian Weichel, John Hardy, Jason Alexander, and Hans Gellersen. 2015. ReForm: Integrating Physical and Digital Design through Bidirectional Fabrication. In Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15). Association for Computing Machinery, Charlotte, NC, USA, 93–102. https://doi.org/10.1145/2807442.2807451

[68] 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 (Yokohama, Japan) (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 307, 12 pages. https://doi.org/10.1145/3411764.3445527

[69] Karl D.D. Willis, Cheng Xu, Kuan-Ju Wu, Golan Levin, and Mark D. Gross. 2010. Interactive fabrication: new interfaces for digital fabrication. In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (TEI '11). Association for Computing Machinery, New York, NY, USA, 69–72. https://doi.org/10.1145/1935701.1935716

[70] Yifan Wu, Joseph M. Hellerstein, and Arvind Satyanarayan. 2020. B2: Bridging Code and Interactive Visualization in Computational Notebooks. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20). Association for Computing Machinery, New York, NY, USA, 152–165. https://doi.org/10.1145/3379337.3415851

[71] Nur Yildirim, James McCann, and John Zimmerman. 2020. Digital Fabrication Tools at Work: Probing Professionals' Current Needs and Desired Futures. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20). Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376621

[72] Tianhong Catherine Yu and James McCann. 2020. Coupling Programs and Visualization for Machine Knitting. In Symposium on Computational Fabrication (SCF '20). Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3424630.3425410

[73] Amit Zoran. 2016. A Manifest for Digital Imperfection. XRDS 22, 3 (apr 2016), 22–27. https://doi.org/10.1145/2893491

[74] Amit Zoran and Joseph A. Paradiso. 2013. FreeD: A Freehand Digital Sculpting Tool. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). ACM, New York, NY, USA, 2613–2616. https://doi.org/10.1145/2470654.2481361

[75] Amit Zoran, Roy Shilkrot, Suranga Nanyakkara, and Joseph Paradiso. 2014. The Hybrid Artisans: A Case Study in Smart Tools. ACM Trans. Comput.-Hum. Interact. 21, 3 (June 2014), 15:1–15:29. https://doi.org/10.1145/2617570