Is the Scratch Programming Environment Ideal for all? Enhancements to the Scratch IDE to Make it Easier to Use and More Useful for Students and Teachers

Stefano Federici[a], Elisabetta Gola[b] and Elisabetta Sergi[c]
University of Cagliari, via Is Mirrionis, 1, Cagliari, Italy

Keywords: Scratch, Integrated Development Environment, Block-Based Programming, Scratchaddons, Snap, Dyslexia.

Abstract: Even the most successful programming environment for beginners, Scratch, when compared to other block-based IDEs, is not always the ideal tool for everyone. Scratch Team's development policy -creating a single environment for all users where no customization is allowed- makes Scratch a great environment for most users to start with computer programming, but not the ideal environment for those users that -thanks to Scratch's outstanding features- want to use it to create complex projects, or for teachers that use it for their lessons. In this paper we will highlight the weaknesses of Scratch for expert developers and for teachers. We will then look at other popular block-languages and tools to find which weaknesses they can help to solve. Finally, we will propose some new enhancements, that are still not available in Scratch and in the tools we analysed.

1 INTRODUCTION

Scratch 1, whose public release dates back to 2007, has evolved very slowly since then, especially when we look at it as a programming language. Comparing Scratch 1.3 -the first public release that could handle string and list variables other than numbers- to Scratch the last stable release 3.0, very little has changed in the Scratch language or IDE (Integrated Development Environment). Every advancement is carefully planned by the Scratch Team, the main goal remaining to keep Scratch always completely unambiguous and easy to use even by "younger users […] focus[ing] on self-directed learning" (Maloney et al., 2010). These goals have generated an absolute absence of possibility of customization: everything must be always visible, and, more importantly, all users must share the same experience (Figure 1). All users will see the *block palette* (1), the *coding* area (2), the *Stage* (3), and the *sprite* area (4).

So, even users with specific needs, like advanced users, or teachers, or students with Learning

Figure 1: The "Breakout" project in the standard Scratch IDE. The main areas are the block palette (1), the coding area (2), the Stage (3), and the sprite area (4).

Disorders (LDs) can't do much except enabling a larger or a smaller Stage (area 3 at the top right of Figure 1) in the IDE, so to get a somewhat larger coding area (area 1 in Figure 1). Only a few of the scripts composed by colourful blocks are visible in the central area. All users must stick to the Scratch IDE as it is.

[a] https://orcid.org/0000-0001-8506-1360
[b] https://orcid.org/0000-0003-0966-9520
[c] https://orcid.org/0009-0009-7795-5861
[1] http://scratch.mit.edu

Federici, S., Gola, E. and Sergi, E.
Is the Scratch Programming Environment Ideal for all? Enhancements to the Scratch IDE to Make it Easier to Use and More Useful for Students and Teachers.
DOI: 10.5220/0011838200003470
In *Proceedings of the 15th International Conference on Computer Supported Education (CSEDU 2023) - Volume 1*, pages 171-181
ISBN: 978-989-758-641-5; ISSN: 2184-5026

Even if there are alternatives to using Scratch online (Federici et al., 2022), every month the Scratch website is visited by more than 30 million visitors and about 2 million projects are created online[2].

Being the usage of the Scratch online editor so widespread, in this paper we are going to first explore the *addon* tools that are already available and that allow to modify the Scratch code editor "on the fly" directly on the website by giving more power to all users, whether they are students or teachers (Figure 2).

Figure 2: The "breakout" project in the Scratch IDE enhanced by the Scratch *addons*, where all scripts of the selected object are visible in the coding area.

As not everything can be done by using these addons, we will also explore powerful instruments available as alternatives to Scratch, and we will also propose new instruments that should be available for particular categories of students and teachers.

This study is the outcome of having used Scratch as an introductory tool to computer programming in three Computer Science courses of the degree in Communication Studies of the University of Cagliari from 2008 to 2022. Most of the proposed enhancements come also from using Scratch as an introductory tool to computer programming for k-3 to k-5 students and their teachers from 2014 to 2020 (Federici et al., 2015, 2018, 2019).

In the following sections we are going to revise the evolution of Scratch, from Scratch 1 to Scratch 2 and from Scratch 2 to Scratch 3, highlighting why the new features and modifications have not always been ideal for students and teachers. Then we will describe 2 interesting tools (an enhanced clone of Scratch and a set of addons that can be added to the Scratch online tool) and will analyse their features with respect to the way they impact on the users. Finally, we propose two addons that could potentially be really useful for users with learning disabilities.

