TabGO: Towards Accessible Computer Science in Secondary School

Ken H. Andriamahery-Ranjalahy[1](✉), Léa Berquez[2], Nadine Jessel[3], and Philippe Truillet[2]

[1] Musicology Department, Univ. Toulouse II Jean Jaurès (LLA-CREATIS), Toulouse, France
ken.andria@univ-tlse2.fr

[2] CS Department, Univ. Toulouse III Paul Sabatier (IRIT), Toulouse, France
lea.berquez@univ-tlse3.fr, Philippe.Truillet@irit.fr

[3] INSPE, Univ. Toulouse II Jean Jaurès (IRIT), Toulouse, France
Nadine.Baptiste@irit.fr

Abstract. While computing skills grow in importance in today's technology-centered society, the learning of these skills still isn't accessible easily for young visually-impaired students: in French schools for example, online platforms (like Scratch) are more and more used by teachers, but unfortunately these platforms rely heavily on visual elements. As an inclusive approach would suggest, modifications and adaptations of such platforms would favour collaboration between sighted and visually-impaired users: tangible stimuli are then favoured to compensate for visual elements, while visually-impaired and sighted communities are prompted to use the same tools. Even if tangible solutions are suggested through scientific studies in the accessibility field, a young visually-impaired student still can't use these solutions autonomously: many of these prototypes still require the intervention of a sighted third party. In this article, we describe our solution TaBGO (Tangible Blocks Go Online) which consists of enhanced tangible Scratch's blocks and an associated optical recognition software. We then present a planned user study to establish if this solution is usable and easily handled by young visually impaired and sighted students, considering users' feedback about usability, satisfaction and cognitive load.

CCS Concepts
• Human-centered computing • Accessibility • Accessibility systems and tools

Keywords: Algorithmic · Accessibility · Block programming · Tangible objects · Optical recognition

1 Introduction

With the technological improvements of the last decades, the field of computer science and associated computing skills grow to have an increased importance in today's world. Thus in France, the learning of these computing skills begins in primary school since

M. Antona and C. Stephanidis (Eds.): HCII 2021, LNCS 12769, pp. 12–24, 2021.
https://doi.org/10.1007/978-3-030-78095-1_2

the 2016 reform[1], and in secondary school using block-based Visual Programming Languages (VPL) like Scratch[2] or Blockly[3]. Thanks to these educational strategies, students can have a first introduction to computer science.

However, for visual impaired persons, these computing paradigms aren't accessible since they rely on visual elements [1]. Thus, this graphical user interface (GUI) prevents visually impaired students from this easy approach to computing skills, and subsequently doesn't encourage them to pursue a developer's career, nor towards socio-professional integration in today's technology-centered world [2]. The issue is clear: in secondary school, is it possible to give visually impaired student access to block-based VPL, and thus access to a simple acquisition method of these computing skills? This article presents the TabGO (Tangible Blocks Go Online) project, a tangible interface which allows Scratch programming in classrooms for people with visual impairments, using tangible blocks and an associated optical recognition software. First, previous existing technologies are presented and their influences on the TabGo project are highlighted. Then, the TabGo prototype is introduced, with an experiment to evaluate its usability. This article ends with future perspectives on the experimentation's results.

2 Related Works

Programming languages can be accessible for visually impaired users (especially when they are based on textual instructions), but such accessibility necessitates adaptations: some modifications concerning the IDE used or some other additional equipment like a screen reader (JAWS[4], for example). These accessibility-centered processes are *Assistive Technologies* [3, 4], and many of these solutions are based on audio and/or braille feedback to allow an intuitive navigation for visually-impaired users [5]. Moreover, the field of vocal synthesis (*Text-to-Speech*) has known many progresses during the last decades, representing an aural feedback: for example, *Javaspeak* [6, 7], *Emacspeak* [8] and *CAITLIN* [9] are all IDEs that include a vocal synthesis-based text editor; while the first two center around the accessibility of the Java language, CAITLIN targets the use of a design software. In additional equipment and in accessible IDEs, vocal synthesis is used as an efficient solution to make computer-based fields accessible for visually-impaired users.

