Challenges when using accessibility tools

This section will discuss the challenges faced by visually impaired programmers. In recent years, there has been growing attention to the challenges faced by visually impaired programmers, and significant progress has been made in research to understand and address these challenges. We will focus on the challenges faced by porgrammers with severe levels of visual impairment, who use screen-readers and refreshable braille displays.

Challenges associated with using screen readers and braille displays:

\begin{itemize}

\item Limited to reading code sequentially (line by line): (Javaspeak) \cite{code\_mirror\_block\_2019} A consequence of this is lower levels of code comprehension \cite{ Mountapmbeme\_Okafor\_Ludi\_2022}. The limitation of reading code sequentially, creates further challenges for visually impaired programmers.

\begin{itemize}

\item Harder to skim code: Being constrained to reading code line-by-line makes blind programmers unable to “skim” code in the same way that sighted programmers can. Code skimming is a useful way for users to get a quick overview of the main functionality of code without getting bogged down by the details. Screen-readers and braille displays do not offer any method for visually impaired programmers to effectively skim code. The consequences of this are increased time required to read over code, and lower levels of code comprehension. \cite{structjumper}

\item Harder to back-track: When using screen readers and braille displays, it can be challenging for programmers to navigate back to the part of code they were previously focused on. Sight allows programmers to easily identify and correct errors, but for visually impaired programmers using accessibility tools, this process can be much more difficult \cite{Javaspeak}.

\item Harder to understand structure of code represented through visual cues: Many programming languages use visual cues such as indentation and whitespace to indicate the structure of code. Visually impaired programmers who use screen readers or braille displays may struggle to perceive these visual cues, making it more difficult to understand the structure of code. \cite{code\_mirror\_block\_2019}

\item Harder to find information within a code base without losing the position of focus: Visually impaired programmers have reported challenges related to moving around different parts of code without losing previous focus \cite{ Albusays\_Ludi\_2016}.

\end{itemize}

\item Limited access to visual information: Visual information such as diagrams, charts, and graphs can be difficult for screen readers to interpret, which can be a barrier to visually impaired programmers who need to work with this type of information. Awareness of bugs within code is often raised using visual cues, such as squiggly red lines underneath problems. Screen-readers and braille readers often cannot represent visual information pertaining to code or visual cues pertaining to bugs within code. This leads to lower levels of code comprehension amongst visually impaired programmers, and higher numbers of bugs within code produced by visually impaired programmers.

\item Incompatible graphical user interfaces: Graphical user interfaces, such as IDEs, are commonly used by sighted software developers to speed up the development process. Often there is a lack of compatibility between screen reader or braille display software and the graphical elements of these tools \cite{GUIDL}. This puts visually impaired programmers at a significant productivity disadvantage compared to their sighted colleagues, who have unhindered access to productivity boosting features of IDEs \cite{ Mountapmbeme\_Okafor\_Ludi\_2022} and other GUI based tools, such as block-based programming languages such as Scratch \cite{scratch}. Whilst these tools have brought great benefit to the software development and computer science education community, it is important to recognise that the development of GUI based software development and educational tools has increased the accessibility gap between sighted and visually impaired developers. \cite{code\_mirror\_block\_2019}

\item Web accessibility: The web provides many resources to aid programmers in learning computer science concepts, debugging help, and important software documentation. According to the WebAIM Million report \cite{webAIM} , which analyzed the top 1 million home pages on the web in 2021, only about 2.4% of home pages met the Web Content Accessibility Guidelines (WCAG) 2.0 Level AA accessibility standards, meaning that the majority of web pages have accessibility issues to some extent.

\item Harder to edit code without losing focus: Editing code within the original source file can make blind programmers lose context

\end{itemize}

Challenges specific to screen readers:

\begin{itemize}

\item Time: Screen readers can read aloud large amounts of code, but the time taken to listen to this code can be much longer than the time it would take for a sighted programmer to read the same code. This can be a significant issue, as programmers often need to read and understand large amounts of code in a short period of time. Screen readers typically have settings to adjust the speaking speed, but increasing the speed too much can make it difficult to understand the code being spoken. Punctuation is a vital element in the syactic nature of code, whilst this can be represented by a single character which can be quickly read and understood in text or braille format, screen readers deal with punctuation in one of two ways. The first way is skipping over punctuation, when this is done the user may miss out on vital information about the program code and lead to a lower level of code comprehension. The alternative is to explicitly read out punctuation. This method is time-consuming, and putting too much emphasis on punctuation may cloud the users focus, leading to lower levels of code comprehension.