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

[3] These blocks are now available in the Scratch 3 Music extension

2 SCRATCH AS A VERY STABLE AND FULLY LOCKED PROGRAMMING IDE

Starting from 2007, very little has changed in the Scratch editor/language. Furthermore, not all changes added something. Some of them made the IDE simpler but poorer in some respect. Every change has always been imposed to all users that wanted to use the new features available in the new version, even those users that, for very good reasons, needed some of the features that were available only in the older version(s).

2.1 The Evolution of the Scratch IDE

The main changes that have impacted on previous users are the following:

- Scratch 2.0 (2013):
  - the Java Scratch editor is reimplemented using Adobe Flash Player, so that users can now create Scratch projects online (2013) and can benefit of a faster project execution; as a downside, the quality of the sound of the MIDI instruments produced by the blocks in the Sound category[3] was critically reduced, due to Flash's smaller sound library[4];
  - users can now create their own command blocks, that is their *procedures*;
  - the Stage is moved to the left-hand-side of the IDE;
  - the "FOREVER IF" block, whose purpose was to trigger an action *every time that* something happened, is removed. The reason is that some novice users felt that the way it worked was ambiguous[5]. Since then, users must use two blocks to get the same result, namely "FOREVER" and "IF". Using two blocks instead of one is felt as unnatural by users that are learning Scratch for the first time.
  - the project cannot be run anymore step-by-step
- Scratch 3.0 (2019):
  - Scratch editor is reimplemented in JavaScript, due to the Flash Player being discontinued by Adobe for security reasons;

[4] https://en.scratch-wiki.info/wiki/Sound_Blocks

[5] https://scratch.mit.edu/discuss/topic/343602/

- the Stage is moved back to the right of the IDE, as this arrangement is felt more intuitive and understandable[6];
- the Scratch blocks are made thicker, so to allow both new users and touch device users to drag them more easily[7];
- the Pen category and Sound blocks related to notes and musical instruments are moved to the *extensions*.

All these changes cannot be reverted by the user, they must use the tool "as is" or totally reject it.

2.2 The Drawbacks of the Evolution Strategy of the Scratch IDE

The evolution strategy of the Scratch IDE followed by the Scratch Team has drawbacks both as for the changes they decide to apply and for the changes they decide to reject[8] or that they don't feel as important for their goals.

2.2.1 Changes Applied Without Appeal

As the IDE cannot be customised, every change had some drawback for a part of the Scratch users or teachers. When Scratch 2 appeared:

- if a music teacher or a user had created high quality music projects in Scratch 1, these projects became lower quality MIDI music when loaded in Scratch 2;
- If a user was used to the FOREVER IF block, they will have to discover by themselves that this block must be replaced by an IF block embedded in a FOREVER block, or they will have to ask for help in the Scratch Forum and hope that they will quickly find an answer
- If a class was used to the Scratch 1 IDE style, their teachers will have to keep using the old Scratch 1 version if they didn't want to risk confusing their students with a Stage placed at the opposite side.
- Many very useful features, especially when teaching to very young students, are gone forever (they are missing in Scratch 3 too). Among them:
  - the possibility to highlight the blocks currently executed and to run the project more slowly in order to better understand the program flow;
  - the possibility to 'shoot pictures' of the -partial or total- Stage content to create either new sprites or new sprite costumes on the fly (useful for example to create "sensible zones" on a sprite, or to avoid using the paint editor to cut off parts of the imported images);
  - sprites cannot be visually rotated/resized directly on the Stage;
  - buttons above the Stage (to cut/copy/duplicate sprites, blocks, costumes, and sounds), and contextual menus for items on the Stage are gone forever.

If a user or a teacher had started using Scratch 2 with their students, when Scratch 3 appeared:

- if they were used to create fairly long scripts when necessary, now the thicker blocks force them to scroll a lot to show all the blocks in the coding area to their students. The new possibility to zoom out the coding area is not a solution, as the text of the blocks gets quickly unreadable;
- if an art or a music teacher, or a user used a lot the Pen and Music blocks to create drawings on the Stage or to create music, they will have to ask their students to add the Pen and the Music extension to the Scratch IDE every time they start a new project. Curiously enough, the faster JavaScript reimplementation of Scratch 3 Pen blocks caused an upsurge of 3D projects created by using these blocks;
- the x/y coordinates of the mouse pointer are no longer shown under the Stage.

2.2.2 Requests for Changes that are Not Taken into Consideration

As the evolution of the IDE is very slow, at least compared to the number of proposals for changes requested by Scratch users [9], many reasonable changes that teachers would really need, basing on their -often long- experience, are fully rejected.