In addition to vocal synthesis, it's important to highlight the focus on some languages, such as Java (in the previous examples) and Python. While these languages may present accessibility issues for visually impaired persons [10], they are nonetheless more and more used among scientific, engineering and computer-centered communities: as a consequence, blind programmers may learn to use it anyway, as some languages have "features that more than make up for any inconvenience that indentation may cause" [10]. While additional equipment and IDEs are useful tools for accessibility, the use of

[1] "Bulletin Officiel spécial n°11 du 26 novembre 2015", http://cache.media.education.gouv.fr/file/MEN_SPE_11/35/1/BO_SPE_11_26-11-2015_504351.pdf.

[2] https://scratch.mit.edu.

[3] https://blockly.games/?lang=fr.

[4] https://www.freedomscientific.com/products/software/jaws.

new languages, easier to learn, is also a point worth highlighting. Moreover, some languages are designed while consciously considering accessibility for visually impaired users, such as Quorum[5]: it is "evidence-based" (as they rely on scientific studies[6] [11]), and its syntax is specially designed to be easily read by a vocal synthesis engine. Quorum is therefore an accessibility solution which exemplifies the use of both vocal synthesis and modifications of IDE/existing programming languages.

If these studies contribute to making programming languages as accessible as possible, they aren't necessarily adapted for an integration in a classroom environment. However, this classroom integration can favour the possibility of collaboration between visually-impaired pupils and their visually-paired peers: this collaboration has a crucial role in the integration of visually impaired people [12], as seen in previous inclusive experiments in the classroom environment [13]. Moreover, it's important to highlight that for visually-impaired teenagers, the use of adapted interfaces give them an additional opportunity to develop their logical, cognitive and motor skills, and thus these interfaces may have a positive influence on their personal development, on both physical and psychological levels [14, 15].

For children, computing skills' learning is generally based on simpler IDEs, like Blockly, Scratch *StorytellingAlice* [16] and *LookingGlass*[7] [17]: it's worth mentioning these IDEs are often used by secondary school students. However, all these environments also rely heavily on visual elements, and thus need modifications to be accessible. For an easy and intuitive approach, one strategy is then to replace the visual stimuli by tactile stimuli: a tangible interface (TUI) is then preferred to a graphical one (GUI) [18].

Many studies have been conducted on usable TUIs in a scholar environment. In primary school, adapted TUIs are often derived from electronic audio-based toys: for example, Microsoft's CodeJumper/Torino project [19] rely on linkable blocks; Bee-Bot is a bee-shaped controller, linked to Lady Beetle and World of Sounds, two simplified music-centered programming softwares [20]. The modifications of toys into TUIs represent a simple but efficient way to introduce some computing basics to primary schools pupils, especially block-based toys: it's also possible to mention some block-based prototypes among scientific communities, such as the T-Maze [21], based on maze-construct maze-escaping tasks using tangible blocks; the P-Cube [22], based on RFID blocks and cards; and a grid-like LEGO-based TUI [23] that uses aural communication. To appeal to older students, other (more android-like) toys have been derived to act as feedback for simple algorithms, otherwise only accessible by sighted students since they rely on visual modifications of an avatar. For example, the Roamer is a turtle-like robotic toy, which would move entirely as the avatar would, translating visual stimuli into movements [24]. While a toy-based approach remains an effective one for accessibility, toys are not the only raw material for accessible TUIs helping visually-impaired pupils.

In secondary school, other tangible solutions have also been created: *Blocks4All* [25] presents a sensibly augmented IDE on Android tablet (which emulates different textures), while *AccessibleBlockly* [5] give access to every Blockly modules (*ArduBlockly*[8],

[5] https://quorumlanguage.com/.

[6] As advertised on https://quorumlanguage.com/evidence.html.

[7] http://www.alice.org/.

[8] github.com/carlosperate/ardublockly.

OzoBlockly[9], *BlockyTalky* [26]) through audio feedback. It's worth mentioning that this Blockly-based inclusive approach uses audio and tactile feedback, as did the solutions previously mentioned. Scratch has also been adapted for visually impaired users: the *Accessi-DV* Scratch briefcase [27] contains tangible programming blocks based on the Scratch blocks; the *CodeBox64* [28] is an Arduino-based Scratch controller. These two examples are crucial influences for the actual project TabGO [29]. All of these technologies consider the use of TUIs and hardware as possible accessible solutions that allow a first approach towards programming skills in secondary school.

