A Multi-Modal Approach for Blind and Visually Impaired Developers to Edit Webpage Designs

Venkatesh Potluri, Liang He, Christine Chen, Jon E. Froehlich, Jennifer Mankoff
Paul G. Allen School of Computer Science & Engineering, University of Washington
{vpotluri, lianghe, chenc55, jonf, jmankoff}@cs.washington.edu

ABSTRACT
Blind and visually impaired (BVI) individuals are increasingly creating visual content online; however, there is a lack of tools that allow these individuals to modify the visual attributes of the content and verify the validity of those modifications. In this poster paper, we discuss the design and preliminary exploration of a multi-modal and accessible approach for BVI developers to edit visual layouts of webpages while maintaining visual aesthetics.

Author Keywords
Accessible programming; visual design; web; blind and visually impaired.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces

INTRODUCTION
Content creators who are blind and visually impaired (BVI) are actively building interfaces that contain visual elements, such as blogs and personal websites [6]. While these interfaces are expected to have good visual design, the necessary tools and information to build visually pleasing [3,12] layouts are inaccessible to BVI developers [11]. To address this problem, we propose a multi-modal approach that allows BVI developers to edit webpages without breaking visual aesthetics.

Prior research in accessible programming tools for BVI developers has addressed challenges such as effective code navigation for BVI developers [2] and proposed new integrated development environment (IDE) techniques to make programming more accessible [4,16,18]. These approaches, however, do not address the accessibility barriers specific to editing the visual attributes of user interfaces (UIs). To bridge the gap, *Borka* [5], an addon for the NVDA screen reader [14], informs the developer of the location, height, and width of web UI elements in pixels. Li *et al.* [11] built a self-voicing application on a tablet and tactile graphics to support visual layout editing. While related, our approach is unique in that it allows for web design edits both through an IDE as well as directly on the rendered output of a touchscreen—edits made via either mode are automatically assessed via a design validator.

Specifically, we introduce a multi-modal UI modification approach that combines an *accessible code editor* along with a *touchscreen* and *controller* to directly access, edit, and validate UI designs (Figure 1.). With our system, BVI developers edit CSS code from an existing template either using an accessible IDE or through a touchscreen. On the touchscreen, the BVI developer can *directly explore* the design via touch and use touch gestures to make *direct edits.* When an edit is made, it is assessed by the validation engine for visual design adherence. The screen reader that runs alongside the IDE announces the outcome of the edit. For example, if visual design guidelines are violated, the changes are reverted on both the IDE and the touchscreen, and the screen reader announces the design guideline violations.

In a preliminary pilot study, we found that, with minimal training, a BVI developer could effectively make meaningful visual edits with real-time validation using our system.

MULTI-MODAL SYSTEM
To explore our multi-modal approach, we created a three-part system (Fig. 1): an *accessible canvas* that allows BVI developers to use touch and gestures to modify the visual properties of web UI elements, a *code editor* that supports direct code modifications, and a *controller* that processes proposed updates from either the canvas or code editor and then checks if the updates violate design guidelines. Please refer to our supplementary video for a demonstration.

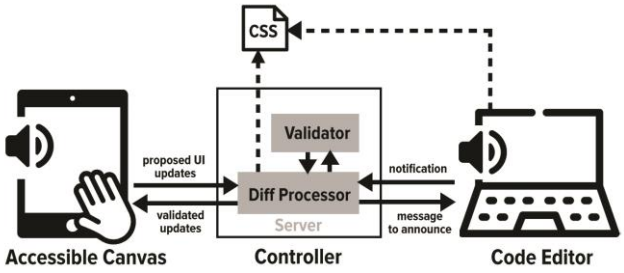

Figure 1. Multi-modal system overview. See video demo.

Design Guidelines
While visual design is subjective and hard to quantify [3], we ensure that the edits BVI developers make with our system adhere to an initial set of design guidelines extending from web design standards [1,12]:

ASSETS '19, October 28–30, 2019, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-6676-2/19/10.
https://doi.org/10.1145/3308561.3354626

- *Spacing consistency*. The horizontal and vertical spacing between elements at the same level should be consistent.
- *Typeface consistency*. Significant variation in fonts can be distracting and confusing. A maximum of three different font types on a page is enforced.
- *Color consistency*. UI elements of the same type should share the same color.

