A Framework-Informed Analysis of Accessibility Barriers in Desktop 3D Printing Software

Audrey Ballarin
Boston University
Boston, Massachusetts, USA

Abigale Stangl
Georgia Institute of Technology
Atlanta, Georgia, USA

Sushil Oswal
University of Washington
Seattle, Washington, USA

Emily Whiting
Boston University
Boston, Massachusetts, USA

Abstract

3D printing in principle enables Blind and Low-Vision users to create tactile reference materials and objects, but inaccessible software creates a major barrier for these users. Many graphical user interface (GUI) elements in desktop 3D printing software are undetectable by standard accessibility APIs, such as UI Automation, used by screen readers to expose interfaces. We compare the hierarchies of API-detected elements with those obtained from the software source code, emphasizing detectability as a key metric in desktop accessibility. We assess the screen reader-based operability of core tasks in the 3D printing workflow, such as model positioning and slicing, and identify interface design and framework implementation patterns linked to accessibility failures. This evaluation is conducted on three open-source 3D printing software — Ultimaker Cura, PrusaSlicer, and Bambu Studio — using the NVDA screen reader. These findings provide framework-specific insights to inform retrofit and redesign decisions, contributing to accessible interaction paradigms in fabrication.

CCS Concepts

• **Human-centered computing** → **Accessibility design and evaluation methods**; **Empirical studies in accessibility**.

Keywords

3D printing, screen readers, Blind and Low Vision

ACM Reference Format:
Audrey Ballarin, Abigale Stangl, Sushil Oswal, and Emily Whiting. 2025. A Framework-Informed Analysis of Accessibility Barriers in Desktop 3D Printing Software. In *Designing Interactive Systems Conference (DIS '25 Companion), July 5–9, 2025, Funchal, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3715668.3736342

1 Introduction

3D printing has emerged as a powerful resource for accessibility, enabling Blind and Low Vision (BLV) users to create personalized libraries of tactile materials for practical use [6, 14, 16, 17]. However, BLV users often encounter significant accessibility barriers when using the slicing software required for preparing 3D prints due to their lack of keyboard navigability and compatibility with screen readers. These barriers often stem from the custom graphical user interface (GUI) frameworks on which these software are built, which limit interoperability with standard accessibility APIs.

The *Guidelines for Producing Accessible 3D Prints* [13] published by the Round Table on Information Access for People with Print Disabilities and Monash University outline best practices for accessible 3D modeling and printing, including independent workflows for BLV users. These guidelines highlight the scarcity of accessible slicing software, recommending paid software Simplify3D as the most accessible option, followed by Slic3r and OctoPrint, which provide partial support. Many users rely on educational institutions and makerspaces with preset machines and software [2]. Additionally, some 3D printers require specific slicing software due to firmware compatibility constraints (e.g., Cura for Ultimaker printers), underscoring the need to improve accessibility across all mainstream applications.

A key aspect of this study is the role of accessibility APIs and their interaction with software GUI frameworks. Screen readers rely on APIs such as Microsoft Active Accessibility (MSAA) and UI Automation (UIA) to interpret and convey GUI elements to users. While web accessibility standards such as the Web Content Accessibility Guidelines (WCAG) [21] and Accessible Rich Internet Applications (ARIA) roles have led to significant advancements in web-based software, desktop applications lack equivalent industry-wide accessibility guidelines. Furthermore, automated accessibility evaluation tools often fail in desktop applications as they cannot reliably detect or interpret custom GUI frameworks, obscuring both the causes and extent of accessibility problems.

This study examines the accessibility of three widely used open-source 3D printing software — Ultimaker Cura [3], PrusaSlicer [18], and Bambu Studio [8]. Cura is built on Uranium, a custom Qt-based framework that primarily utilizes custom-built QML components over standard Qt elements with native accessibility roles. PrusaSlicer, built on the open-source Slic3r project, uses wxWidgets, a C++ framework that incorporates standard Win32 elements detected by MSAA and UIA but also allows extensive GUI customization, which can impact accessibility. Bambu Studio is a fork of PrusaSlicer with additional wxWidgets-based GUI modifications.