Just to give an example, one of the authors of this paper officially proposed the Scratch Team[10] in 2014 to add a mechanism to temporarily hide the palette of the blocks and/or the Stage, in order to get a larger coding area where two scripts could be put side by

[6] https://scratch.mit.edu/discuss/topic/300543/
[7] https://scratch.mit.edu/faq
[8] https://scratch.mit.edu/discuss/topic/343602/
[9] https://scratch.mit.edu/discuss/1/

[10] "What's Next for Scratch?" Panel of the Scratch Conference 2014; program available at the page https://cdn.scratch.mit.edu/scratchr2/static/__3602e382ef14079729b83917a837581a__/pdfs/conference/2014/Scratch%20at%20MIT%202014%20Conference%20Schedule.pdf

side and then easily compared[11]. The very quick answer of the Scratch Team at that time was that this could confuse young Scratchers. This very same mechanism, 8 years later, has been easily implemented and embedded into one of the third-party Scratch *addons* that will be described in section 3.1 (Figure 2).

3 TOOLS TO ENHANCE THE SCRATCH IDE

The code of the Scratch website and editor are open source, so the obvious solution to the need of having an enhanced tool with all the features of Scratch would be creating a Scratch clone with all the necessary additions. This has already been done. TurboWarp[12] is a much faster, higher-quality, and fully customizable version of Scratch. It can be run online or downloaded as a standalone desktop app. Even better, it is able to package Scratch projects as executables. The only drawback is that to be run on the Scratch website, a TurboWarp project must be first downloaded on your own device and then uploaded to the Scratch online IDE. Furthermore, TurboWarp still doesn't offer all the useful additions that are useful for both students and teachers.

In the following subsections we will describe two tools, a set of addons called ScratchAddons and an advanced Scratch clone called Snap! presented by the University of California at Berkeley.

3.1 Scratch Addons

The fact the TurboWarp cannot be used as an editor in the Scratch website is why all addons available in TurboWarp are also available as a Scratch browser extension for Chrome, for all Chromium-based browsers and for Firefox[13], called *Scratch Addons*[14].

The Scratch Addons development community is very active. They interact very effectively with their users and, as of December 2022, almost 150 addons affecting all aspects of the Scratch IDE and the Scratch website have been developed[15]. Almost everything in the Scratch IDE can be coloured, stretched, or hidden. In the Scratch Addons settings' page, every addon can be individually enabled (with optional sub settings, when available; Figure 3).

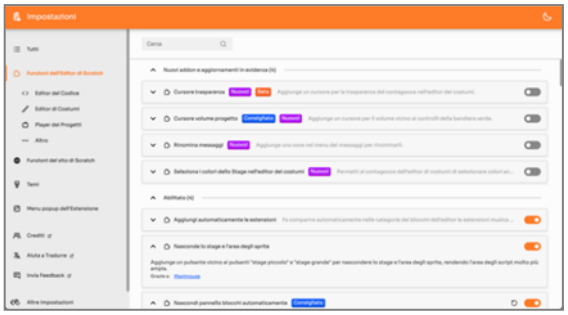

Figure 3: The Scratch Addons setting's page.

In the following sections we will describe a subset of the addons made available by Scratch Addons that can make the life of students and teachers much easier. Features that we are looking for are the following:

- IDE space management:
  - hide Stage and sprite area;
  - hide block palette;
  - customizable block shape
- saving time:
  - search bar in the editor;
  - move costumes/sounds to the top/bottom of their list;
  - pasting duplicated scripts right at the mouse position;
  - automatically add Scratch extensions to the IDE at startup;
  - "load costume" as default action for new costumes
- reducing errors:
  - disable automatic saving;
  - progress bar for projects during loading/saving;
  - confirm sprite removal;
  - fix loading of specific SVG costumes/backgrounds;
  - do not automatically run scripts after duplication;
  - do not run scripts when clicked;

[11] Note that reducing the IDE zoom would not work as a solution, as the scripts would not be readable when projected on a screen for the students.

[12] htttp://turbowarp.org

[13] the development of ScratchAddons for Safari has started on October 2022, https://github.com/ScratchAddons/ScratchAddons/discussions/3452

[14] http://scratchaddons.com

[15] Several addons described here have been proposed to the Scratch Addons community for consideration by the authors of this paper. When an addon is considered as relevant for the community, the development and availability in the official distribution of Scratch Addons is often a matter of weeks, sometime even days

  - transparent blocks when dragging them around
- user friendliness:
  - debugger;
  - onion skin in the paint editor;
  - visible mouse coordinates;
  - pause button;
  - replace block with other blocks from the same category;
  - dragging files to costumes/sounds tabs;
  - save blocks as image;
  - coloured border in running blocks;
  - jump to custom block definition;
  - rename broadcast;
  - associate icons to block category buttons;
  - fine tuning of project sounds' volume
- accessibility:
  - customizable block colours;
  - customizable block text style;
  - customizable block shape.

Figure 4: Addon code name.