While the prototypes mentioned above offer excellent guidelines, they still present issues for an implementation in the classroom environment. The *Codebox64*'s concept is issued from an Arduino-based approach and requires constructions and realizations from sighted competent teachers, for each student. These chain productions necessitate time and different components for each type of feedback required in each activity planned with it. Plus, this Arduino-based approach doesn't highlight Scratch's block-based logic and its benefits for young users [30]. However while the Accessi-DV briefcase does underline Scratch's block-based logic, the prototype doesn't allow the user to communicate directly with the Scratch online platform, and thus requires a sighted person to copy the algorithm completely on the Scratch online platform. While "activity-based unplugged coding and robotic coding training, integrated with the preschool education curriculum, enhanced the basic coding and robotic coding skills" [31], unplugged coding still have limitations: for example, the algorithm must be transposed to the Scratch platform for the user to experience feedback These solutions therefore require the help of a third party: to avoid involvement of this third party and to maximize autonomy, we have designed the TabGo solution.

3 The TabGO Project

In addition to related works, the TabGO project is also based on many others focused on the conception of block-based TUIs [32, 33], their use and efficiency [34, 35] or their reception by users [36–38]. Those works also include prior block-based TUIs prototypes [27, 39], improved to communicate with online Scratch. The actual TabGo prototype consists of two parts: a briefcase with tangible blocks, and an associated visual recognition device and software.

The blocks have been modeled corresponding to the virtual ones, both in aspect and in functions. They are wood-made or plastic-made (using a laser cutter through plastic sheets of 3 mm width), and are implemented with thin magnets (2 mm diameter × 3 mm width) that allow for connection between blocks, just like the Scratch system. These functions have been preserved since this system minimizes syntax errors, just like the online platform. While the blocks can be connected depending on their shape (like puzzle pieces) using incorporated magnets, the TabGo blocks also are enhanced with ropes in case of complex functions like conditional branches and Boolean loops, which are essential notions in the secondary school educational program.

In the Scratch platform, different colors indicate different function domains: for example, orange is associated with variables, while yellow is associated with events.

[9] ozoblockly.com.

For this useful characteristic, colors have been kept the same while designing the TabGo prototype, mostly for visually paired users. For visually impaired users, they have been carved with different patterns (such as straight lines, multiple dots, etc.) to mimic these colored functionalities: the logic applied here aims to compensate visual information by tangible stimuli.

Moreover, it's worth mentioning that even the briefcase has an indicative function here: the briefcase contains five compartments, one for each color/set of texture (see Fig. 1). The first blocks in the chain are in the top compartments: the event blocks in the top left compartment (such as the "when button pressed" event) and the micro:bit blocks in top right compartment. The center compartment holds the variable-based blocks (such as the "add 1 to my variable" function), as these notions are at the center, the heart of computing skills. The feedback blocks are placed at the bottom: the sound blocks are in the bottom left compartment (such as the "play alarm sound" function); while the Text-To-Speech blocks are in the bottom left compartment. This organization also places Scratch's basic functions at left and advanced extensions (Text-to-Speech and micro:bit) at right: Scratch's possibilities are organized left to right from simple to more complex. Thanks to this organization, a visually-impaired user can differentiate the blocks more easily, when trying to recreate an algorithm.

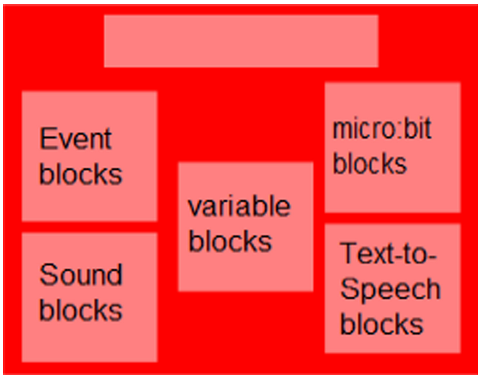


Fig. 1. Drawing representing the organisation of the TabGo briefcase.

While compensating for visual communication, it's important to note that the aural communication is also used: The TabGO project uses Scratch extensions to create non-visual feedback. Each of these extensions represents specific blocks that allow the integration of additional equipment or functions. Focusing on non-visual stimuli, we chose to adapt the "Text-to-Speech", "Music/Sound" (MIDI and musical language) and "Micro:bit"[10] extensions, which offer audio or tactile feedback. The TextToSpeech blocks use vocal synthesis (with functions like "pronounce hello world"); the music/sound blocks use and organize sample sounds in time (counted then as a fraction of time, 0.25 s for example) and the Micro:bit blocks allows the use of micro:bit cards, little

[10] https://microbit.org/.

electronic devices that integrates audio and vibrotactile feedback to Scratch algorithms (with functions like “when micro:bit shaked”).