This study investigates how GUI framework implementations and broader design choices cause user-side accessibility barriers, with a focus on (1) the detectability of GUI elements by accessibility APIs, (2) the operability of key 3D printing workflow tasks using a screen reader, and (3) the potential for accessibility retrofits within

DIS '25 Companion, Funchal, Portugal

these frameworks. Our study contributes to the broader effort to improve both 3D printing software and desktop applications at large for BLV users.

2 Related Work

While much of previous work in 3D printing for BLV users focuses on the accessibility of fabricated outputs [4–6, 14, 17], less research has been done on making the fabrication workflows themselves independently accessible to BLV users. Studies have investigated alternative 3D modeling interfaces, such as shape-based modeling and tangible CAD tools [9, 19], and explored making 3D printers accessible using a camera-based prototype that reads firmware menus from the printers' otherwise inaccessible LCD screens [20]. However, the accessibility of the slicing software required for 3D printing has not yet been examined. Beyond fabrication, prior work in desktop software accessibility has assessed UI/UX prototyping tools [10] and programming environments [15], and developed probabilistic models for desktop software evaluation [7]. However, this work has not extensively analyzed how highly customized software frameworks, GUI elements, and interaction structures affect accessibility. This study addresses this gap with the broader goal of making complex desktop software workflows more accessible.

3 Methodology

This evaluation is conducted on three slicing software: Ultimaker Cura 5.6.0, PrusaSlicer 2.8.1, and Bambu Studio 1.10.1.50. Testing is performed on a Windows system using **NVDA** 4.4.2, a widely adopted open-source screen reader. The NVDA-detected GUI elements are cross-referenced with **Windows Inspect**, a UI inspection tool for analyzing element hierarchies and metadata detected by the UIAutomation and MSAA APIs.

3.1 Definitions of 3D Printing Workflow Tasks

We define **five core tasks** in 3D printing software workflows: 1) **File Import and Management**; 2) **Model Positioning and Transformation**; 3) **Printer, Material, and Extruder Selection**; 4) **Print Parameter Setting**; 5) **Slicing and Print Job Execution**. The Round Table guidelines [13] emphasize that 3D printing workflows for BLV users include steps for adjusting the camera view angle to take screenshots of the 3D model to obtain AI-generated feedback describing the images, remote printing to bypass inaccessible firmware in the 3D printer itself, and generating supports on all 3D prints in case of unknown overhangs; these are included as subtasks in the evaluation.

3.2 Navigation Methods: System Focus versus NVDA Object Navigation

We distinguish between two main methods of keyboard-based navigation. **System focus navigation** (also called tab-cycling, tab navigation, or tabbing) uses `Tab`, `Shift+Tab`, and `arrow` keys to move between interactive controls, such as buttons and edit fields, as defined by the software's tab order. This is the intended means of accessible keyboard navigation but is often very limited. **NVDA object navigation** refers to the use of the `NVDA+Shift+Arrow` keyboard commands implemented by NVDA to move through all detected GUI elements, including non-interactive elements such as text, in a hierarchical structure. Primarily designed for interface exploration, it is significantly slower than system focus and should not be the primary means for performing essential functions. However, it can become necessary when system focus navigation fails to reach key controls.

3.3 Testing Procedure

Our evaluation began with an initial exploration of the main interface of each slicing software using tab-cycling to determine the extent of the built-in keyboard accessibility for high-level elements. Given that tab-navigable elements were highly limited, we relied heavily on keyboard shortcuts for high-level functions and NVDA object navigation to access nested elements. For each 3D printing workflow task, we performed the following:

(1) **Tab Navigation:** We first attempted standard tab navigation to identify which interactive controls were reachable and tested whether the `Esc` key could close menus.
(2) **NVDA Navigation:** We then re-attempted the task using any applicable keyboard shortcuts and used NVDA object navigation for the remaining unreachable elements.
(3) **Manual GUI Analysis:** For elements that were undetected, behaved unexpectedly, or were otherwise inaccessible per WCAG 2.2 guidelines [21] we conducted the GUI analysis: We searched the UIA hierarchy through Inspect to determine whether and how elements were exposed to the API. We then inspected the software source code to analyze relevant classes, objects, and interactions modes, including custom GUI elements (e.g., `wxWindowNR` panes), missing labels or accessibility roles, parent-child relationships, and custom interaction modes based on mouse input or non-default key behaviors.

