Evaluation of Low-threshold Programming Learning Environments for the Blind and Partially Sighted

Shirin Riazy[1], Sabrina Inez Weller[2] and Katharina Simbeck[1]
[1]*HTW Berlin, Germany*
[2]*BIBB, Bonn, Germany*

Keywords: Educational Programming Language, MINT, Teaching for the Visually Impaired, Scratch.

Abstract: Approximately 1.3 billion people worldwide live with visual impairments and are therefore dependent on the use of aids. With regard to the acute shortage of skilled workers in the IT sector, the question arises as to the extent to which people with visual impairments can enter the IT professions. While web-based programming environments have been used to introduce children to programming, their accessibility for the blind and visually impaired is questionable. This report introduces 17 web-based programming learning environments and examines their usability for blind and visually impaired people. In total, none of the 17 graphical programming environments were suitable for the usage by blind and visually impaired people.

1 INTRODUCTION

Worldwide there are approximately 1.3 billion people living with visual impairments, 36 million of whom are blind (WHO, 2019). At the end of 2017, around 7.8 million people with severe disabilities were living in Germany. Compared to the population as a whole, this was equivalent to approximately 1 in every 11 German citizens being severely disabled (9.4 percent). Blind and partially sighted people comprised just under 5% of all people with a severe disability in 2017 (Destatis, 2017)[1]. While almost 21% of these suffered from blindness or the loss of both eyes, just under 14% suffered from severe visual impairment. Almost 66% had another form of partial sightedness. As we grow older, the risk of losing our sight increases. More than half of the blind and partially sighted in Germany are 75 years old. General illnesses are the most frequent cause of sight loss or partial sightedness, accounting for 90% of all cases. In only just under 3% of cases, those affected were blind or partially sighted at birth. Around 25% of all blind and partially sighted people also suffer from impaired functionality of internal organs or organ systems. With the number of blind and partially sighted people having fallen over the last 25 years, a study now suggests the figure is set to rise in future. In fact, the proportion of those affected is set to rise by almost 6 percent by 2020 (Ackland et al., 2017). According to this study, the anticipated increase in diabetes cases and the ageing population is likely to cause increased eye disease and visual problems (IAPB, 2018).

In recent years there has been a substantial increase in the employment rate of people with a disability (Aktion Mensch, 2019). However, people with a severe disability state in surveys that the work demands placed on them are too high and that they perceive progression opportunities as poorer compared to those of people without a disability. Just under one third of blind people and people with severe visual impairment of employable age are actually in employment. Previous studies have shown that blind people and people with severe visual impairment[2] are particularly disadvantaged in the job market (Lauenstein et al., 1997; Schröder, 1997). This is due on the one hand to the limited range of activities as a result of the disability. In addition, a large number of typical

[1]Between 2007 and 2017, the total number of blind or partially sighted people in Germany rose by almost one percentage point.

[2]The generic term ”visual impairment” is often used to describe blindness and partial sightedness. Under German social legislation, anybody with vision of less than 2% in their better eye is regarded as blind. For people with severe visual impairment, their vision is reduced to between 2 and 5% of the norm. Those affected can be treated the same way as the blind people. For the partially sighted vision in the better eye —despite the visual aids such as glasses or contact lenses—is a maximum of 30%.

Riazy, S., Weller, S. and Simbeck, K.
Evaluation of Low-threshold Programming Learning Environments for the Blind and Partially Sighted.
DOI: 10.5220/0009448603660373
In *Proceedings of the 12th International Conference on Computer Supported Education (CSEDU 2020) - Volume 2*, pages 366-373
ISBN: 978-989-758-417-6

"professions for the blind" such as telephonist, massage therapist, balneotherapist, and shorthand typist have in some cases become obsolete and in others more complex. However, new professional opportunities for the blind and the severely visually impaired have also emerged[3]. This is particularly the case for people with sight loss and partial sightedness who are skilled IT workers (e.g. in the areas of application development and system integration) and information technology specialists in the area of database development, for whom opportunities on the labour market have been very good over the last 10 years. The placement rate in this area of supported vocational education and training is almost 80%. Besides the good opportunities on the labour market and attractive remuneration for the partially sighted, the IT sector represents an interesting area of work not least due to the anticipated levels of accessibility. Many of those affected who opt to train or study in this area see it as their social duty to play a part themselves in creating and advancing digital accessibility. Another reason why people with sight loss and the partially sighted often opt for the IT sector is that, due to their limitations, those affected are more frequently forced to deal with digital technologies. For this reason, when dealing with digital technologies they very quickly develop a natural acceptance and greater competence. The dividing line when dealing with the medium of the PC is not between the sighted and the partially sighted, but instead between the partially sighted and the blind. While the partially sighted use technologies such as the Microsoft magnifier which generally allows them to work in the way sighted people work, these tools do not function for people with a severe visual impairment or sight loss. Screen readers are therefore used by these people (mainly Jaws and NVDA). In terms of IT skills, marked differences therefore exist between the partially sighted and the blind.