Now we will see which addons can be used to solve the more common problems of students or teachers when using Scratch. In the following, we will refer to each addon with its code name, as shown in the addon description page[16] (Figure 4).

3.1.1 Enhanced Management of the IDE's Space

Whereas standard programming IDEs show the code in a linear fashion, the code of Scratch projects is organized in a 2D space. This, added to the new, taller shape of Scratch 3 blocks, makes only a few of the project scripts visible in the coding area (Figure 1). This makes difficult to have a clear understanding of the behaviour of each sprite, as the user has to scroll the coding area in all directions. Lack of space to code, due to the everything-always-visible philosophy of the Scratch IDE described in the previous sections, can be greatly improved by enabling the following addons:

- *hide-flyout*: the block palette can be hidden, either automatically or -even better- when the selected category icon is clicked again;
- *hide-stage*: add a button above the Stage to hides/shows the Stage and the sprite area;
- *custom-block-shape:* allow to reduce the size -especially the height- of the blocks.

Figure 5: The "Breakout" project in the enhanced Scratch IDE. The addons for IDE space management allow for a larger coding area and thinner blocks.

Thanks to these addons, both teachers and students can arrange in a clearer way their scripts in the coding area (Figure 5), so to better see and understand their dependences, their similarities, or their differences without having to use the zoom out feature of the Scratch IDE coding area, that will make the text of the blocks too small.

3.1.2 Saving Time

Students, especially those ones not very self-motivated as they are not interested per se in the topics of computer programming, tend to drop very quickly the study of computer science as they find annoying having to repeat the same actions without real necessity. Time spent in repetitive operations can now be reduced, so to avoid students to be annoyed by having to repeat actions without necessity, by enabling the following addons:

- *editor-devtools*: an additional search bar allows to find all "event" blocks -for examples all messages sent and received- that occur in all the scripts and all the sprites of the project;
- *editor-devtools*: when a script is duplicated with CTRL+V, it is attached to the mouse pointer, so that it can be immediately placed where desired;
- *jump-to-def*: allow to immediately find the definition of a custom block;

[16] http://scratchaddons/addons

- *load-extensions*: the categories of the extensions that are most used by the user, for example the Pen and Music categories, and that have been removed from the standard set of Scratch 3.0 categories, are automatically added back to the block palette every time the editor is open;
- *block-switching*: allow to replace a block with a related block -for example to replace *if* with *if else*; *hide* with *show*, etc- by selecting the name of a new block in the contextual menu;
- *rename-broadcasts*: allow to rename a message in a send/receive block. Much more efficient when compared to creating a new message and replacing it in all send/receive blocks;
- *ctrl-click-run-scripts* and *fix-pasted-scripts*: avoid scripts involuntarily run (causing undesired and not easily revertible changes to the Stage content) when scripts are dragged or snapped to other blocks;
- *move-to-top-bottom*: costumes and sounds can be moved right to the top/bottom of the list they belong by right-clicking them; this is really useful when, for example in animation-based projects with hundreds of costumes, dragging costumes to the top/bottom takes a long time as the costume list does not autoscroll during dragging, so that the costume must be dropped and regrasped many and many times; this also avoids problems when the costume is dropped by mistake during dragging and it is "lost";
- *blocks2image*: images of scripts can be exported in PNG/SVG format and can be used to quickly prepare slides or books for the lessons;
- *paint-by-default*: this addon allows, for example, to automatically upload a new costume/sound when the "new costume/background/sound" buttons is clicked, instead of having to select the upload option in the popup menu (Figure 6).

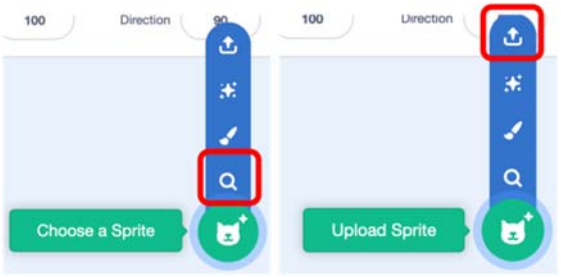

Figure 6: Default "*choose a sprite* (from the Scratch Library)" action for the "new sprite" Scratch button (left) and new custom "*upload sprite* (from your computer)" action (right).