The blocks have been enlarged compared to the original interface (18 cm × 10 cm) to be enhanced by braille text and *cubarithms* to be identified by visually impaired users: the cubarithms here are little plastic cubes (1 cm × 1 cm × 1 cm) based on the Aubrey wheel[11]; they can be parameterized by users to hold a single braille cell. These cubarithms can then display numbers and variables’ names. These enhancements aim to maximize the autonomy of visually-impaired users, since they can read and identify blocks, but also correctly set them to build customized algorithms.

As the first part of the TabGo prototype, the blocks presented here are derived from the Scratch’s interface to retain the most useful features of this pupil-friendly approach to computer science, such as magnetic linking between blocks and color code. But they’re also enhanced with many features favouring the blocks’ correct identification and setup: these features are multiple, from the briefcase’s organization to additional carved textures upon blocks.

While the precedent features were highlighted because they favour recognition by the user, the second part of the TabGo prototype centers around optical recognition of the algorithms (recreated with blocks) by the system. The solution presented here, in addition to blocks, centers around the use of a webcam to automatically recognize the *cubarithms* and the TopCodes[12] symbols placed upon the blocks.

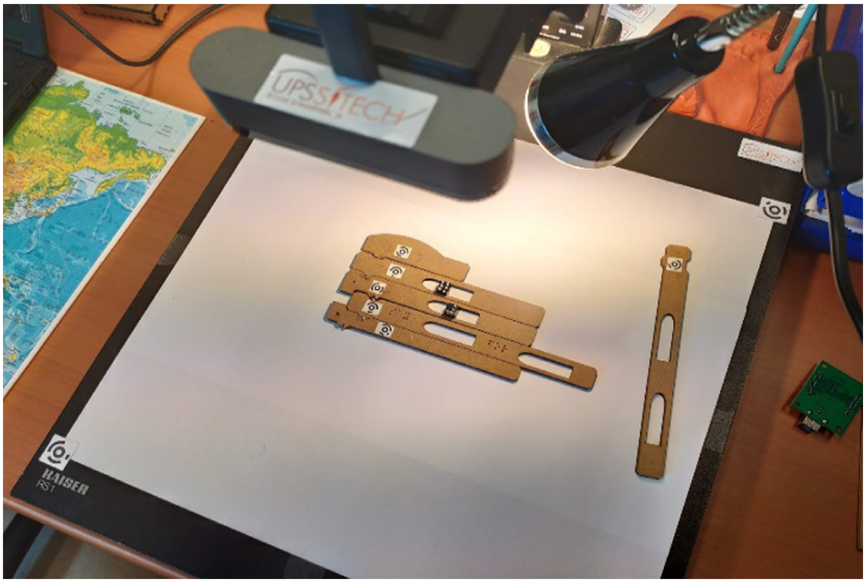


Fig. 2. Full platform overview.

[11] https://www.utopiamechanicus.com/article/braille-display-2017/.
[12] http://users.eecs.northwestern.edu/~mhorn/topcodes/.

By adding TopCodes symbols and associated libraries, each block's type is optically recognized by the software part using a web-cam; while the values inside blocks are recognized by *cubarithms'* analysis: the TopCodes' analysis identify the types of blocks used in the algorithm; while the *cubarithms'* analysis identify the values held into such blocks. A JSON file and related resources are then generated as a SB3 file (common extension for Scratch language) to be read by Scratch (as shown in Fig. 3).