Because of the general demand for IT-skilled personnel (Pauly and Holdampf-Wendel, 2019), a number of gateways to typical IT problems have been developed. One such attempt is the development of education-oriented programming languages, which are intended to support the first steps in programming (Grover, 2015), specifically for children (Baron, 2014). While these web-based programming environments have been widely successful, their accessibility for the blind and visually impaired is questionable. In the following, education-oriented programming languages are introduced (while distinguishing graphical and tangible programming), then, guidelines for the verification of accessibility are presented, which are then used to evaluate 17 programming environments.

[3]The "network for the professional participation of blind and partially sighted people" has put together 58 qualifications for the administration activities, telecommunication, commercial occupations, the skilled trades, landscaping, healthcare and artistic occupations. There is also training at universities.

2 EDUCATION-ORIENTED PROGRAMMING LANGUAGES

Many of the common programming learning environments are based on simplified programming languages, where the syntax is reduced to the essentials. With semantic, syntax and type errors being typical mistakes among novice programmers (Altadmri and Brown, 2015), this allows beginners to focus on semantic properties but also decimates the possibilities. The text-based, graphical or tangible programming languages and environments presented in this chapter were found through:

- literature reviews of novice programming environments, mostly comparing text-based and block-based programming environments, such as (Xu et al., 2019),
- a web-based search, where websites (such as (Baron, 2014)) were used to find easily accessible programming environments for children.

Although introductory courses exist for most of the commonly used programming languages (such as Java and Python), several text-based programming languages and environments were specifically designed for novices. One famous example is BASIC, which stands for "Beginner's All-purpose Symbolic Instruction Code" and was developed in the 1960s (Kemeny and Kurtz, 1971). BASIC has since been widely used as an introductory programming language due to its simple syntax (Kemeny and Kurtz, 1971). Many so-called "dialects" of the programming language were developed, including Visual Basic and VB.NET, which belong to the most commonly used programming languages (Stack Overflow, 2019).

Other examples of novice programming languages include Lisp and Logo, which is a variant of Lisp. These languages were developed at MIT in the 1950s (Papert, 1978; Winston and Horn, 1986). Whereas Lisp was mainly developed for the processing of lists, Logo was developed especially for the introduction into programming paradigms for children. Variants of both programming languages are still used today.

Furthermore, programming environments targeting children as novice programmers were developed.

KidsRuby (The Hybrid Group, 2011), for example, is a programming environment for children and teenagers that uses a simplified version of the Ruby programming language. An exemplary code is displayed in Listing 1.

Listing 1: An example of KidsRuby code.

```
Turtle.draw do
    background black
    pencolor red
        100.times do
        distance = rand(100)
        turnright rand(25)
        forward distance
        backward distance
    end
end
```

The commands that can be implemented in KidsRuby can be used without instructions, the syntax includes for and while loops, but without the need to implement variables.

In the next subsections, we will introduce graphical and tangible programming languages and environments, to assess their extent of accessibility for the blind visually impaired in the following sections.

2.1 Graphical Programming Languages

Building on top of the text-based programming languages (such as Logo), new languages and programming environments have been developed specifically in order to further ease the access to programming. More so than text-based programming environments, graphical programming environments have been found to appeal different students in primary and secondary education (Papadakis et al., 2019). The so-called graphical programming languages represent an attempt to reduce the susceptibility to errors in the syntax of programming languages by using ready-to-use building blocks. It is inconclusive, however, whether they should be preferred in comparison to text-based programming languages (Price and Barnes, 2015; Xu et al., 2019)

