# Increasing the Accessibility of Programming to People who are Blind

**Amelia Wang**
University of Washington
Human Centered Design and Engineering
Seattle, Washington 98105
aw1998@uw.edu

## INTRODUCTION

In modern software development, IDEs (Integrated Development Environments) play a critical role in efficient workflow but rely heavily on visual cues. Programming, with the correct logic and support should be an accessible skill, but even accessible environments within IDEs fail to give visually impaired users an understanding of their code to the same level as their sighted counterparts. Workarounds and compatibility issues with their screen readers are often inconvenient and overwhelming, forcing many to default to simpler editors.

NVDA (Non-Visual Desktop Access) is an open-source screen reader offered for free, making it a popular choice for not only programming, but everyday use.

Blind programmers are at a crossroads in which the IDEs being used by their colleagues are not well accessible and their discomfort with bringing up their disability status. Many have taken these issues into their own hands and write custom scripts as a workaround to their barriers, but those who have succeeded only represent a small and experienced number of programmers. Barriers still exist for the majority of blind programmers and there is still a need for technology that supports them.

## APPROACH

My approach is to review literature that surrounds the issue of accessibility in programming and work on a plugin that works with NVDA as an add-on rather than a separate tool for optimal accessibility and use. Through this add-on, I hope to provide a more streamlined experience and better access to information like that IDEs should typically communicate, including:

- **Debugging**

It is important to locate bugs, understand their place order to fix errors in software. Due to the visual emphasis IDEs place on this typical process, a system will be developed to express wrong outputs, syntax errors, and runtime errors in real time.

- **Line and Column Numbers**

Understanding not only the location of lines of code, but in the context of the rest of the structure is important for development, especially for higher level projects and work.

- **Style**

Code style is very much a visually focused detail important in keeping things clean and understandable by users.

### Literature Review

Attempts to understand and resolve issues for blind programmers has increased over the years. These include design solutions like developing different code structures [2], using tactile aid [3], as well as conducting interviews of blind users already in the field. [1]. Together, these identify accessibility issues in programming and provide insight to integrating solutions.

1. **Design: StructJumper**

Baker et al. [2] developed a tool which allows screen reader users to move through their code quickly, communicating hierarchical information through sound. This eclipse add-on helps present tree-based information, which is important to understand context and structure of their code, without losing their place in the editor.

2. **Design: Blocks4All**

Milne [3] created a prototype to teach programming concepts to blind students using touchscreen laptops to test in schools. Their evaluation of block-based environments gave context to current issues in design

within the rest of the code, and edit existing work in and revealed fived accessibility barriers in programming.

### 3. SODBeans

SODBeans is a design intervention by Stefik et al. [4] that identifies the difficulty of debugging and the visual emphasis it has. They approach this by creating their own auditory based programming environment with a code compiler that sonifies programming errors.

### 4. Interviews

Albusays et al. [1] surveys blind developers who use and work around IDEs despite their complexity and inaccessibility. This exploratory study provided insight into the challenges and workarounds in the industry, as well as user preferences and pain points to improve on for future work.

Studies and tools around are becoming increasingly innovative, and they reveal the need for more solutions that can be worked into the industry.

## METHOD

With under 10 weeks to complete a project without prior experience in accessibility, my timeline will include learning material as well as building the add-on.

1. **Get to know NVDA + IDEs to decide what to work with - 7/20**
2. **Start Development- 7/27**
3. **Presentable Draft - 8/27**
4. **Expert Evaluation - 8/31**
5. **Final write up/documentation - 9/7**

## EVALUATION

This plugin will be evaluated by an expert in the field, Catie Baker, to provide high level feedback and will be made available to the public through Github at https://github.com/wangamelia/nvdaproj

## REFERENCES


1. K. Albusays, S. Ludi, M. Huenerfauth, "Interviews and Observation of Blind Software Developers at Work to Understand Code Navigation Challenges," in *Proc. of the 19th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2017, October 20 - November 1 2017, Baltimore, Maryland, USA*

2. 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

3. Lauren R. Milne. 2017. Blocks4All: making block programming languages accessible for blind children. SIGACCESS Access. Comput. 117 (February 2017), 26-29. DOI: https://doi.org/10.1145/3051519.3051525

4. A. Stefik, A. Haywood, S. Mansoor, B. Dunda and D. Garcia, "SODBeans," *2009 IEEE 17th International Conference on Program Comprehension*, Vancouver, BC, 2009, pp. 293-294. doi: 10.1109/ICPC.2009.5090064