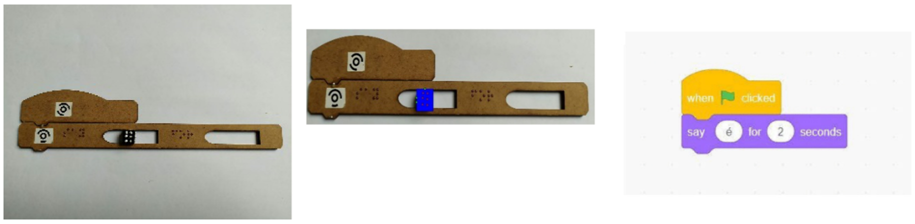


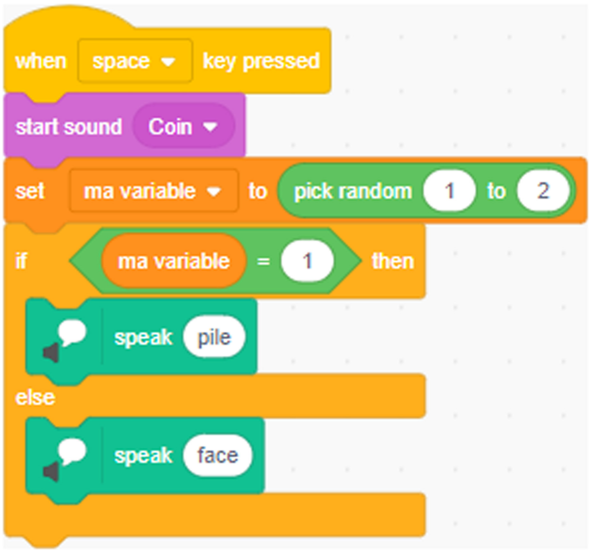
Fig. 3. From a simple algorithm to the algorithm in Scratch after the visual recognition.

The main objective is to give autonomy to visually-impaired students when using Scratch with the TabGO TUI (as shown in Fig. 4): thanks to this TUI, a visually-impaired user can use Scratch autonomously, from recreation of algorithms to optical recognition by the software, to a SB3 file exploitable by Scratch.

With the use of these enhanced blocks and this optical recognition software (as previously mentioned), it is then possible for users to design simple algorithms that help introducing important computing notions. So, these algorithms represent an easy way to enhance computing skills for students, whether they're visually paired or impaired. Finally, these algorithms also represent the first experimental phase for testing this solution's viability.

4 User Study

The tests planned to evaluate the efficiency of the TabGO solution are based on a multiple case study, centered around a sequence of four simple algorithms the subject has to build, four stages arranged by increasing difficulty. These algorithms were limited to a few blocks and were constructed accordingly to the French secondary school's program, thus remaining accessible for young students. Each stage of the experiment (except the first one) can be realized by one subject or by two subjects working together, mirroring the will previously mentioned to focus on (respectively) autonomy and inclusive collaboration. For instance, one of the first algorithms is based on a simple coin tossing machine (a six-block algorithm, see Fig. 4) inspired by an 8th grade assessment, introducing pupils to conditional branches and probabilities.



```
when space key pressed
start sound Coin
set ma variable to pick random 1 to 2
if ma variable = 1 then
    speak pile
else
    speak face
```

Fig. 4. Scratch interface – Coin Toss algorithm.

While the usability of this technology (as a first approach to computer science) is the main subject of these test series, parameters of age, technology affinity and braille knowledge are considered while evaluating efficiency of the TabGO prototype. Thus, four groups of sighted or visually impaired users of different ages will be confronted (P1 to P4).

- P1: Visually Impaired Teenagers
- P2: Sighted Teenagers
- P3: Visually Impaired Adults
- P4: Sighted Adults

The four groups are asked to construct specific algorithms following steps and instructions, under an informed instructor's watch. For each step, the instructor has to answer three questions, (1) how well does the subject identify blocks, (2) how well does the subject connect blocks and (3) how well does the subject experience feedback. Instructions are given to the test subject at the start of the task, and the instructor is told not to intervene, unless told to during crucial steps or if the subject encounters many difficulties. However, the instructor is also told to write any demand of intervention originated by the test subject: after he/she has been asked for help four times, the instructor is told to skip the actual task and begin the next one.

The parameters observed essentially focus on completion (or non-completion) of a given task, and the time necessary to complete each step/stage. Each parameter is appreciated by the observer with the use of a 5-point Likert scale (see Table 1). The experiment's duration is thus notated in relative (Likert scale) values, for each step and each stage. The experiment's completion includes progress between steps and between stages: these progresses indicate how well does the subject adapt to the prototype while apprehending new computing notions. Other parameters observed include collaboration if the stage is realized by two subjects working together and autonomy if the stage is realized by a single subject. These two parameters are appreciated according to the number of times the subject(s) has/have asked for help or verification, and also according to the number of tries for each step. The collaboration parameter also depends on the communication between users, higher if the communication is productive and/or enjoyable for each user.