3.1.3 Reducing User Errors

Avoiding mistakes is an excellent way to reduce the frustration and to increase the trust of the students both in their teachers and in the Scratch tool (Hertzu and Hornbæk, 2023). When errors are frequent, less motivated students tend to lose interest and to drop out (Figueiredo and García-Peñalvo, 2020). Some of the most common sources of errors can be removed from Scratch by enabling the following addons:

- *disable-auto-save*: every time we update our projects, the online Scratch IDE automatically saves all the changes, so that we do not risk losing our work if the internet connection is lost. So, when we add new features to already fully working projects, the projects will temporarily stop working if we do not save them with a new name before we start updating it. Disabling autosave is a risky operation, but this will very likely avoid us to lose working versions of our projects;
- *remove-sprite-confirm*: when a sprite is selected, a trash bin button with an "x" sign shows up at the sprite thumbnail's top-right corner. In standard Scratch, by clicking this button the sprite will be deleted without any warning. If we then delete another sprite, only the last deleted sprite will be recovered by using the Restore option of the Edit menu. So, if we delete a sprite by mistake, we are not sure that we will be able to recover it. This addons allow us to get a warning every time a sprite is deleted (Figure 7), avoiding that sprites are lost unwillingly.

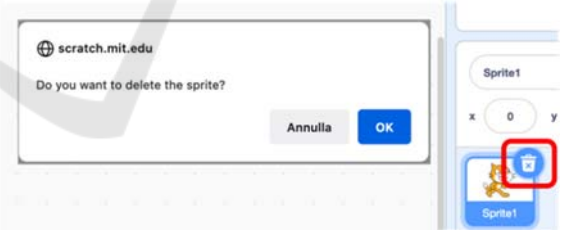

Figure 7: The warning enabled by the *remove-sprite-confirm* addon when the trash bin icon is clicked.

- *transparent-orphans*: dragged blocks can be made transparent. This is particularly handy when reporter blocks are dragged to be embedded in other blocks' arguments, as it is often difficult to see if the block, when dropped, it is going to certainly fit the argument (Figure 8, left). With this addon enabled, the "fitting" argument highlighting is visible under the dragged reporter (Figure 8, right);

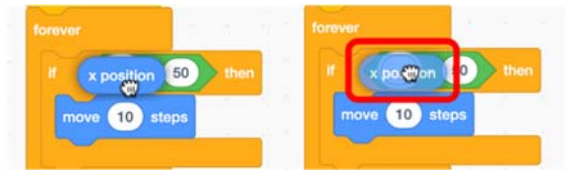

Figure 8: The not-fully-visible first argument of the green operator in a classic Scratch IDE (left) and the same highlighted argument visible behind the transparent reporter (right) when the *transparent-orphans* addon is enabled.

- *progress-bar*: when a large project is loaded or saved, it can take a while to complete the operation. In the meantime, there are no signs that everything is going as expected. When there are network problems, the "saving in progress" To avoid that students will stop the loading/saving process just before it is completed, a progress bar giving a clear indication of how the operation is proceeding can be enabled (Figure 9);

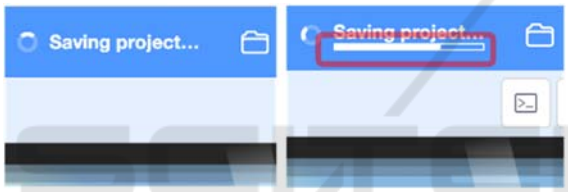

Figure 9: The standard Scratch saving indicator (left) and the richer indicator enabled by the *progress-bar* addon (right).

- *fix-uploaded-svgs*: SVG files produced by several drawing apps (ArtWorks, Affinity Designer, etc.) are not correctly loaded by Scratch. Enabling this addon, all SVG files are correctly imported as Scratch costumes.

3.1.4 Enhanced User-Friendliness

Scratch IDE has been designed to be very user-friendly. Everything is clearly visible and organized, and the IDE is almost self-explanatory. Adding further user-friendly features to the Scratch IDE makes both programming with Scratch and understanding Scratch scripts even easier, so users are more likely going to get interested in exploring the environment and teachers can more clearly explain the behaviour of their scrips. New useful features in this direction can be added by Scratch Addons to the Scratch IDE by enabling the following addons:

- *debugger*: the debugger is a fundamental tool in every programming language. This addon allows users to run Scratch scripts step by step. This feature allows the teacher to highlight the behaviour of each block when they are illustrating a given script, or to clarify the interaction between several scripts. It also allows the user to identify problems caused by an unforeseen incorrect interaction between several scripts;
- *pause*: a pause button is added between the start and stop buttons, right above the Stage. When it is clicked it pauses the project execution until the user press it again. It is useful when a teacher is explaining how a given project works, allowing them to momentarily pause the execution, show the scripts or describe their inner working, then resuming the execution;
- *editor-stepping*: when the debugger is not enabled, the possibility of highlighting the block currently executed, that is lost since Scratch 2, is brought back by this addon; even if it is less precise than the original Scratch 1 feature, it is very useful to understand more easily how an algorithm works;
- *onion-skinning*: Scratch is a programming environment based on media. If the teacher/student is interested in creating good quality animations -useful for science or storytelling, where you need to perfectly position multiple costumes of a single sprite [17] - the possibility of seeing transparent overlays of the previous/next costume(s) is very useful. This is something that has been often requested to the Scratch Team in the past as a necessary addition to the Scratch paint editor[18];
- *mouse-pos*: whereas in Scratch 1 and 2 the mouse location was always visible under the Stage, this reference disappeared in Scratch 3. As students often struggle to clearly discriminate between the x and y coordinates (Battista, 2007; Battista et al., 2017), this feature it is very useful when the coordinates of the Stage are explained, or to know what the exact final position of a sprite movement is going to be without having to actually move the sprites around. When this addon is enabled, the x/y coordinates of the mouse are clearly shown right above the Stage;
- *drag-drop*: both pictures for new costumes or new backgrounds, and audio files for new sounds can be dragged from the PC folder to the