Our findings categorize GUI development patterns responsible for accessibility barriers across the evaluated software and highlight critical obstacles preventing completion of each 3D printing task. Since this evaluation was conducted by a sighted user, our findings focus on **basic operability** — whether a screen reader user can independently navigate to and complete these tasks — while recognizing that a more granular assessment of **efficiency and usability** will require further evaluation by BLV expert users.

4 Findings

4.1 3D Printing Task Evaluations

This subsection summarizes key accessibility findings across five core tasks in the 3D printing workflow. Bolded terms refer to specific GUI development patterns identified as causes of accessibility barriers throughout the software, as categorized in Table 2. The heatmap in Table 1 depicts the severity of accessibility barriers across the tasks and evaluated software. Model transformation and print parameter setting are the least accessible tasks overall due to their reliance on inoperable custom menus and mouse-based interactions on the build plate without accessible alternatives through standard controls. Cura is most affected by undetected custom menus, while PrusaSlicer and Bambu Studio are most affected by overridden navigation keys and undetected text labels on custom panels.

Table 1: Heatmap of Accessibility Barrier Severity Across 3D Printing Tasks and Software (1 = lower severity, 4 = higher severity)

Task	Cura	PrusaSlicer	Bambu
File Import & Management	1	1	2
↓			
Model Positioning & Transformation	4	3	4
↓			
Printer, Material, & Extruder Selection	2	3	3
↓			
Print Parameter Setting	4	3	3
↓			
Slicing & Print Job Execution	2	2	2

Legend of Severity Levels:

Level	Description
1	Low barrier; all functions are keyboard-operable without NVDA object navigation.
2	Critical functions are keyboard-operable without NVDA object navigation; some information or secondary functions are not accessible.
3	One or more critical functions inoperable; caused primarily by modifiable implementation choices (e.g., remapping a keyboard shortcut).
4	One or more critical functions inoperable; caused by framework incompatibility or low-level design decisions (requiring deeper restructuring).

File Import and Management: All three slicers support shortcuts for opening, importing, and saving files. Recent files are available from the *File* menu bar, which is accessible with standard keyboard shortcuts in Cura and PrusaSlicer, but not in Bambu.

Model Positioning and Transformation: Transform menus (e.g., Move, Rotate, Scale) are widely inoperable across all three applications. A major barrier is the **tab key override**, which blocks standard keyboard navigation: in Cura, focus becomes trapped inside input fields; in PrusaSlicer and Bambu, pressing `Tab` initiates slicing instead of moving the focus to menu controls.

Mouse-dependent operations are also common – Cura lacks input fields for rotation and mirror tools, and PrusaSlicer's *Place on Face* function, a semi-automated orientation tool, requires mouse input (Figure 3). While not yet standard across slicers, auto-orientation features could be particularly valuable for non-visual users if implemented accessibly; Bambu notably offers a keyboard-accessible auto-orientation shortcut. While PrusaSlicer and Bambu do provide input fields for rotation, these use relative rather than global values; the fields automatically reset to zero after each input, preventing screen reader users from verifying the model's actual orientation over time.

Finally, axis, unit, and function context is unannounced across all applications. This is particularly severe in Cura and Bambu, where this information is rendered using **undetectable** custom panel components. While PrusaSlicer exposes more detectable information, it remains unreachable due to navigation overrides. All applications also **lack labeling** metadata for the input controls themselves.