Table 1. Experimental protocol – Observation grid

Parameter observed	1	2	3	4	5
Step duration (Likert scale)					
Block handling					
Progress between steps					
Progress between stages					
Collaboration					
Autonomy					
Total (/35)					
Stage duration (35 min at most):					
Number of times the subject asked for help:					
Number of times the subject asked for verification:					

At the end of each stage, the subject will also be asked to fill (indirectly by interview for visually impaired users) a satisfaction survey, a task load assessment (NASA-TLX) and a usability survey (SUS, or System Usability Scale). With these methods of data collection, many aspects and related questions will be answered to establish the actual efficiency of the TaBGo prototype.

Q1 - Were the blocks easily used and recognized by subjects? (Usability aspect, data collected by direct observations and by System Usability Scale).
Q2 - Was the task difficult for subjects? (Cognitive aspects, data collected by direct observations, NASA-TLX and by Satisfaction survey).
Q3 - Was the approach offered by using the TabGO prototype helpful for the subject to develop computing skills? (Progression aspect, data collected by direct observations and by NASA-TLX).
Q4 - Was the collaboration with other users pleasant and useful to subjects? (Collaborative aspect, data collected by direct observations and by satisfaction survey).
Q5 - Was the subject able to use the prototype autonomously? (Autonomy aspect, data collected by direct observations and by satisfaction survey).

All of the aspects of this sample experimental phase are summed up in Table 2 below.

Table 2. Experimental protocol – Aspects investigated and methods used to investigate them

Aspects investigated	Method of data collection
Usability	Direct observations, System Usability Scale
Cognitive load	Direct observations, NASA-TLX, Satisfaction survey
Progression	Direct observations, NASA-TLX
Collaboration	Direct observations, Satisfaction survey
Autonomy	Direct observations, Satisfaction survey

With these data collection methods combined, the experimental protocol presented here aims to be as complete as possible, considering both quantitative and finite aspects of task-resolving, as well as qualitative and human aspects in task-involvement. The subsequent stages imply different feedback and/or more complex algorithms, composed of multiple pairs of blocks connected together, or realized in collaboration with another user. Though knowledge progression and computing skills' acquisition (Q3) represent an important aspect of these tests, it is a secondary one as the main goal here is before and foremost to evaluate usability of the prototype in different conditions, with increasing difficulty.

The data collected are then analyzed to establish if there are differences in performance between each group of subjects, differences that can thus be linked to each subject's individual skills and experience. For each data collection method, the results' mean of each item is then calculated to establish if, overall, the TabGo TUI-based first approach towards programming skills' learning is actually intuitive enough to be used by secondary school students, whether they're visually impaired or sighted.

According to similar studies [28, 29], the results expected revolves around a clear apprehension and identification of the blocks used. The tasks composing the protocol are centered around the secondary school program, so the results shouldn't vary much from group to group, especially if the many enhancements reveal to be useful, as seen in similar studies [35, 39] and as announced by the first results of the pre-test phase. Though satisfaction may not be optimal for adults (as the tasks may not be particularly mentally challenging for them), we expect these tasks to be simple and enjoyable enough to encourage involvement and collaboration between all users, and especially between sighted and visually-impaired users. If the results are promising, the subsequent experimental protocols focus on the learning and acquisition of computing skills with the TabGo prototype. As previously mentioned, these results will help confirm hypotheses on the prototype's usability while also considering important factors playing a role in a learning (cognitive load, progression) and social (collaboration, autonomy) context: such data will help to establish if the prototype will be efficient and adapted in a classroom environment.

5 Conclusion

While pursuing the development of the TabGO prototype and while relating its progresses, this article has first presented possible guidelines for the conception of *Assistive*

Technologies for visually-impaired users to learn computing skills, in an educational context. Related works and major influences in the accessibility field were then presented, with an emphasis on classroom-integrated TUI solutions. Some characteristics have been highlighted: the use of block-based VPL (such as Scratch) may be adapted by the implementation of an adapted IDE centered on non-visual feedback (audio and tactile cues), and by the use of tangible blocks. By modifying and enhancing a learning tool, this inclusive approach has proven to improve collaboration between visually-impaired users and their sighted peers.