[17] See for example projects https://scratch.mit.edu/projects/554574719 or https://scratch.mit.edu/projects/622657902.

[18] https://scratch.mit.edu/discuss/topic/556991

costume/background/sound lists in the Scratch IDE. This is particularly handy for young and non-CS-majoring students, that are used to their touch devices and are not familiar with the tree-structure of the folders in a file system (Chin, 2021);

- *block-palette-icons*: in order to learn faster how to associate category colours to their functions, icons are added inside every coloured category button (Figure 10) as widely recognized (Lodding, 1983; Harrison et al., 2011);

Figure 10: Standard coloured category buttons in the Scratch IDE (left) and the enhanced category buttons with an icon inside (right). For example, the *Looks* category is represented by an eye-shaped icon, the *Sound* category by a loudspeaker, etc.

- *vol-slider*: when explaining projects with sounds or voice, teachers may want to keep the sound/voice running but at the same time to reduce their volume to allow students to clearly hear their voice. Teachers can use this addon to make a volume slider visible right above the Stage, at the right of the stop button. In this way they can have a finer control than just switching off the browser volume and an easier way to handle the volume instead of using the volume controls of their OS.

3.1.5 Accessibility

As up to 20% of students can be affected by dyslexia/dyscalculia (Federici et al. 2022), the Scratch Team started in October 2022 to study possible modifications to the Scratch IDE to make it more accessible. Their first proposal (currently under review at their Scratch Lab website[19]) was to change the colours of Scratch blocks, so that people with low vision can read them better (Figure 11). If accepted, this proposal will make the future Scratch blocks "easier to read *for all Scratchers*".

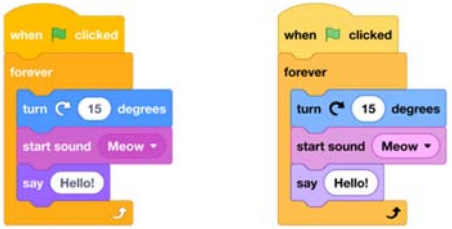

Figure 11: Standard Scratch blocks (left) and the new, more accessible proposal (right).

The same result can be obtained by using a specific addon. The advantage of using an addon is twofold: if Scratch Addons users want to immediately use more contrasted block colours, they can do that stand they can keep using all their customizations that would not be available in the Scratch Lab website or in the future Scratch; if, instead, they do not like the new proposal (that, as already said about Scratch evolution, if accepted would become the standard for everyone) the can go back to the current block colours, by means of the same addon. Useful addons of Scratch Addons for dyslexic students are the following ones:

- *editor-theme3*: it allows you to choose your own colour for each block category, so to create a better contrast, as in Scratch Lab (Rello and Bigham, 2017; Federici et al., 2022);
- *custom-block-text*: it allows you to select a thicker font with an optional shadow, so to create a more readable block text (Rello et al., 2014; Federici et al., 2022);
- *custom-block-shape:* it allows you to increase the size -especially the height- of the blocks, so to increase the line spacing (Madhavan et al., 2016; Federici et al., 2022);

3.2 Snap!

A lot has already been done by Scratch Addons developers, but many, major or minor, enhancements to the Scratch IDE are still missing. Some addons - like the *pause* button, the *debugger*, the possibility to enlarge/reduce the space taken by the block palette or the Stage- seem borrowed from a famous Scratch modification whose developer, Jens Moenig, started in 2007 by adding strings and lists to Scratch 1.2 and then in 2010 moved to create BYOB, its own first Scratch modification[20] where users could create their own blocks (Harvey and Moenig, 2010). In 2011, in collaboration with Brian Harvey, Moenig started the development of *Snap!*[21], a very advanced Scratch

[19] http://lab.scratch.mit.edu/contrast-blocks/
[20] https://snap.berkeley.edu/old_site/old-byob.html
[21] http://snap.berkley.edu

modification similar to Scratch but packed with many powerful mechanisms.