Printer, Material, and Extruder Selection: In Cura, the main printer and material selection menus are fully unreachable due to being **undetected** `Cura.ExpandableComponent` elements (Figure 1). However, the same functions are also made available through the *Settings* menu (Figure 2), which is reachable with a standard keyboard shortcut (`Alt+S`), making printer, material, and extruder assignment fully operable. One barrier is that adding a new printer to the list of presets opens an inaccessible **popup dialog**. Meanwhile, PrusaSlicer and Bambu Studio both rely largely on a custom `PresetComboBox` for printer and material selection, which is **undetected** and unreachable with NVDA navigation. This element technically supports tabbing, but the **tab override** causes the `Tab` key to trigger slicing, making it essentially inoperable. The only workaround during testing was to navigate to a detectable element using NVDA commands and then tab forward to the combo box – a process unlikely to be discovered non-visually as the combo box's presence is not detected or announced to begin with.

Print Parameter Setting: Cura's print settings menu is unreachable as its selector is an **undetectable** `Cura.ExpandableComponent` (Figure 1). In PrusaSlicer and Bambu, **tab override** prevents tabbing to the print settings, requiring slower NVDA navigation. The labels on the custom panels are **undetectable**, and the input fields are **missing labeling**. PrusaSlicer offers a search bar which could, in principle, facilitate non-visual access, but the text on the custom dropdown `wxDataViewItem` list is **undetectable**, read by the screen reader as "Row 1", "Row 2", etc.

Slicing and Print Job Execution: All slicers allow slicing and sending print jobs via shortcuts. However, print time and filament estimates are unannounced in Cura, and errors are unannounced across all three software, due to **undetectable** custom panels.

4.2 Overview of Problematic GUI Development Patterns Identified In Source Code

Table 2 outlines specific GUI development patterns identified throughout the source code that are linked to accessibility failures in the 3D printing workflow (4.1). These include undetectable and unlabeled GUI elements, custom navigation modes that interfere with standard tab navigation, exclusively mouse-based functions, and mishandled popup dialogs.

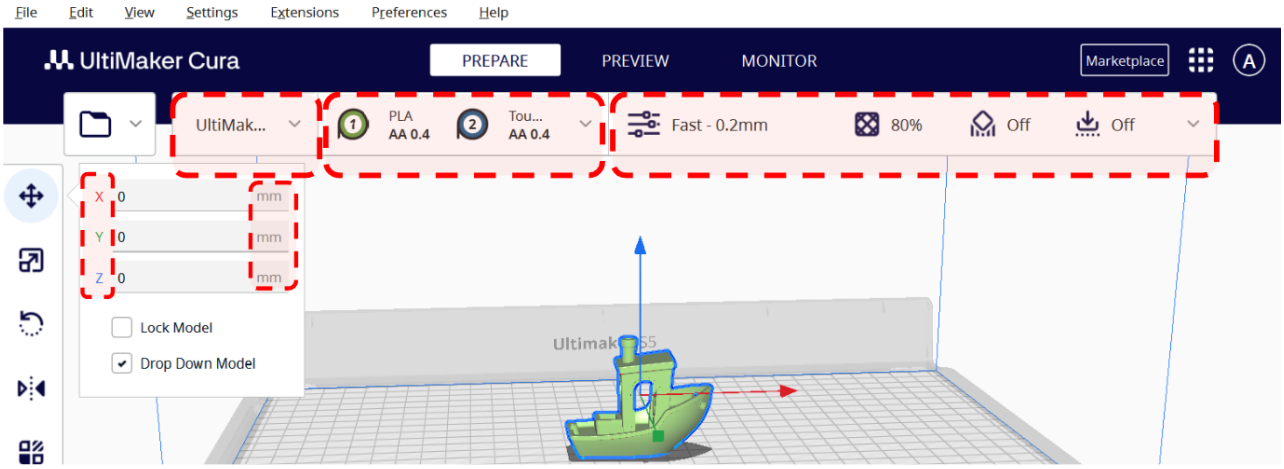

Figure 1: Screenshot of the Cura build plate interface highlighting GUI elements undetectable by accessibility APIs (red dashed boxes). These include the selectors for the printer, material, and print parameter menus, and axis and unit labels on the *Move* menu.

