Designing Collaborative ScratchJr for Multi-touch Tabletops

Pedro Paredes Barragán*,
Maximiliano Paredes Velasco†, Jaime Urquiza-Fuentes‡,
Guillermo J. García-Delgado Álvarez§
Universidad Rey Juan Carlos,
Madrid, Spain
Email: *pedro.paredes@urjc.es,
†maximiliano.paredes@urjc.es, ‡jaime.urquiza@urjc.es,
§guj.garciadelga.2018@alumnos.urjc.es

Craig Anslow‖, Michael Homer**
Victoria University of Wellington
Wellington, New Zealand
Email: ‖craig.anslow@vuw.ac.nz, **michael.homer@vuw.ac.nz

***Abstract*—Teaching fundamental aspects of computer science at pre-university levels has been a relevant topic for several years. Visual programming environments based on block-based languages, such as Scratch, have gained prominence due to their ability to simplify entry into the complex field of programming. This work focuses on ScratchJr, a user-friendly environment specifically designed for early childhood and primary education.**

Collaborative learning is one of the most effective and widely used methodologies in classrooms. With this in mind, our research centers on designing a collaborative interface that enables group work among multiple students within a shared workspace. This workspace will be deployed on a multitouch table, leveraging the advantages provided by such devices. These advantages include enhanced collaboration and the need for hands-on experiences for young learners.

We present the interface design process, which began with a participatory and collaborative approach involving students from early childhood and primary education degrees. Following this, we proceeded with both low-fidelity and high-fidelity prototyping stages.

I. Introduction

Teaching programming to young children is not a novel idea, as is demonstrated by the Logo language, invented in the 60's [1]. However, this trend has become widespread in the last two decades. The term STEM (Science, Technology, engineering and Maths) was coined by Judith Ramaley [2] to identify fields with a lack of knowledgeable workers. This necessity was highlighted by many events, e.g. The US NAP report regarding dedicating efforts to create high quality jobs related to science and technology [3] or the President Obama's 2016 State of the Union Address [4]. Within this context, technology related contents have more presence not only in universities or colleges but also in schools. For instance, learning to program is currently a common task in most of schools from many different countries.

But students from schools can hardly cope with the complexity of textual programming. This is the reason because the number and presence of visual programming environments has been increased. Within this type of programming environments, the block-based ones [5] have been, by far, the most successful.

Block-based programming languages use a programming-command-as-puzzle-piece metaphor. This approach allows to provide the user with visual cues, based on shape and color of the pieces, about how they can be assembled. And these pieces are assembled using drag-and-drop interactions to build the programs [5]. Nowadays, block-based programming is widely used to teach programming concepts within different scopes, e.g. the MIT APPInventor[1] for creating APPs of mobile devices, the LEGO MINDSTORM Block language[2] for educational robotics, the well known Scratch [6] for primary and secondary school or the ScratchJr language for early primary school [7]. In fact, many toys are distributed together with some kind of programming facilities [8]. The focus of this work is on ScratchJr, which represents a way to introduce young children to programming. ScratchJr was specifically developed with a user-friendly interface that makes it easy for young learners to get started.

In addition, collaborative learning is a well-known and effective methodology used in these contexts [9]. Collaborative learning is a pedagogical technique in which a topic or subject is addressed by teams of students who actively participate in the learning process. Thus, by working in groups, students strengthen different skills such as teamwork, logical reasoning or creativity. The key aspects of collaborative learning is the mutual influence between peers and the equality of participation in tasks [10]. Despite the potential of the collaborative approach, the effects of collaborative programming in education are still under research [11], e.g. some pair programming experiences show no advantages against a non-collaborative approach [12].

Finally, multi-touch tabletops have shown their advantages in supporting collaborative activities and enhancing learning experiences [13]. Despite its potential benefits, few approaches (e.g. [14], [15]) have been found using the combination of block-based programming, collaborative learning, and multi-touch tabletops. This work describes the design of a prototype

[1]https://appinventor.mit.edu/
[2]https://makecode.mindstorms.com/blocks

Fig. 1. Participatory co-design session with students.

where young students can work together to create their own ScratchJr programs while using the tabletop as a shared workspace. The building process of this prototype consists in two phases: the design of a low-fidelity prototype using a cooperative design approach and the design of a high-fidelity prototype.