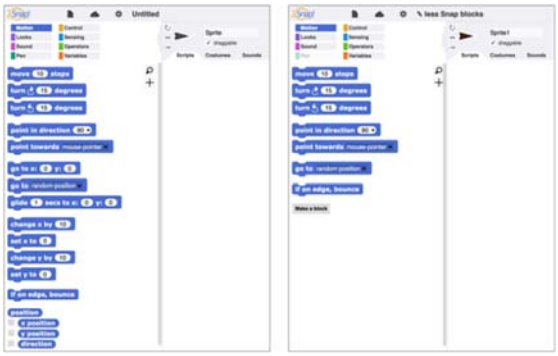

Figure 12: Standard Snap! blocks of the Movement category (left) and a reduced set of Movement blocks that can be easily created in Snap! (right).

The Snap! 8.0.0 environment has now many other interesting features. Among them, after being requested by many teachers using Snap! for their lessons[22], there is the possibility to selectively hide the blocks in the palette, so that beginners are not overwhelmed by the amount of available -but still unknown- blocks (Figure 12). Incidentally, this possibility is also very useful for students with dyslexia, that can avoid the effect of *visual crowding* of the IDE (Bellocchi, 2013; Federici et al., 2022).

4 FUTURE ENHANCEMENTS

Not all the advanced features that would help us as teachers and would -very likely- help our students when we use Scratch in the lessons of our Computer Science courses, or that could help in general students with dyslexia (Federici et al. 2022) are available in ScratchAddons or Snap!.

One very important feature that is still missing is for example the possibility to show the coding areas of several sprite at the same time (Figure 14). This possibility would allow us and our students to compare the scripts of several sprite, to easily notice small differences in similar scripts of different sprites or to immediately see the connection among scripts from different sprites (e.g., when using broadcast or sensors). As of now, the ways to get a similar result are the following ones:

- using slides: the visualization in this case is limited by the limited space of the slide, but every necessary element can be clearly organized and highlighted. Another problem with this technique is that the visualization is static;
- using several Scratch windows at the same time: thanks to the *hide-flyout*, *hide-stage*, and *custom-block-shape,* more windows can be put side-by-side, still making their coding area visible. The visualization is dynamic but, due to the size of the Scratch IDE frame, when the scripts are compared on a projected screen that must be clearly visible at a distance, no more than two windows can be used (Figure 13).

Figure 13: Opening two Scratch windows to see the scripts of two sprites side-by-side.

Clearly, organizing the windows so to get the correct visualization takes some time that is unfortunately wasted, so it cannot be done on a need basis during the lesson. The correct way of doing this could be applying the hide-flyout and hide-stage addons (Figure 14).

Figure 14: Showing three sprite coding areas instead of a single coding area.

By using these addons more sprites -for example by selecting them by CTRL/CMD+click on their thumbnails in the sprite area- could be allowed to show several coding areas at the same time.

Last, to improve the comprehension of students with dyslexia (Thompson, 2019; Federici et al., 2022), the scripts could be read out loud, either by

[22] "The Future of Snap!", Snap!Con 2019 conference, Stuttgart, https://youtu.be/YCqzXAIAhJw

pressing a key combination or by selecting an option in the script's contextual menu (Figure 15).

Figure 15: New option to read scripts out loud in the script contextual menu.

Using the accessibility features already available in modern OSs (e.g., the "Narrator" in Windows or the "Spoken Content" in macOS) would not be sufficient to solve this problem, as only the arguments of the blocks would be read out loud, not the block text. To solve the problem, a better integration with the accessibility features is required. This is not true for the reverse need, that is entering text in the block arguments. Activating the accessibility features of the OS (e.g., "Voice Typing" in Windows or "Dictation" in macOS) allow to solve very well this problem (Federici et at. 2022). The integration with the accessibility features could be improved by adding a microphone icon in the block arguments when they are clicked (Figure 16).

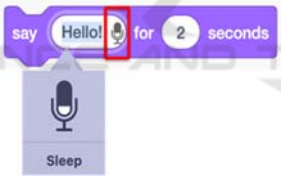

Figure 16: Integration and usage of accessibility features to enter the values of block arguments.

Clearly, to make these future enhancements easily available to all users, they will have to be implemented as addons that, once installed in the browser, will allow to further extend the features of the online version of Scratch.

5 CONCLUSIONS

Using a block language like Scratch is a great help when teaching either to young students or to students that are not majoring in STEM subjects. However, the strengths of Scratch, like being a very simple visual environment, or being available in more than 60 languages, are not sufficient to give to all students and teachers all the necessary tools in order to make the explanation and the comprehension process quick and easy.