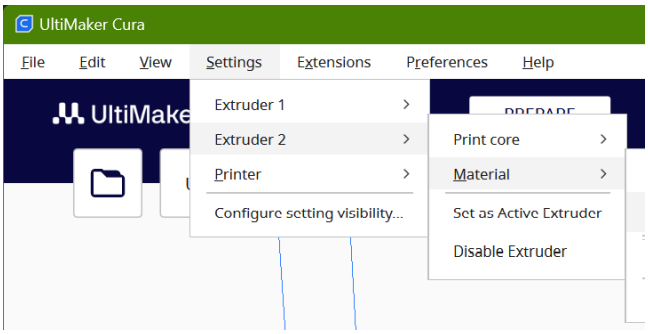

Figure 2: Accessible printer, extruder, and material selection in Cura via the standard Settings menu. Unlike the custom GUI elements shown in Figure 1, this top menu bar implementation is keyboard-navigable and screen reader compatible.

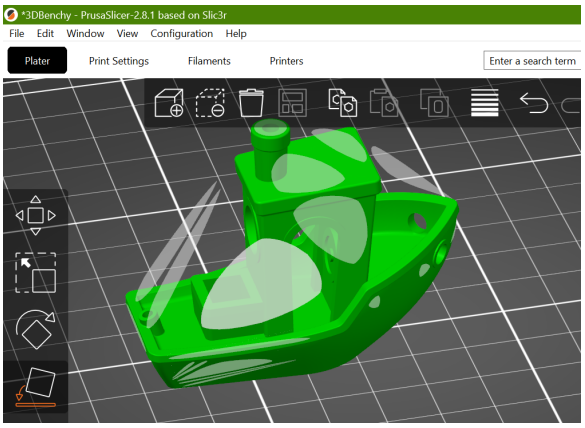

Figure 3: Screenshot from PrusaSlicer of a 3DBenchy object on the build plate after the *Place on Face* function was selected using the `F` keyboard shortcut. A user must mouse-click one of the translucent regions to re-orient the object on the corresponding face.

5 Discussion

GUI Recommendations for Designing and Engineering 3D Printing Software. Retrofitting accessibility into 3D printing software poses significant challenges due to the widespread use of custom elements and overlapping barriers. Qt developers might improve accessibility by experimenting with Qt Quick *Accessible* roles, while wxWidgets developers would need to derive new versions of custom elements from the wxAccessible class — both of which are significant undertakings. However, while some improvements require deep restructuring, others can be achieved through simpler retrofits. We recommend the following practices:

- Do not override system focus keyboard commands (i.e., `Tab`, `Shift+Tab`, `Space`).
- Add sounds for key actions such as selecting objects, opening menus, and applying transformations.
- Leverage built-in keyboard support by making simpler configuration tasks, such as selecting a material or printer, available through the application menu bar.
- Keyboard shortcuts should follow predictable one-to-one mappings rather than state-dependent behaviors.
- Enable keyboard inputs for all transformation operations, with axis and unit information incorporated in edit field name and `Esc` key enabled for closing menus. Add a global rotation menu where the displayed input value represents the total rotation angle relative to the original orientation, so that users can reliably verify the current state without needing to track incremental changes.

Table 2: Inaccessible GUI Development Patterns Identified in Software

GUI Development Pattern	Description	Examples
Undetectable Custom Classes	Custom GUI elements are not exposed to the API, and therefore not detected by the screen reader	**Cura**: `Cura.Menu` elements. **Bambu**: Wx-based `ScalableBitmap` used for interactive panels. **PrusaSlicer**: `DynamicConfig` tooltip used for slicing error notifications
Tab Key Override	Software assigns custom function to Tab key, disabling tab-navigation	**Bambu & PrusaSlicer**: `Tab` switches between 3D View and Preview modes, fully overriding standard tab navigation; Preview mode also automatically triggers slicing, changing the interface functionality entirely without any non-visual alert **Cura**: In transform menus, `Tab` presses are hard coded to specific edit fields rather than to next interactive control, trapping focus within menu
Desynchronized Tab Support	Activating an element using system focus updates the current element incorrectly or incompletely	**Cura**: When transform menus (e.g., Rotate) are selected using system focus (Tab, Space) instead of Cura shortcuts, active tool fails to update correctly and previous menu re-opens
One-to-Many Custom Navigation Shortcuts	Non-standard one-to-many navigation mechanisms have unpredictable behavior and lack feedback for screen reader users	**Bambu**: Custom `Ctrl+Tab` shortcuts cycles between interface panels, but no announcement of active tab means current state is unknown and unverifiable to user. **PrusaSlicer**: `Ctrl+1/2/3/4` shortcuts for each panel are also unannounced, but one-to-one mapping makes navigation more predictable
Incomplete Labeling	Interactive controls (e.g., buttons, edit fields) have missing or insufficient labels	**Cura**: Edit fields announced as "Edit value 0.2" without function, axis, or unit context such as "scale", "X", "mm"
Mouse-Dependent Interaction Modes	Some functions require precise mouse interaction, with no alternative keyboard-based input	**Cura**: Rotate and Mirror functions rely exclusively on clicking and dragging axis handles, with no value input option **Bambu**: Custom expandable panels for adding printers defined only with `wxMouseEvents`
Mishandled Popup Dialogs	Popups lack proper control type announcement, focus handling, or keyboard escapability	**PrusaSlicer**: Configuration Update popup does not auto-focus; it is unreachable to screen readers while also disabling main window access