II. Design of the Low-Fidelity Prototype

The design process of the low fidelity prototype followed three steps. Firstly, a participatory co-design process with students was performed. Secondly, the prototypes produced by the students were evaluated. Finally, the low-fidelity prototype was designed using the main requirements of the application, the students' prototypes and their evaluations.

The participatory co-design process engaged four instructors and 23 students from two different degrees: early childhood education (five students), and computer science (18 students). The former shared their knowledge about the features of children between 3 and 6 years old and their experience using ScratchJr, while the latter put into practice the principles of Human Computer Interaction and User Interaction design. Students were divided in five groups mixing students from both degrees. During a 2 hours long session, they produced low-fidelity prototypes for the use of ScratchJr on a multi-touch surface from a collaborative perspective, see Fig. 1.

Prototypes were designed with markers, cardboard, scissors, and glue. We collected information through video recordings of their explanations, photographs, and the prototypes themselves that were evaluated and used to produce the one presented in this work.

A. Evaluation of Students' Prototypes

One of the prototypes was discarded as the group had not had time to create a useful artifact for our research. The other four were evaluated by three experts in collaborative interface design.To achieve this, recordings, photographs and the prototype itself were used.

As can be seen in Fig. 2, this prototype work screen is centered on a smaller central box, leaving the sides for extension of activities. At the top are control buttons: settings/start execution/undo/communicate (an envelope) and a main menu button. At the bottom there are several empty buttons for function extension. Finally, much more visible are four pencils that suggest editing. Clicking on them hides only areas of the table (an L-shaped corner of the table) but the area of the program scene is always visible.

Advantages:

- Use full-screen mode.
- Four sides of the table with the pen to allow anyone to edit.
- The central part visualizes the look/execution of what they build.
- The basic construction functionalities should appear on the sides and try not to hide the central part.
- Infrequent functionalities (configuration and templates) in the central part, appear and disappear pop-up.

Disadvantages:

- Corner space is not used most of the time.
- Large "CODE" keys do not have good orientation.
- Reserve central fixed space for things you don't often do: add characters or backgrounds.

In the second prototype (Fig. 3) the entire screen is divided into four individual spaces. Each user has a semicircle with a play button, a character button, and a background button. The character and background controls open pop-ups with the available characters and backgrounds. When you select a character, it stays in your personal area and you can assign actions to it. By pressing the play button, we can see the result of those actions in the background we have chosen. The execution window is also pop-up and includes scroll buttons to place the character in the place of the scenario where we want, and a stop button to stop the execution.

Advantages:

- Pop-ups allow the space not to be cluttered with many components.
- That space can be used to display different windows with the relevant information at any given time.
- Minimalist design, with its advantages and disadvantages.

Disadvantages:

- It doesn't seem like it's meant to collaborate, but rather for one to be in control and take turns.
- The low persistence of information on the screen can be an obstacle to editing actions and seeing the result at the same time (having both things visible on the screen).
- The background icon is an F ("Fondos" is the translation of background in Spanish), not very intuitive for young children.

The third prototype (Fig. 4) seems to focus on a workspace with no individualized spaces for different users. Basically, it can be divided into a scene area (left half) and a character area (right half). There is a blank slot with unused available space.

Advantages:

- Can be worked on simultaneously with the tasks of: scenarios and characters.
- Leveraged Scratch controls and organization (leverage user insights): go back to start, run, run from start again,