\item Consistency: Different screen readers may have different ways of presenting information and navigating code, which can make it challenging for visually impaired programmers to switch between platforms or collaborate with sighted colleagues

\end{itemize}

Challenges specific to braille displays:

\begin{itemize}

\item Price: Braille displays can be much more expensive than traditional displays, making them less accessible to visually impaired programmers who may not have the resources to purchase them.

\item Low levels of braille literacy: Again, low levels of braille literacy within the visually impaired community restrict the use of braille displays to a small subset of visually impaired people who are proficient in braille.

\item Size: Braille displays can range in size from portable devices with 14-20 cells, to larger desktop models with 40-80 cells. Each cell can represents a character. The ideal braille display for a programmer would be 80 cells, however, larger sizes of braille display come at a much higher cost. Developers using smaller displays may need to read one line of code in multiple lines of braille. Additionally, this may also be the case for users of 80 cell braille displays. Whilst 80 characters per line is accepted as a standard for many programming languages (<https://en.wikipedia.org/wiki/Characters_per_line>) it is often not enforced. This means that reading code that does not comply with this standard will pose a challenge for all braille display users, regardless of the size of their braille display.

\end{itemize}

DISCARDED SECTION

\subsection{Challenges associated with assistive technology}

There are a number of challenges associated with each of the aids used by visually impaired computer users. In the following section we will provide a structured overview of the challenges faced by visually impaired users when using the tools described above to complete programming tasks.

\begin{itemize}

\item \textbf{Screen readers}: One challenge with using screen readers is that they can sometimes struggle to accurately read and interpret complex web pages or applications. This can result in incorrect or incomplete information being conveyed to the user. Additionally, some screen readers may not be compatible with certain software applications or websites, which can limit their usefulness.

\begin{enumerate}

\item Non-visual presentation of information: Screen readers present information using synthesized speech or Braille output, which can make it difficult for visually impaired programmers to get a sense of the structure of the code or the overall context of the program \cite{petrie2014inclusive, lazar2015ensuring}.

\item Limited speed and precision: Screen readers can be slow and imprecise when navigating complex code, which can be frustrating and time-consuming for visually impaired programmers who need to work efficiently \cite{wobbrock2005user, barnard2016coding}.

\item Inadequate support for complex languages: Some screen readers may not be able to fully support complex programming languages, which can limit the ability of visually impaired programmers to work with these languages \cite{petrie2014inclusive, eide2005universal}.

\item Lack of consistency across platforms: Different screen readers may have different ways of presenting information and navigating code, which can make it challenging for visually impaired programmers to switch between platforms or collaborate with sighted colleagues \cite{lazar2015ensuring, barnard2016coding}.

\item Limited access to visual information: Visual information such as diagrams, charts, and graphs can be difficult for screen readers to interpret, which can be a barrier to visually impaired programmers who need to work with this type of information \cite{petrie2014inclusive, rello2017good}.

\item Incompatibility with software applications:

\end{enumerate}

\item \textbf{Braille output device}s: One challenge with using Braille output devices is that they can be quite expensive, which can make them difficult to obtain for some users. Additionally, not all users may be proficient in reading Braille, which can limit the usefulness of this aid.

\item \textbf{Magnification software}: One challenge with using magnification software is that it can sometimes distort images or graphics on the screen, making them difficult to interpret. Additionally, some users may find that the magnified text or other elements are still too small to read comfortably.

\item \textbf{High contrast themes}: One challenge with using high contrast themes is that they can sometimes be overwhelming or difficult to look at for extended periods of time. Additionally, some users may find that the high contrast can actually make it more difficult to distinguish between different elements on the screen.

\item \textbf{Speech recognition software}: One challenge with using speech recognition software is that it can sometimes struggle to accurately interpret voice commands, particularly if the user has an accent or speaks in a non-standard way. Additionally, some users may find that they are uncomfortable speaking aloud or that their voice is not clear enough for the software to accurately interpret.

\end{itemize}

<\^ make this more specific for challenges in reading code with these tools>

The above assistive tools are diverse and provide ways for which visually impaired users to use computers with greater ease, however, these tools do no completely remove the accessibility barriers surrounding software development and coding for the visually impaired \cite{epifani2015designing}.