As one of the most prominent representatives of graphical programming languages, Scratch was developed and published in May 2007 by a team from the MIT Media Lab (Resnick et al., 2009). Scratch is a visual, lisp-based programming environment for children and teenagers that has approximately 40,000 users per day (Scratch, 2013).
As can be seen in Figure 1, the syntax contains blocks (so-called scratch blocks) that are joined together by drag & drop. The different colors and shapes symbolize different command groups (e.g. movements in animation or sounds) and their modality (e.g. loop, definition, if query). Scratch is cloud-based and runs without installation in the browser. Spin-offs running as apps (such as Hopscotch for iPhones and iPads (Hopscotch Technologies, 2019)) were developed as alternatives. Other graphical programming environments that were inspired by Scratch, or represent extensions, include:

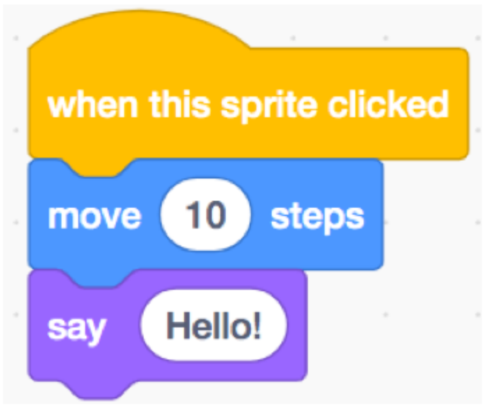

Figure 1: Sample implementation in the scratch environment.

- Snap! (BYOB): Snap! (Moenig and Harvey, 2009), which used to be called “Build Your Own Blocks” was developed by Jens Mönig and Brian Harvey (Harvey, 2019) at the University of California, Berkeley, in 2009 (Scratch, 2013). Snap! is an extension of Scratch in the way that it offers more complex functions and concepts for implementation.
- OpenRoberta: OpenRoberta (IAIS, 2014) is an open source programming environment at the center of the Roberta Initiative of Fraunhofer IAIS, Sankt Augustin (IAIS, 2014). The programming language NEPO used in OpenRoberta, which was designed based on Scratch, is developed as Open Source and is suitable for use with robots such as EV3, Calliope, Arduino, etc..
- MIT App Inventor: Originally developed by Google and later adopted by MIT, MIT App Inventor is a graphical programming environment for developing apps (MIT Media Lab, 2010). The environment consists of two layers (design and block editor) representing the front and back end of app development. Both levels were developed in the style of Scratch and use a syntax of building blocks that are inserted by drag & drop.

Other programming environments worth mentioning are Alice, a block-based programming environment for creating 3D animations (Alice, 1995) and Kodu (Microsoft Research, 1999), which is also implemented in a 3D environment using building blocks. Their goal is primarily the development of games. Even though the graphical programming environments bear similarites, they mostly target different age groups, or programmable features (Kaya and Yıldız, 2019). As for their fitness for the blind and visually impaired, this will be systematically evaluated using criteria defined in Section 4.

2.2 Tangible Programming Languages

Tangible programming, meaning programming based on haptic objects, began at the MIT AI Lab. Using the programming language Logo, Papert and his colleagues developed a hemispherical robot (”TORTIS”) that could be controlled with simple commands such as forward, backward, left, right (Perlman, 1974). Based on these results, Perlman developed new user interfaces and designed prototypes in box form, with haptic buttons and levers

One branch of tangible programming consists of programming robots. Several robots can be programmed by simulating certain sequences. Topobo (MIT Media Lab, 2003) for example is a robot with two states (active/passive) in which motion sequences can be stored and executed. The shape of the robot is shown in Figure 2. Another example is Curlybot (MIT Media Lab, 1999), a hemispherical robot developed by MIT Media Lab, which stores and reproduces movements.

Other variants of haptic programming languages include

- Primo (Cubetto, 2017): Cube-shaped wooden robot that can be programmed on a control board using pluggable coding blocks.
- Tangible Programming Bricks (McNerney, 1999): Haptic blocks with different functions that are 1) plugged together to implement or 2) equipped with parameter cards (see Figure 3 for an illustration of the haptic programming blocks).
- Tern (HCI Tufts, 2008): Different blocks of wood are plugged together for programming. The conceptual structure of the programming language is very similar to that of Scratch and it can be used to create programs for robots of the LEGO Mindstorms RCX series.
- Strawbies (Hu et al., 2015): Haptic puzzle pieces are put together to create a virtual simulation.
- Cubelets (Cubetto, 2017): Modular robot designed by selecting its components.

