https://www.mdpi.com/2076-3417/10/12/4214 → Programming Real-Time Sound in Python

Pandey, Maulishree, et al. “Understanding Accessibility and Collaboration in Programming for People with Visual Impairments.” Proceedings of the ACM on Human-Computer Interaction, vol. 5, no. CSCW1, 13 Apr. 2021, pp. 1–30, https://doi.org/10.1145/3449203. Accessed 31 May 2021.

In the paper, it aims to investigate the collaborative experiences and challenges of visually impaired programmers in professional contexts through semi-structured interviews with 22 visually impaired software professionals. It further aims to provide insights and recommendations for improving the accessibility and inclusivity of collaborative programming environments. The information provided by the study can help us understand how to accommodate and create an accessible and inclusive environment for programming. Programming has been deemed relatively accessible as it is text- based, and new assistive tech like graphical user interfaces (GUIs), but there is still more to discover and analyze how those findings can be applied into Word Playpen. The study grounded its focus by looking at the challenges of visually impaired programmers in professional contexts through semi-structured interviews with 22 visually impaired software professionals.

The paper focuses on how assistive technology is used in social settings. There has been research on how accessibility in Human-computer interaction emphasizes the social contexts of assistive technology use, and found that they often lag behind mainstream products and draw unwanted attention. As a result, users have to balance utility with the desire to avoid attention and maintain self-esteem. Thus, it is important to denote that when designing assistive tech one must consider both functional and social scenarios and involve both users with and without disabilities in the design process, which often is overlooked.

There is an emphasis on how research has indicated that people with disabilities face social costs when seeking help, as it can make them appear less competent. Due to that, people with disabilities prefer using external assistance to avoid burdening others. In mixed-ability contexts, accessibility is achieved through collaboration, often requiring people with visual impairments to perform additional work to address accessibility challenges, continually advocating for their needs.

The accessibility challenges that are faced is from being hesitant to seek one’s employer for accommodations. It mainly stems from assistive tech having an expense, or even fearing being seen as making excuses, and it would additionally reflect negatively on their programming ability. The study’s participants preferred to demonstrate their preferred work practices to colleagues to familiarize them with assistive tech and workflows. Through the study, he researches the need for more accessible internal tools as it helps enhance work experience by enabling more efficient work and reducing the need for assistance. Overall, the paper proved the need for accessibility and showed insights on the need for implementing assistive technology as well as working with programmers who have disabilities.

Yee-King, Matthew John, et al. “Automatic Programming of vst Sound Synthesizers Using Deep Networks and Other Techniques.” IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 2, Apr. 2018, pp. 150–159, https://doi.org/10.1109/tetci.2017.2783885. Accessed 26 Mar. 2021.

This thesis investigates the techniques and applications of automatic sound synthesizer programming. It discusses the types of systems such as tone matching programmers and synthesis space explorers. Tone matching programmers take a sound synthesis algorithm and a target sound as input. Synthesis space explorers, provides users with a representation of the synthesizer's sound space, which allows for interactive exploration of the space. It uses Studio tools, autonomous musical agents, and self-reprogramming drum machines.

This paper was much too advanced for me, however I wrote down what I was able to understand. But I think it is a thesis worth revisiting again. But what I was able to understand from the conclusion, the thesis delves into the concept of automatic sound synthesizer programming, which is aimed at removing the need for users to specify parameter settings explicitly. It believed that preset banks, presentation of synthesizer sound space, and search algorithms to find specific target sounds, would help remove the need. It also discussed the Integration of automated timbral exploration into a drum machine enhances creativity for musicians. I believe the thesis explores the implications and impacts of sound systems, but at the moment my knowledge is lacking to fully understand the scope of what this thesis is trying to explain.

Pires, Ana Cristina, et al. “Exploring Accessible Programming with Educators and Visually Impaired Children.” Proceedings of the Interaction Design and Children Conference, 21 June 2020, https://doi.org/10.1145/3392063.3394437. Accessed 13 Mar. 2023.

This thesis explored the emergence of computational thinking as a discipline in schools, which further emphasized the importance of it beyond just computing context. It believes that through the use of visual programming environments such as Scratch and Blocky it will help promote computational thinking to enhance children’s abilities and prepare them for programming in the future. However, they found out that the tools provided were not accessible to visually impaired children. The paper proposes new approaches to address the lack of accessibility, and explore more opportunities for spatial activities. The paper focuses on two of the studies that they did.