Accessible Canvas

To support exploration and manipulation of the web UI elements, our system displays a representation of the webpage (HTML and CSS) on a touchscreen (an iPad Air running iOS 12.2). The user can navigate and edit their design directly using gestures (Table 1), which are implemented via Apple UIkit Gestures. Since performing custom gestures in iOS conflicts with built-in VoiceOver, we built the accessible canvas to be self-voicing. The canvas announces the element the user touches and provides audio feedback on the user edits, which are performed by gestures (Table 1). Our gestures are based on prior work, such as risk-free exploration suggested by Kane *et al.* [7] and leveraging the familiarity of edge elements [8].

Operation	Touch and Gestures
Navigating	Touch and move
Aligning to one side	One-finger double taps and two-finger swipe
Style change mode	**Two-finger triple taps**
Changing width	Horizontal edge tap and one-finger moving horizontally
Changing height	Vertical edge tap and one-finger moving vertically
Margin/padding	**Four-finger long press**
Changing margin/padding	One-finger swipe to one side and one-finger moving vertically or horizontally
Background	**Three-finger long press**
Changing color	One-finger swipe left or right
Foreground	**Two-finger long press**
Changing font color	One-finger swipe left or right
Changing font size	One-finger swipe up and down
Changing font type	Two-finger swipe left or right
Submitting changes	**Three-finger triple taps**

Table 1. The full list of gestures used for navigation, alignment, and adjustment of element style.

Code Editor

In addition to making touchscreen edits, the developer can also directly edit the webpage CSS in Microsoft's Visual Studio Code (VS Code) using a keyboard and a screen reader. When the developer saves changes to the CSS file in the editor, a custom VS Code extension notifies the controller about the new code edits. Additionally, the extension receives messages from the controller (on the validity of the edits made in the editor or the accessible canvas). These messages are announced by the screen reader running alongside the editor.

Controller

The controller has two parts: a *diff processor* and *validator*. The *diff processor* receives proposed changes from the accessible canvas or a notification from the code editor, updates a working copy of the original CSS file on the controller, and then evaluates the updates with the *validator*. If the proposed change passes validation, the working copy is checkpointed, the accessible canvas is updated, and the code editor receives an accept message (announced by the screen reader). If the change violates a design guideline, the processor reverts the CSS to the last checkpoint, leaving the accessible canvas unchanged. A rejection message, including the violated design guideline, is then reported in the code editor (see the video).

To check for adherence to design guidelines, the validator maintains an internal representation of the visual attributes of the webpage elements (from the CSS file) and the hierarchy of elements (from the HTML). For spacing consistency, we used the margin attributes of adjacent elements for validation. For example, the margins of horizontally distributed elements should be the same. We examined the font and color attributes for typeface consistency and color consistency, respectively.

PRELIMINARY USER STUDY

We conducted a preliminary user study with a congenitally blind professional software engineer (age 24; male). The study had three parts: a semi-structured interview on current webpage development strategies, an evaluation of our system through three tasks (*i.e.*, changing the typeface of a paragraph, increasing the bottom margin for a list of hyperlinks, and aligning an image and a paragraph), and a post-study interview regarding usability. The participant completed all three tasks successfully and agreed that the gestures are usable, although he had trouble memorizing gestures for operations. He could understand the outcomes of the tasks—edits either passed or failed design guideline violation check. The participant expressed the need to have more control, for example, he wanted to have access to the history of changes that lead to a rejection.

CONCLUSION AND DISCUSSION

We presented an initial exploration of enabling BVI developers to edit webpage design using a multi-modal approach. Our system allows BVI developers to change visual aspects of web UI elements either by performing gestures on a touchscreen or by directly modifying code in an editor. We plan to enhance our validator by integrating the design guidelines with VizAssert [15], a framework that verifies accessibility of webpages. We are also interested in exploring machine learning techniques both for validation as well as to suggest alternative designs (*e.g.*, based on design trends suggested in [10]) [17]. Finally, we will investigate the discoverability and learnability of our current gesture set through a formal study. Besides gestures, we will assess viability of accessible interaction techniques in [9] and alternative input devices (*e.g.*, touchpads) for modifying visual attributes of interfaces.