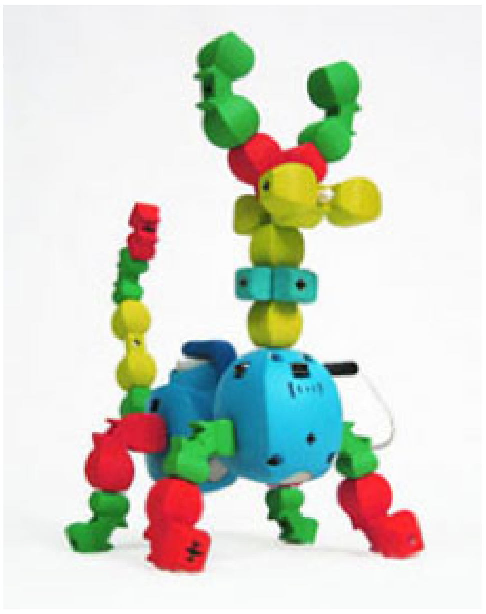
Figure 2: Topobo, a robot that can be programmed by examples. The image was taken from the official homepage (MIT Media Lab, 2003).

- Note Code (Kumar et al., 2015): Puzzle tool with musical function, to teach programming paradigms.

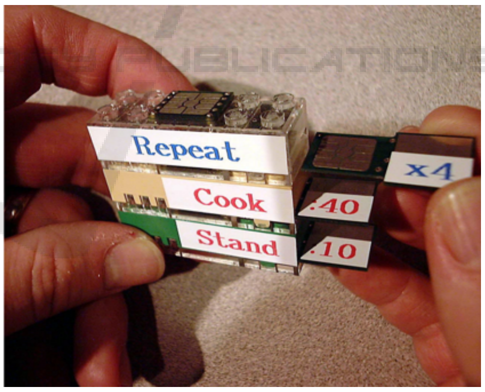

Figure 3: Tangible Programming Blocks. Figure by McNerney (McNerney, 2004).

3 WEB CONTENT ACCESSIBILITY

The World Wide Web Consortium (W3C)[4] is an international community that develops standards for the accessibility of web content. In order to be accessible, web content must be:

[4]www.w3.org

- **Perceivable:** text alternatives available, description of elements available, sufficient color contrasts
- **Operable:** usable without mouse, content can be viewed for a sufficiently long amount of time, no content, which might cause seizures or physical reactions
- **Understandable:** structured, e.g. with titles, easy language
- **Robust:** compatible with commonly used assistive technology

To test webpages for the conformity with the accessibility standard, several tests have been developed (BIK, 2019; ACT task force, 2019). Depending on the testing procedure and the web content, conformity levels from A to AAA with the Web Content Accessibility Guidelines can be earned (Consortium, 2019).

In order to make web content accessible, several aiding tools exist. Perhaps the most commonly known tool for aiding people with blindness and visual impairments are Braille fonts or displays. This font displays letters as tactile representations, using *raised dots* in rectangular block-formation. Other formats are Braille writers, notetakers or embossers, where notes can be taken in the Braille font.

However, when it comes to the best fitting tools, people distinguish between blindness and visual impairment. A group of software aimed at aiding the visually impaired are magnification softwares, which magnify content on the screen. Examples are products made by Google[5], MAGic[6] or SuperNova[7]. Furthermore, video magnifiers are electronic visual aids, which record writings using a camera and display them strongly enlarged on the screen.

For the blind, several software products are commonly used as aids in order to make computer- and web-content accessible. Among them are screenreaders, which read the content on the screen and either read it out loud or display it on a Braille display. Examples of such software include JAWS[8], the openly available NVDA[9] and SuperNova[10].

[5]https://www.google.com/accessibility/. Last accessed 30.10.2019

[6]https://www.freedomscientific.com/products/software/magic/. Last accessed 30.10.2019

[7]http://www.dolphin-de.de/produkte/dolphin/supernova-magnifier-screenreader/index.html. Last accessed 30.10.2019

[8]https://www.freedomscientific.com/products/software/jaws/. Last accessed 30.10.2019

[9]http://meinnvda.de/. Last accessed 30.10.2019

[10]http://www.dolphin-de.de/produkte/dolphin/supernova-magnifier-screenreader/index.html. Last accessed 30.10.2019

In the IT sector, the blind and partially sighted are generally trained using common text-oriented programming languages such as Java, Sharp and SQL. The web languages PAP and Framework are also learned. These languages are very accessible because they consist of understandable texts. The linear progression of the language is like the output of the screen reader. By contrast, for example, C poses a particular challenge for the blind because the commands produce no readable words due to the symbols. This means the symbols cannot be read out by the screen reader and therefore mastery of the Braille display is required.