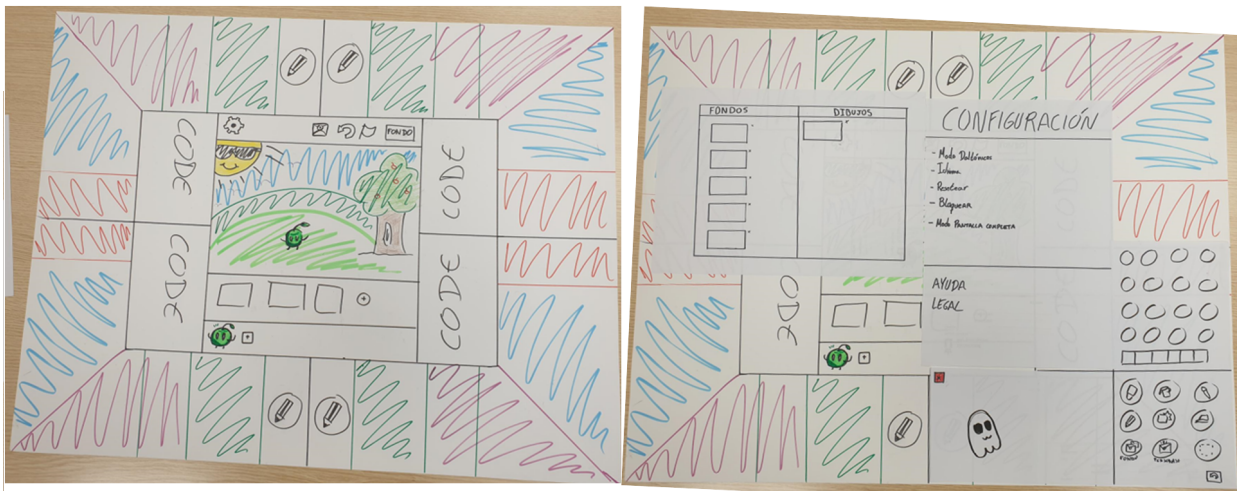

Fig. 2. Prototype 1

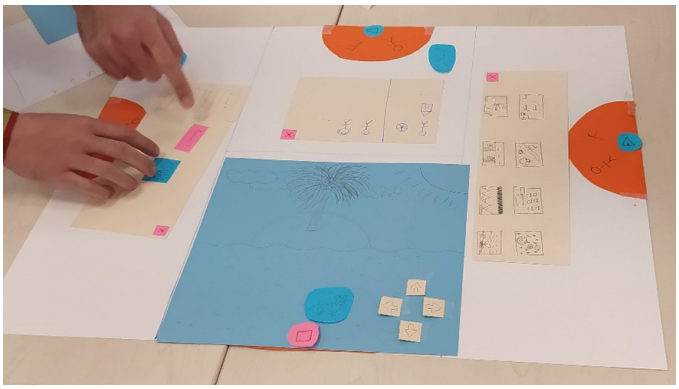

Fig. 3. Prototype 2

Fig. 4. Prototype 3.

view grid, view full screen, part categories, parts, and program.

- Backgrounds are editable (they can be silhouettes that can be colored, text added, etc.). An interesting option for young children.
- Restricted to 4 characters, for ScratchJr it's more than enough. Each character button is differentiated by the color that may be associated with the character's main color.
- The character area could be shared among several, so that two could be viewed/programmed at a time (more than two may be more complicated), and the area could also be used to the maximum depending on the number of characters you want to see at once.
- Character can be customized with classic editing controls.

Disadvantages:

- Users have to share the entire space, no different workspaces per user.
- Due to the unique orientation it would only support two users, three at most.
- There is no control to stop the execution.
- The background palette doesn't need to be there all the time, it takes up space.
- Very little space for the work area that needs the most attention, character behavior design.

The latest prototype (Fig. 5) is based on individual workspaces in the corners and a central common space. Controls are concentric circular areas, which expand from the execution flag. Each area serves a different purpose.

Advantages:

- "Edit control" can be transferred.
- Aesthetic concentric controls.
- Aesthetic tree controls.
- Everyone sees in their orientation how the work turns out.

Disadvantages:

- There doesn't seem to be any simultaneous work, only one can edit at any given time, the rest just see.