Due to the will of the Scratch Team not to add too many new features to Scratch, fulfilling their main goal of keeping it simple, enhancing the online version Scratch with third-party mechanisms is an excellent way of making available a more powerful IDE still keeping all the advantages of the online website and community.

The additions proposed in this work to make Scratch easier to use for students and teachers have been mainly independently developed by the Scratch Addons community (only a small part of them being requested by the authors of this work) but aim to bring back useful features of the Scratch IDE that have been lost in more recent releases or to add new features that are felt necessary even by other users. For each feature that we have personally verified as being useful for students and teachers, the didactic motivations behind our strong support have been presented.

REFERENCES

Battista, M. T. (2007). *The development of geometric and spatial thinking*. In F. K.Lester, Jr. (Ed.), Second handbook of research on mathematics teaching and learning, Vol. 2 (pp. 843-908).

Battista, M. T., Winer, M. L., Frazee, L. M. (2017). *How spatial reasoning and numerical reasoning are related in geometric measurement*. In Geometry and Measurement, Proceedings of the 39th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education. Indianapolis, IN: Hoosier Association of Mathematics Teacher Educators.

Bellocchi, S. (2013). Developmental Dyslexia, Visual Crowding and Eye Movements. In *Eye Movement: Developmental Perspectives, Dysfunctions and Disorders in Humans*. Nova Science Publishers.

Chin, M. (2021). *File not Found*. The Verge. https://www.theverge.com/22684730/students-file-folder-directory-structure-education-gen-z (last retrieved on 02/16/2023).

Federici, S., Gola, E., Brau, D. and Zuncheddu, A. (2015). *Are Educators Ready for Coding? From Students back to Teacher: Introducing the Class to Coding the Other Way Round*. In Proceedings of CSEDU 2015, Lisbon, Portugal, Vol. 1 (pp. 494-500).

Federici, S., Gola, E., Giorgi, M., and Sergi, E. (2022). *Enhancing a block-based IDE to improve learning of computer programming for people with and without dyslexia and/or dyscalculia*. In Proceedings of Didamatica 2022, Nov 10-11th, Milano.

Federici, S., Medas, C. and Gola, E. (2018). *Who Learns Better: Achieving Long-Term Knowledge Retention by*

Programming-Based Learning. In Proceedings of CSEDU 2018, Funchal, Portugal, Vol. 2 (pp. 124-133).

Federici, S., Sergi, E., and Gola. E. (2019). *Easy Prototyping of Multimedia Interactive Educational Tools for Language Learning based on Block Programming*. In Proceedings of CSEDU 2019, Heraklion, Greece, Vol. 2 (pp. 140-153)

Figueiredo, J., and García-Peñalvo, F. J. (2020). *Increasing student motivation in computer programming with gamification*. In Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON), Porto, Portugal (pp. 997-1000).

Harrison, C., Hsieh, G., Willis, K. D. D., Forlizzi, J., and Hudson, S. E. (2011). *Kineticons: Using Iconographic Motion in Graphical User Interface Design*. In Proceedings of the 29th Annual SIGCHI Conference on Human Factors in Computing Systems, Vancouver, Canada (pp. 1999-2008). ACM.

Harvey, B., and Moenig, J. (2010). *Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists?* Proceedings of Constructionism 2010, Paris.

Hertzu, M., and Hornbæk, K. (2023). *Frustration: Still a Common User Experience*. In ACM Transactions on Computer-Human Interaction. http://dx.doi.org/10.1145/3582432.

Lodding, K. (1983). *Iconic Interfacing*. IEEE Computer Graphics and Applications, Volume 3, issue 2 (pp. 11-20).

Madhavan, I., Sharanjeet-Kaur, Hairol, M. I., Mohammed, Z. (2016). *Spacing improves reading in dyslexic children*. In Asia Pacific Journal of Developmental Differences, Volume 3, Issue 1 (pp. 3-20). DOI: 10.3850/S2345734114000183.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). *The Scratch Programming Language and Environment*. In ACM Transactions on Computing Education. Volume 10, issue 4.

Rello, L., and Bigham, P. J. (2017). *Good Background Colours for Readers: A Study of People with and without Dyslexia*. In Proceedings of The 19th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 72-80). ACM. https://doi.org/10.1145/3132525.3132546

Rello, L., Saggion, H., and Baeza-Yates, R. (2014). *Keyword Highlighting Improves Comprehension for People with Dyslexia*. In Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (pp. 30–37), Association for Computational Linguistics.

Thompson, R. H. (2019). *HAWK/KW: An Online System for Studying Dyslexic Children's Reading, Writing, and Programming*. https://courses.cs.washington.edu/courses/cse154/20su/staff/about-me/rob-thompson/Thesis%20-%20Rob%20Thompson.pdf (last retrieved on 02/17/2023).