The first study explores current approaches to promote computational thinking environments for visually impaired children putting a focus on spatial programming activities. The different environments that they investigated were fully virtual ones, virtual environments with tangible output, tangible environments with virtual output, and fully tangible environments. Though schools (Portugal) have adopted it, these environments are still not fully accessible to visually impaired children. As such, the study gathered a focus group with special needs educators and information tech instructors from inclusive schools. The participants were asked to discuss the qualities and limitations of the environments and further explore avenues to make them more accessible. The researchers found the importance of using robots, tangible blocks, boards, and maps to help programming activities for visually impaired children. The participants also believed that there needs to be more tactile feedback, auditory cues, and Braille inscriptions in order to enhance accessibility for the children. As such, the study saw that in order to design an accessible programming environment, one must engage robots with feedback mechanisms, tactile-rich maps for spatial perception, and tangible blocks with sensory representations. Overall, the participants did feel enthusiastic about using these new environments with visually impaired children but further stressed the importance of making them more accessible through sensory enhancements.

The second study explored the adaptation of solutions from Study 1 in order to create accessible programming environments for visually impaired children. They mainly focused on using tangible blocks and a robot with augmented physicality. The study worked with seven visually impaired children in a workshop, and later analyzed them through thematic analysis and validation with educators. They implemented tangible blocks and a robot with augmented physicality to facilitate programming. The robot called DASH was chosen for its existing usage in schools, however modifications were made such as adding tactile cues and audio feedback to the blocks and robot actions. The researchers wanted to make sure that the workshop was unstructured and just had goal-directed spatial activities. As a result, children were more excited to participate and had better interactions that helped perceive agency in controlling the robot. Through the workshop's analysis, it revealed that the children were able to understand the programming concepts, and looked like they wanted to learn more. Though successful, the educators put focus on the importance of using step-by-step instructions and real life context that will help with learning. During the workshop, the researchers also saw how children naturally collaborated and engaged with the robot through the use of tangible elements that helped sharing and exploration.

The researchers also looked at the children's cognitive development, and found that older children showed more intentional movements and debugging skills compared to younger children, which indicated the development of abstract thinking. They also looked at spatial cognition and saw how it was enhanced through activities involving the robot's movement, promoting spatial orientation and conceptual understanding. The setup of the workshop proved to be beneficial for visually impaired children, improving their spatial cognition, mental rotation, and navigation skills. The educators from study 1 recognized the potential of tangible programming environments to reinforce existing educational goals and promote inclusive learning. They identified opportunities to integrate programming activities with other subjects like math and science. Collaboration was seen as a crucial aspect, facilitated by tangible elements and spatial activities. The researchers also looked at the limitations of the study, and saw how they focus solely on visually impaired children and the novelty effect of the activities. However, the study offers valuable insights for researchers, developers, and educators to develop inclusive programming environments and foster collaborative learning among children with mixed abilities.

The educators in the previous study recognized the potential of tangible programming environments to reinforce existing educational goals and promote inclusive learning. They identified opportunities to integrate programming activities with other subjects like math and science. Collaboration was seen as a crucial aspect, facilitated by tangible elements and spatial activities. The researchers also looked at the limitations within the research. One of them was just focusing exclusively on a group of visually impaired children, but it did not include sighted children. This in turn restricts the understanding of how these activities might function in a fully inclusive classroom setting where children of mixed abilities interact. Additionally, the study was only conducted in a single session, which doesn’t show how children might behave in a long-term setting. Another limitation is lack of diverse educational context, it limited the generalizability of the findings to different educational contexts or settings. Different schools or cultural environments may get different results.

Overall, I think this paper helps understand the importance of providing accessible tools in programming languages. Not only do programmers need it, but as Wordplay also wants educators to use it, it is important that the assistive technology we are using can also help include children with disabilities that want to learn programming languages.

Romano, Simone, et al. “The Effect of Noise on Software Engineers’ Performance.” ArXiv (Cornell University), 11 Oct. 2018, https://doi.org/10.1145/3239235.3240496.

In this paper it doesn’t particularly talk about the accessibility with programmers who vision impairments but more or else shows the effects of noise on programming. It looks into different theories of noise effects on performance. The theories consist of, Arousal Theory, Composite Theory, and Maximal Adaptability Theory.