ACKNOWLEDGEMENTS

This work is partly supported by NSF Award IIS-1836813. We thank Michael Ernst and Pavel Panchekha for their feedback on the development of our system.

REFERENCES

1. Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. 2011. Design principles for visual communication. *Commun. ACM* 54, 4 (April 2011), 60-69. DOI: https://doi.org/10.1145/1924421.1924439
2. Khaled Albusays, Stephanie Ludi, and Matt Huenerfauth. 2017. Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility* (ASSETS '17). ACM, New York, NY, USA, 91-100. DOI: https://doi.org/10.1145/3132525.3132550
3. Ahamed Altaboli and Yingzi Lin. 2011. Objective and subjective measures of visual aesthetics of website interface design: the two sides of the coin. In *International Conference on Human-Computer Interaction*, pp. 35-44. Springer, Berlin, Heidelberg
4. Catherine M. Baker, Lauren R. Milne, and Richard E. Ladner. 2015. StructJumper: A Tool to Help Blind Programmers Navigate and Understand the Structure of Code. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 3043-3052. DOI: https://doi.org/10.1145/2702123.2702589
5. Andy Borka. 2019. NVDA Community Addons / Addons / Developer Toolkit. Retrieved July 8, 2019 from https://addons.nvda-project.org/addons/developerToolkit.uk.html
6. Dolphin. 2016. Blogging when you're blind or visually impaired. Retrieved July 9, 2019 from https://yourdolphin.com/news?id=223
7. Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. 2008. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility* (Assets '08). ACM, New York, NY, USA, 73-80. DOI: https://doi.org/10.1145/1414471.1414487
8. Shaun K. Kane, Jacob O. Wobbrock, and Richard E. Ladner. 2011. Usable gestures for blind people: understanding preference and performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 413-422. DOI: https://doi.org/10.1145/1978942.1979001
9. Rushil Khurana, Duncan McIsaac, Elliot Lockerman, and Jennifer Mankoff. 2018. Nonvisual Interaction Techniques at the Keyboard Surface. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (CHI '18). ACM, New York, NY, USA, Paper 11, 12 pages. DOI: https://doi.org/10.1145/3173574.3173585
10. Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: design mining the web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 3083-3092. DOI: https://doi.org/10.1145/2470654.2466420
11. Jingyi Li, Son Kim, Joshua A. Miele, Maneesh Agrawala, and Sean Follmer. 2019. Editing Spatial Layouts through Tactile Templates for People with Visual Impairments. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (CHI '19). ACM, New York, NY, USA, Paper 206, 11 pages. DOI: https://doi.org/10.1145/3290605.3300436
12. Laura Martin. 2017. Typography Elements Everyone Needs to Understand. Retrieved July 8, 2019 from https://medium.com/gravitdesigner/typography-elements-everyone-needs-to-understand-5fdea82f470d
13. Microsoft. Visual Studio Code. Retrieved July 9, 2019 from https://code.visualstudio.com/
14. NV Access. Non Visual Desktop Access. Retrieved July 9, 2019 from https://www.nvaccess.org/about-nv-access/
15. Pavel Panchekha, Adam T. Geller, Michael D. Ernst, Zachary Tatlock, and Shoaib Kamil. 2018. Verifying that web pages have accessible layout. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (PLDI 2018). ACM, New York, NY, USA, 1-14. DOI: https://doi.org/10.1145/3192366.3192407
16. 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* (CHI '18). ACM, New York, NY, USA, Paper 618, 11 pages. DOI: https://doi.org/10.1145/3173574.3174192
17. Venkatesh Potluri, Tadashi Grindeland, Jon E. Froehlich, Jennifer Mankoff (2019). AI-Assisted UI Design for Blind and Low-Vision Creators. In the *ASSETS'19 Workshop: AI Fairness for People with Disabilities.*
18. Emmanuel Schanzer, Sina Bahram, and Shriram Krishnamurthi. 2019. Accessible AST-Based Programming for Visually-Impaired Programmers. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (SIGCSE '19). ACM, New York, NY, USA, 773-779. DOI: https://doi.org/10.1145/3287324.3287499