4 SYSTEMATIC EVALUATION

To evaluate programming environments for beginners and children, we have distinguished 6 categories, in which the programming languages will be examined, in order to find their measure of accessibility.

Alt-text. Using the Google Chrome extension "Alt Text Tester", it was checked whether all of the images had alternative text descriptions.

Font Size. It was checked to see whether the font size can be enlarged without endangering the structure and make-up of the website.

Keyboard Usable. It was checked to see whether the website is navigable using only the keyboard via prompts defined in (Accessibility Developer Guide, 2018).

Contrast. Contrast of at least 1:3 for important items (text, graphics) was checked via the Color Contrast Analyzer extension for Google Chrome in accordance to the BITV testing procedure (BIK, 2019).

Content. It was determined, whether the language of a website could be automatically detected using a web service[11]. This may be necessary for reading software.

Robustness. To ensure that all viewers of a website can display and process it correctly, it is very important to specify the character encoding ('charset') correctly. False encoding can cause problems for assistive technology. The robustness was checked using the W3C validation service[12], as recommended by (Faulkner, 2019) and returns the errors in the character encoding.

[11]https://translate.google.com

[12]https://validator.w3.org/

Table 1: Testing procedures for the accessibility of programming environments for children and beginners. Some of these environments were only accessible through an application that could not be tested for robustness. In these cases, the robustness is unknown.

Language	Alt-text	Font size	Keyboard usable	Contrast	Content	Robustness
KidsRuby	✗	✓	✗	✓	✓	application
Scratch	✗	✓	✗	✗	✓	1 error
Hopscotch	✗	✓	✗	✗	✓	application
Snap!	✓(no images)	✗	✗	✓	✓	5 errors
OpenRoberta	✗	✓	✗	✗	✓	285 errors
MIT App Inventor	✗	✓	✗	✗	✓	6 errors
Blockly	✗	✓	✗	✗	✓	(not found)
Alice	✗	✗	✗	✓	✓	application
Kodu	✗	✗	✗	✗	✓	application
Swift Playgrounds	✗	✗	✗	✓	✓	application
Kodable	✓	✓	✗	✗	✓	9 errors
Thimble moz://a	✓(no images)	✓	✗	✗	✓	7 errors
Code Monster	✗	✓	✗	✗	✓	22 errors
Gameblox	✗	✓	✗	✓	✓	10 errors
Code.org	✗	✓	✗	✗	✓	54 errors
Tynker	✗	✓	✗	✗	✓	48 errors

These categories were chosen as a minimal representation of the 60 test items of the BIK (BIK, 2019), which is a test procedure for the comprehensive testing of the accessibility of websites. These criteria are all required for the lowest level of accessibility (A) and may easily be replicated on any typical computer using open online resources. Since these conditions were directly picked from the BIK items, any web content, which fails to fulfill these conditions, will definitely not be classified as accessible by the larger catalog of requirements.

5 RESULTS AND DISCUSSION

The advancing digitalisation can represent an opportunity but also challenge for the blind and for the partially sighted. A key criterion in this regard is accessibility. If the trend towards accessibility is continued in the context of the digitisation process, this may result in very good future opportunities in terms of labour market integration for the blind and partially sighted as a group. However if the opposite trend continues, the blind and partially sighted will ultimately be largely excluded from digital work processes.

The IT sector has for some years already been offering a range of employment opportunities for the partially sighted and for the blind. Besides the activity of programming, working with databases is also possible for those concerned. Working in the area of testing is generally also possible provided the testing tools used are accessible. Some areas such as software planning continue to represent a risk of exclusion due to the two-dimensional graphical language standards which is virtually impossible to represent in text.

Within the scope of this evaluation, 17 computer-based and 6 other programming languages were presented and evaluated. Most of these were evaluated using the criteria defined in Section 4, which were introduced as a minimal check for accessibility of web content. The results are visible in Table 1.