Broadbent's Arousal Theory explains noise effects through an arousal-induced attentional narrowwing mechanism. Meaning that the noise can increase arousal helping intially with exclusig irrelevant cues, and imporving performance. However, it doesn’t include beyond the optimal arousal level, so there is no way of knowing when performance declines as relevant cues are excluded. Moreover, the theory believes that noise intensity and duration influence one’s performance, with intermittent noise causing more impairment than continuous noise.

Poulton's Composite Theory believes that noise degrades performance when it is masked with inne speech, which is a crucial for task performance. They believe continuous noise can increase arousal, offsetting masking effects, but over time, arousal decreases, and masking dominates, impairing performance. Noise effects are similar across tasks and types but vary with intensity, duration, and schedule.

The Maximal Adaptability Theory believes that theb stress from noise affects performance through input (environmental factors like noise), adaptation (individual coping mechanisms), and output (task performance). Noise impairs performance by masking relevant auditory information. Individuals adapt to varying stress levels, but beyond a threshold, performance declines.

These theories all suggest that noise effects on performance depend on the nature of the task, the characteristics of the noise, and individual adaptation mechanisms.

The study evaluates the effect of noise on software engineering tasks through two controlled experiments. Noise negatively impacted fault fixing but not the comprehension of functional requirements. This indicates that tasks requiring more cognitive resources, like fault fixing, are more susceptible to noise, highlighting the need for quieter work environments for such tasks.

I am not sure yet how I could connect this with WordPlay, but it give me a deeper understanding on the effects of noise in engineering work environments.

Sánchez, Jaime, and Fernando Aguayo. Blind Learners Programming through Audio. 2 Apr. 2005, https://doi.org/10.1145/1056808.1057018. Accessed 30 Mar. 2024.

The paper discusses the efforts to make programming more accessible to end-users which include languages like Basic, Logo, Smalltalk, Pascal, and others. Those programming languages have improved beginners skills by using user interface principles. However, the languages are not accessible to visually impaired learners. Many studies have shown that audio-based applications are able to enhance cognitive skills to blind children, which focus on 3D audio interfaces for spatial and abstract reasoning.

The Audio Programming Language (APL) was developed to aid blind novice programmers by simplifying syntax and enhancing problem-solving and thinking skills through audio interfaces. The APL uses a circular command list and query system, making programming accessible without requiring memorization of commands. It features a dynamic command list and unconventional variables to store sounds, facilitating interaction with the machine.

APL underwent usability testing with expert and beginner users, which revealed initial functionality issues and showed that blind learners eventually grasped programming concepts through concrete experience and interaction. Learners created simple and complex programs, demonstrating increased understanding and enthusiasm. The study indicates that audio interfaces can help blind learners develop algorithmic thinking and cognitive skills, suggesting the need for further research on how blind users map the programming process differently from sighted users.

Though this paper didn't show that much insight, it helped understand what other people are doing in order to make programming languages more accessible, and show that it is possible.

Howard, A. M., et al. “Using Haptic and Auditory Interaction Tools to Engage Students with Visual Impairments in Robot Programming Activities.” IEEE Transactions on Learning Technologies, vol. 5, no. 1, 2012, pp. 87–95, https://doi.org/10.1109/tlt.2011.28.

The paper first recognizes the number of college freshmen with disabilities has been increasing, with vision impairments accounting for 16% of these students. However, only 3.9% of disabled students major in computer science. This disparity is largely due to inadequate pre-college math and science education, which is foundational for computing degrees. Approximately 11% of children aged 6 to 14 have disabilities, but they take fewer science and math courses than their peers, often due to inaccessible information and unfamiliarity with nonvisual teaching methods.

There isn’t much effort being made to engage visually impaired students in computing at the precollege level. There have been some initiatives such as the National Center for Blind Youth in Science, the AccessComputing Alliance, and Project ACE. Conversely, robotics appeals broadly to students, including those with disabilities. However, the lack of accessible interfaces for educational robots means visually impaired students often cannot participate equally in robot-based computing activities. Most robot programming interfaces rely on visual and keyboard-based inputs, which are unsuitable for many visually impaired students.

The research in the paper focuses on creating accessible interfaces for robot programming to engage visually impaired students. Their goal is to leverage the appeal of robotics to involve students with disabilities towards computing, hypothesizing that alternative interface technologies will enable active participation and encourage future interest in computing.

The Programming/Robot Interaction System uses a lot of calculus which I am still working on learning. But I believe that it will help me understand the code they used and how it will inspire me to implement it on Word Playpen.