The TabGO project followed these guidelines and proposed a prototype composed of tangible enhanced Scratch blocks (with braille symbols, cubarithms, TopCodes and carved patterns) organized in a briefcase, and an associated optical recognition software (using a webcam), translating tangible block-based algorithms into exploitable files for the Scratch platform. With this prototype, the project is aiming to help visually-impaired users learn computing skills autonomously while still favoring inclusion and collaboration through a tangible activity in the classroom environment. With promising preliminary results, the experimental phase will soon be launched to evaluate the usability of this prototype.

Acknowledgments. This work is supported by the French UNADEV association (Grant 2019.49).

References

1. Carver, J. (ed.): Quality, nontechnical skills, blind programmers, and deep learning. IEEE Softw. **36**(2), 127–136 (2019). https://doi.org/10.1109/MS.2018.2883874
2. Kearney-Volpe Cl: Web Development Training for Students That Are Blind W4A 2019, 13–15 May 2019, San Francisco, California, USA (2019)
3. Senjam, S.: Assistive technology for students with visual disability: Classification matters. Kerala J. Ophthalmol. **31**(2), 86 (2019). https://doi.org/10.4103/kjo.kjo_36_19
4. Truillet, Ph.: L'informatique, pour un monde plus accessible. Bulletin de la Société informatique de France – numéro 15, avril 2020, France (2020)
5. Ludi, S., Merchant, W., Simpson, J.: Exploration of the use of auditory cues in code comprehension and navigation for individuals with visual impairments in a visual programming environment. In: ASSETS 2016, 23–26 October 2016, Reno Nevada, USA (2016)
6. Cheong, C., Burge, A. (eds.): Coding without sight: Teaching object-oriented java programming to a blind student. In: 8th Annual Hawaii International Conference on Education, Honolulu, Hawaii, 7–10 January 2010, pp. 1–12 (2010)
7. Francioni, J.M., Matzek, S.D., Smith, A.C.: A java programming tool for students, with visual disabilities. In: ASSETS 2000, November 2000, Arlington, Virginie, USA (2000)
8. Raman, T.V., Tauber, M.J. (eds.): Emacspeak—a speech interface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1996), pp. 66–71. ACM, New York, USA (199)
9. Vickers, P., Alty, J.L.: Musical program auralisation: a structured approach to motif design. Interact. Comput. **14**(5), 457–485 (2002). https://doi.org/10.1016/S0953-5438(02)00004-8
10. Teaching Modern Object-Oriented Programming to the Blind: An Instructor and Student Experience – Dr. Charles B. Owen, Michigan State University; Sarah Coburn, Michigan State University, Ms. Jordyn Castor