Most of the programming IDEs were implemented in a responsive format, allowing the font size to vary flexibly. The content was suited for children, with easy and clear language that could be identified programmably. The programming environments that were tested were not completely navigable by the keyboard and thus not deemed operable. This includes the web-based, as well as the locally operating software. In particular, the widely used graphical programming environments based on drag-and-drop blocks used in Scratch and Scratch-like IDEs are not suitable for the blind and visually impaired. Apart from missing alt-text descriptions, the sites were not operable using only the keyboard. Therefore, our current findings indicate that the widely used (web-based) novice programming environments are currently not suited to be used by the blind and visually impaired.

Other alternatives, such as robots, can reproduce movement, and may be used by people with visual impairments, but they offer little leeway in programming their own functions. Classic programming paradigms, such as for and while loops, cannot be learned from these robots. The other haptic programming languages (e.g. Tern and Tangible Programming Bricks) could, in general, be used by blind and visually impaired people. In their currect form, however, the blocks would have to be supplemented with Braille lettering or other haptic features to identify the individual functions.

In principle, the current investigation has also shown that classical programming languages (such as PAP and Framework) may be more accessible as starting points for beginners, since the syntax consists of understandable texts and may be used in combination with tools, such as a screen reader. However, low-threshold programming environments for novices, who are blind or visually impaired, are scarce.

6 CONCLUSION

In total, none of the 17 graphical programming environments were suitable for the usage by blind and visually impaired people. Some text-based programming languages were deemed more accessible as a starting point for programming. However, in recent decades many approaches to haptic programming have been published which may be extended for use by blind people in the coming years. Due to the great success of low-threshold graphical programming languages with sighted people, corresponding offers should also be developed for blind and visually impaired people. This might pave the way for IT skills to be developed later in life.

ACKNOWLEDGEMENTS

We thank Ottfried Altfeld, Head of the Department for Participation and Vocational Education and Training at blista, the German Institute for the Blind, for their friendly support. This contribution was funded within the framework of the BMBF joint projects “Learning Analytics und Diskriminierung” and “DABEI: Digitalisierung in der betrieblichen Ausbildung von Menschen mit Behinderung”.

REFERENCES

Accessibility Developer Guide (2018). How to browse websites using a keyboard only. https://www.accessibility-developer-guide.com/knowledge/keyboard-only/browsing-websites/. Accessed: 2019-11-02.

Ackland, P., Resnikoff, S., and Bourne, R. (2017). World blindness and visual impairment: despite many successes, the problem is growing. *Community eye health*, 30(100):71.

ACT task force (2019). ACT testing procedure. https://github.com/w3c/wcag-act/. Accessed: 2019-11-01.

Aktion Mensch (2019). Inklusionsbarometer Arbeit 2019. http://atlas.iapb.org/global-action-plan/gap-indicators/#web-indicators. Accessed: 2020-01-15.

Alice (1995). Alice – Tell Stories. Build Games. Learn to Program.

Altadmri, A. and Brown, N. C. (2015). 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 522–527.

Baron, S. (2014). 20 Resources for Teaching Kids How to Program & Code. https://www.apartmenttherapy.com/20-resources-for-teaching-kids-how-to-program-code-200374.

BIK (2019). BITV test. https://www.bitvtest.de/start.html. Accessed: 2019-11-01.

Consortium, W. W. W. (2019). WCAG conformance. https://www.w3.org/TR/WCAG20/#conformance. Accessed: 2019-11-01.

Cubetto (2017). Cubetto: Ein Roboter, der Kinder ans Programmieren heranführt. https://www.primotoys.com/de/. Accessed: 2019-11-01.

Destatis (2017). Statistik der schwerbehinderten Menschen - Kurzbericht - 2017. page 33.

Faulkner, S. (2019). WCAG 2.1 parsing error bookmarklet. https://developer.paciellogroup.com/blog/2019/02/wcag-2-0-parsing-error-bookmarklet/. Accessed: 2019-11-02.

Grover, S. (2015). Learning To Code With Your Kids.

Harvey, B. (2019). HomePage for Brian Harvey (bh@cs.Berkeley.EDU). https://people.eecs.berkeley.edu/~bh/. Accessed: 2019-11-01.

HCI Tufts (2008). Tern - Tangible Programming. http://hci.cs.tufts.edu/tern/. Accessed: 2019-11-01.

Hopscotch Technologies (2019). Hopscotch. https://www.gethopscotch.com/. Accessed: 2019-11-01.