- Consider implementing accessible (automatically focused, announced, and escapable) popup dialogs as an alternative to undetectable custom tooltips for reporting build plate and slicing errors.

Future Work. Future research will pursue a deeper analysis of accessibility metrics such as complexity and navigation efficiency through user testing with BLV users. We will also explore practical and proof-of-concept retrofits, as well as an expansion of this evaluation to a Mac operating system setting with the VoiceOver screen reader. By combining an interaction study with extensive manual source code inspection, this work offers deeper insight into specific interaction functions and implementation details than automation alone would reveal. An alternative approach initially explored was a programmatic UI hierarchy comparison, extracting and comparing UI element trees from the NVDA console and the slicer source code build to quantify the proportion of detectable elements as a novel accessibility metric for framework-based desktop software. While full execution is ongoing, this methodology remains a promising diagnostic tool for developers to improve accessibility in their software frameworks. In contrast, the methodology implemented in this work serves to derive broader design insights specific to 3D printing software.

Broader Implications for BLV User Agency and Maker Participation. As 3D printing continues to expand in accessibility, education, and making, addressing barriers in software interaction paradigms is critical to support both skills development and maker community participation for BLV users. In an educational context, when access to 3D printing workflows is minimally restricted, makers can iterate over prints more extensively and, as a result, have more opportunities to develop effective technical strategies [1]. BLV users often choose software based not only on accessibility, but also on the availability of local support from peers with expertise in a specific program [12]. As a result, enforcing operability throughout the mainstream 3D printing software ecosystem – not just in one application – is essential for BLV users to be able to fully participate in shared knowledge networks. More broadly, maker environments are valued for providing greater user autonomy – especially compared to formal learning environments – which supports motivation, persistence, and identity formation [11]. Improving accessibility in 3D printing software is not only a

matter of usability – it is key to ensuring BLV users have access to the technical, social, and personal dimensions of fabrication.

Acknowledgments

We thank Chancey Fleet and Ka Yat Li for contributing valuable insights on non-visual fabrication. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 2234657. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Gabrielle Benabdallah, Sam Bourgault, Nadya Peek, and Jennifer Jacobs. 2021. Remote Learners, Home Makers: How Digital Fabrication Was Taught Online During a Pandemic. 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 350, 14 pages. doi:10.1145/3411764.3445450

[2] Emeline Brulé and Gilles Bailly. 2021. " Beyond 3D printers": Understanding Long-Term Digital Fabrication Practices for the Education of Visually Impaired or Blind Youth. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[3] Ultimaker B.V. [n. d.]. *Ultimaker Cura.* https://ultimaker.com/software/ultimaker-cura

[4] Kirk Andrew Crawford, Jennifer Posada, Yetunde Esther Okueso, Erin Higgins, Laura Lachin, and Foad Hamidi. 2024. Co-designing a 3D-Printed Tactile Campus Map With Blind and Low-Vision University Students. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility* (St. John's, NL, Canada) *(ASSETS '24)*. Association for Computing Machinery, New York, NY, USA, Article 77, 6 pages. doi:10.1145/3663548.3688537