11. Kaijanaho, A.-J.: Evidence-Based Programming Language Design: A Philosophical and Methodological Exploration Information Technology Faculty. University of Jyväskylä, Ph.D. Dissertation (2015)
12. Archambault, D.: Interaction et usages des modalités non visuelles, accessibilité des contenus complexes, (Thèse) Université Pierre et Marie Curie - Paris VI, Paris (2010)
13. Metatla, O., et al.: Toward classroom experiences inclusive of students with disabilities. Interactions **26**(1), 40–45 (2018). https://doi.org/10.1145/3289485
14. Flammant, J.: De l'œil au regard. SIDVEM, Paris (2016)
15. Hatwell, Y.: Le développement perceptivo-moteur de l'enfant aveugle. Enfance **55**(1), 88 (2003). https://doi.org/10.3917/enf.551.0088
16. Kelleher, C., Kiesler, S., Pausch, R.: SAMMS girls to learn computer programming. In: CHI 2007 Proceedings, April–May 2007, California, USA San Jose (2007)
17. Chou, M.: Designing a community to support long-term interest in programming for middle school children, IDC 2012, 12–15 June 2012, Bremen, Germany (2012)
18. Brock, A.: Tangible interaction for visually impaired people: why and how. In: World Haptics Conference - Workshop on Haptic Interfaces for Accessibility, June 2017, Fuerstenfeldbruck, Allemagne. pp. 3. ffhal-01523745ff (2017)
19. Cecily, M., et al.: Physical programming for blind and low vision children at scale. Hum. Comput. Interact. (2019). https://doi.org/10.1080/07370024.2019.1621175
20. Jaskova, L., Kaliakova, M.: Programming Microworlds for Visually Impaired Pupils, Conférence de Constructionism 2014, Vienne, Autriche (2014)
21. Wang, D., Zhang, C., Wang, H.: T-Maze: a tangible programming tool for children, pp. 127–135 (2011). https://doi.org/10.1145/1999030.1999045.
22. Motoyoshi, T., Tetsumura, N., Masuta, H., Koyanagi, K., Oshima, T., Kawakami, H.: Tangible gimmick for programming education using RFID systems. IFAC-PapersOnLine. **49**, 514–518 (2016). https://doi.org/10.1016/j.ifacol.2016.10.608
23. Utreras, E., Pontelli, E.: Design of a tangible programming tool for students with visual impairments and low vision. In: Antona, M., Stephanidis, C. (eds.) HCII 2020. LNCS, vol. 12189, pp. 304–314. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49108-6_22
24. Virey, M., Renaud, P.: Le Roamer: un robot déjà ancien au service d'apprentissages bien actuels: Utilisation dans une Classe d'inclusion scolaire (Clis 1) de l'Yonne. La nouvelle revue de l'adaptation et de la scolarisation **52**(4), 231 (2010). https://doi.org/10.3917/nras.052.0231
25. Ladner, R.E., Milne, L.R.: Blocks4All: overcoming accessibility barriers to blocks programming for children with visual impairments. In: CHI 2018, Avril 2018, Montréal Canada (2018)
26. Deitrick, E., Sanford, J., Benjamin, R.: BlockyTalky: a low-cost, extensible, open source, programmable, networked toolkit for tangible creation. In: IDC 2014, 17–20 June 2014, Aarhus, Danemark (2014)
27. Boissel, S.: Mallette Accessi DV scratch « Scratch débranché en braille et gros caractères ». La nouvelle revue de l'adaptation et de la scolarisation **77**(1), 183 (2017). https://doi.org/10.3917/nras.077.0183
28. Marco, J.-B., Baptiste-Jessel, N., Truillet, P.: TabGO: Programmation par blocs tangibles. In: 30e Conférence francophone sur l'Interaction Homme-Machine (IHM 2018), 23 October 2018 - 26 October 2018, Brest, France (2018)
29. Amber, W., Zirui, W.: Evaluating a tactile approach to programming scratch. In: ACMSE 2019 Avril 2019 Kennesaw USA (2009)
30. Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E.: The scratch programming language and environment. ACM Trans. Comput. Educ. (TOCE). **10**, 16 (2010). https://doi.org/10.1145/1868358.1868363

31. Metin, S.: Activity-based unplugged coding during the preschool period.Int. J. Technol. Des. Educ. 1–17 (2020) https://doi.org/10.1007/s10798-020-09616-8
32. Sanchez, J., Aguayo, F.: Blind learners programming through audio. In: CHI 2005, pp. 1769–1772, 2–7 April 2005, Portland (2005). https://doi.org/10.1145/1056808.1057018
33. Shreekanth, T., Udayashankara, V.: A review on software algorithms for optical recognition of embossed braille characters. Int. J. Comput. Appl. **81**(3), 25–35 (2013)
34. Horn, M.S., Robert, J.K.: Tangible programming in the classroom: a practical approach. In CHI 2006 Extended Abstracts on Human Factors in Computing Systems (CHI EA 2006), pp. 869–874. ACM, New York, NY, USA. https://doi.org/10.1145/1125451.1125621
35. Observatoire des Ressources Numériques adaptées, Scratch 3D Magnet, janvier 2018, 13p. https://www.apmep.fr/IMG/pdf/Orna_Scratch3DMagnet.pdf
36. Capovilla, D., Krugel, J., Hubwieser, P.: Teaching algorithmic thinking using haptic models for visually impaired students. In: LaTiCE 2013, pp. 167–171, 21–24 March 2013. https://doi.org/10.1109/LaTiCE.2013.14
37. UN General Assembly: Convention on the Rights of Persons with Disabilities: resolution / adopted by the General Assembly, 24 January 2007, A/RES/61/106. http://www.refworld.org/docid/45f973632.html. Accessed 17 July 2018
38. Zuckerman, O., Grotzer, T., Leahy, K.: Flow blocks as a conceptual bridge between understanding the structure and behavior of a complex causal system. In: Proceedings of the 7th international conference on Learning sciences (ICLS 2006). International Society of the Learning Sciences, pp. 880–886 (2006). https://dl.acm.org/citation.cfm?id=1150162
39. Aymard, P.: Algorithmique Scratch et cécité. Exemple d'un support débranché et adapté (2018). http://revue.sesamath.net/spip.php?article1082