Hu, F., Zekelman, A., Horn, M., and Judd, F. (2015). Strawbies: explorations in tangible programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 410–413. ACM.

IAIS (2014). Open Roberta Lab. https://lab.open-roberta.org/.

IAPB (2018). IAPB Atlas: Global Action Plan Indicators – the data in full. International Agency or the Prevention of Blindness. http://atlas.iapb.org/global-action-plan/gap-indicators/#web-indicators. Accessed: 2020-01-15.

Kaya, K. Y. and Yıldız, İ. (2019). Comparing three free to use visual programming environments for novice programmers. *Kastamonu Eğitim Dergisi*, 27(6):2701–2712.

Kemeny, J. G. and Kurtz, T. E. (1971). *Basic programming*. Wiley New York.

Kumar, V., Dargan, T., Dwivedi, U., and Vijay, P. (2015). Note code: A tangible music programming puzzle tool. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 625–629. ACM.

Lauenstein, T., Ritz, H.-G., and Sürth, B. (1997). Sicherung und Förderung der beruflichen Eingliederung Blinder und Sehbehinderter auf PC-gestützten Büroarbeitsplätzen. *Mitteilungen aus der Arbeitsmarkt-und Berufsforschung*, 2:514–521.

McNerney, T. S. (1999). *Tangible programming bricks: An approach to making programming accessible to everyone*. PhD Thesis, Massachusetts Institute of Technology.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, 8(5):326–337.

Microsoft Research (1999). Kodu | Home. https://www.kodugamelab.com/. Accessed: 2019-11-01.

MIT Media Lab (1999). Curlybot. https://tangible.media.mit.edu/project/curlybot/. Accessed: 2019-11-01.

MIT Media Lab (2003). Topobo construction kit with kinetic memory. http://www.topobo.com/. Accessed: 2019-11-01.

MIT Media Lab (2010). MIT App Inventor | Explore MIT App Inventor. http://appinventor.mit.edu/explore/. Accessed: 2019-11-01.

Moenig, J. and Harvey, B. (2009). Snap! (Build Your Own Blocks) 4.2. https://snap.berkeley.edu/. Accessed: 2019-11-01.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., and Zaranis, N. (2019). The appropriateness of scratch and app inventor as educational environments for teaching introductory programming in primary and secondary education. In *Early Childhood Development: Concepts, Methodologies, Tools, and Applications*, pages 797–819. IGI Global.

Papert, S. (1978). Interim report of the logo project in the brookline public schools: An assessment and documentation of a children's computer laboratory. In *Artificial Intelligence Memo No. 484*.

Pauly, B. and Holdampf-Wendel, A. (2019). Erstmals mehr als 100.000 unbesetzte Stellen für IT-Experten. https://www.bitkom.org/Presse/Presseinformation/Erstmals-mehr-als-100000-unbesetzte-Stellen-fuer-IT-Experten. Accessed: 2020-03-12.

Perlman, R. (1974). Tortis (toddler's own recursive turtle interpreter system).

Price, T. W. and Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the eleventh annual international conference on international computing education research*, pages 91–99.

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., and Silver, J. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11):60. Accessed: 2019-11-01.

Schröder, H. (1997). Die Beschäftigungssituation von Blinden: ausgewählte Ergebnisse einer Befragung bei Blinden und Unternehmen (The employment situation of the blind). *Mitteilungen aus der Arbeitsmarkt-und Berufsforschung*, 30(2):502–513.

Scratch (2013). Jens Mönig am Scratch Day 2013 – Das deutschsprachige Scratch-Wiki. https://scratch-dach.info/wiki/Jens_M%c3%b6nig_am_Scratch_Day_2013. Accessed: 2019-11-01.

Scratch (2013). Scratch Statistics | Scratch Usage Statistics. https://web.archive.org/web/20130527044039/http://stats.scratch.mit.edu/community/. Accessed: 2019-11-01.

Stack Overflow (2019). Stack Overflow Developer Survey 2018. https://insights.stackoverflow.com/survey/2018/?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2018&utm_content=social-share.

The Hybrid Group (2011). Kidsruby.com. http://kidsruby.com/. Accessed: 2019-11-01.

WHO (2019). Vision impairment and blindness.

Winston, P. H. and Horn, B. K. (1986). Lisp.

Xu, Z., Ritzhaupt, A. D., Tian, F., and Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: a meta-analysis study. *Computer Science Education*, 29(2-3):177–204.