[5] Leona Holloway, Kim Marriott, and Matthew Butler. 2018. Accessible Maps for the Blind: Comparing 3D Printed Models with Tactile Graphics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3173574.3173772

[6] Leona Holloway, Kim Marriott, Matthew Butler, and Samuel Reinders. 2019. 3D Printed Maps and Icons for Inclusion: Testing in the Wild by People who are Blind or have Low Vision. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility* (Pittsburgh, PA, USA) *(ASSETS '19)*. Association for Computing Machinery, New York, NY, USA, 183–195. doi:10.1145/3308561.3353790

[7] Md Touhidul Islam, Donald E Porter, and Syed Masum Billah. 2023. A Probabilistic Model and Metrics for Estimating Perceived Accessibility of Desktop Applications in Keystroke-Based Non-Visual Interactions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 43, 20 pages. doi:10.1145/3544548.3581400

[8] Bambu Lab. [n. d.]. *Bambu Studio.* https://bambulab.com/en-us/download/studio

[9] Myungjoong Lee, Seung-Won Kim, and Hyunki In. 2024. Archi-TangiBlock: A Modular Block Based Tangible Media as a CAD Tool for Visually Impaired People. *IEEE Access* 12 (2024), 46582–46595. doi:10.1109/ACCESS.2024.3382311

[10] Junchen Li, Garreth W. Tigwell, and Kristen Shinohara. 2021. Accessibility of High-Fidelity Prototyping Tools. 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 493, 17 pages. doi:10.1145/3411764.3445520

[11] Lee Martin. 2015. The promise of the maker movement for education. *Journal of Pre-College Engineering Education Research (J-PEER)* 5, 1 (2015), 4.

[12] Neal McKenzie. 2022. *Getting Started with 3D Printing (A New Hope): Part 1.* Retrieved May 13, 2025 from https://www.perkins.org/resource/getting-started-3d-printing-new-hope-part-1/

[13] Round Table on Information Access for People with Print Disabilities Inc. 2024. *Pyrmont, NSW 2009 Australia.* Round Table on Information Access for People with Print Disabilities Inc., Pyrmont, NSW 2009, Australia.

[14] Mahika Phutane, Julie Wright, Brenda Veronica Castro, Lei Shi, Simone R. Stern, Holly M. Lawson, and Shiri Azenkot. 2022. Tactile Materials in Practice: Understanding the Experiences of Teachers of the Visually Impaired. *ACM Trans. Access. Comput.* 15, 3, Article 17 (July 2022), 34 pages. doi:10.1145/3508364

[15] Venkatesh Potluri, Priyan Vaithilingam, Suresh Iyengar, Y. Vidya, Manohar Swaminathan, and Gopal Srinivasa. 2018. CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3173574.3174192

[16] Samuel Reinders. 2021. Accessible interactive 3D models for blind and low-vision people. *SIGACCESS Access. Comput.* 129, Article 6 (March 2021), 7 pages. doi:10.1145/3458055.3458061

[17] Samuel Reinders, Swamy Ananthanarayan, Matthew Butler, and Kim Marriott. 2023. Designing Conversational Multimodal 3D Printed Models with People who are Blind. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 2172–2188. doi:10.1145/3563657.3595989

[18] Prusa Research. [n. d.]. *PrusaSlicer.* https://help.prusa3d.com/downloads

[19] Alexa F Siu, Son Kim, Joshua A Miele, and Sean Follmer. 2019. shapeCAD: An accessible 3D modelling workflow for the blind and visually-impaired via 2.5 D shape displays. In *Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility*. 342–354.

[20] Naoya Tagawa, Masakazu Iwamura, Kazunori Minatani, and Koichi Kise. 2024. Making 3D Printer Accessible for People with Visual Impairments by Reading Scrolling Text and Menus. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility* (St. John's, NL, Canada) *(ASSETS '24)*. Association for Computing Machinery, New York, NY, USA, Article 131, 4 pages. doi:10.1145/3663548.3688517

[21] World Wide Web Consortium (W3C). 2024. Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/