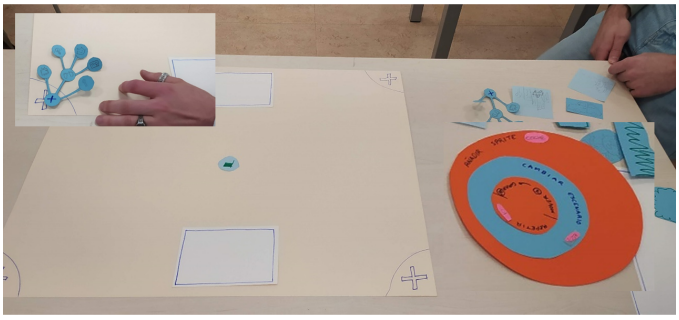

Fig. 5. Prototype 4

- There is no information about program execution controls, nor detailed editing of anything.

B. Low Fidelity Prototype

The environment is divided into up to five workspaces (see Fig. 6) based on territoriality designs [16] including one shared workspace (SW), and a group of user workspaces (UWs) – between two and four– around the SW, ensuring that none of the users has an inverted view of the SW. The border of each UW has a different color, it is used to identify objects visible in the rest of the workspaces that are being modified by a user. Two gestures communicate SW and UWs: a throwing gesture on an object from a UW to the SW publishes it in the SW, and a drag and drop on an object from the SW to a UW allows a user to modify the object in her/his UW.

An important design decision is to take advantage of the user's previous knowledge about ScratchJr. Therefore, the environment imitates the interfaces of ScratchJr for background and character (appearance and scripts) editing and script execution. The SW shows the state of the work accepted by all users. It also allows users to create the scenes, the background, and the characters or modify existing ones. In addition, the UW provides two controls to accept or reject changes proposed by other users. Finally, the collaborative environment has to manage different conflicts regarding scene editing. Thus, within the same scene, the following objects/properties cannot be simultaneously changed by two or more users: background, the appearance of a concrete character, and the scripts associated with a concrete character.

III. Description of the High-Fidelity Prototype

A new prototype has been developed taking into account the low-fidelity prototype proposed previously. The new proposal is a high-fidelity prototype and aims to show the main features of the future software, with a focus on the user interface. The high-fidelity prototypes are often interactive and they may include visual elements such as colors, typography and layout of the elements, all of which aim to simulate the user's real experience with the final software product. The following sections detail the adaptation process from low to high fidelity and the user interface of the high-fidelity prototype. This prototype was developed using Figma[3], a prototyping tool centered on vector graphics editing.

A. Adaptation of the resulting co-design product

The prototype generated during the co-design process has been adapted to create the high-fidelity prototype. The adaptations made are as follows:

- It is decided to maintain a collaborative area (SW) at the top of the user interface, and several individual spaces (UW), one for each user, at the bottom and the left and right sides. However, a new section has been added between these areas to manage the user during the collaborative session. This section shows the number of users connected to the session and allows for adding and removing users, dynamic adjusting the layout of the screen according to the number of users.
- The UW spaces are distributed along the bottom and sides of the screen. These spaces can be resized depending on the number of users, with a maximum capacity of four users. This allows for individual work areas to be extended or reduced to make the most of the available screen space.
- The features or functions are organized into two types: 1) individual functions, which affect to resources of UW space, and 2) collaborative function, which affect to group resources.
- The color coding remains consistent. When students log into the system, they are assigned a specific color. This color is utilized to highlight their personal area and to identify which student has submitted a proposal.
- Adding scenes and characters to the program is maintained. However, the management of these elements, such as editing and removal, is not included in order to simplify the prototype. It is important to note that the objective of a high-fidelity prototype is to obtain a simulation of the product, rather than a final and complete one.
- The management of interaction conflicts in a collaborative digital environment is a key aspect. Identifying the most important interaction conflicts is crucial for adapting the resulting product of co-design process to high-fidelity version. Obviously, the list of potential conflicts may be very long. Therefore, some user actions are restricted to reduce interactions conflicts. Specifically, editing scenes is not permitted in the high-fidelity prototype to avoid conflicts.

B. User interface

The user interface of the high-fidelity prototype is adapted to the number of users connected. Figure 7 shows a screenshot of the prototype with three users creating a basic script. The user interface is structured into three main parts. Firstly, two buttons situated in the upper left and right corners of the screen allow access to the home screen and settings, respectively. Three

[3]https://www.figma.com

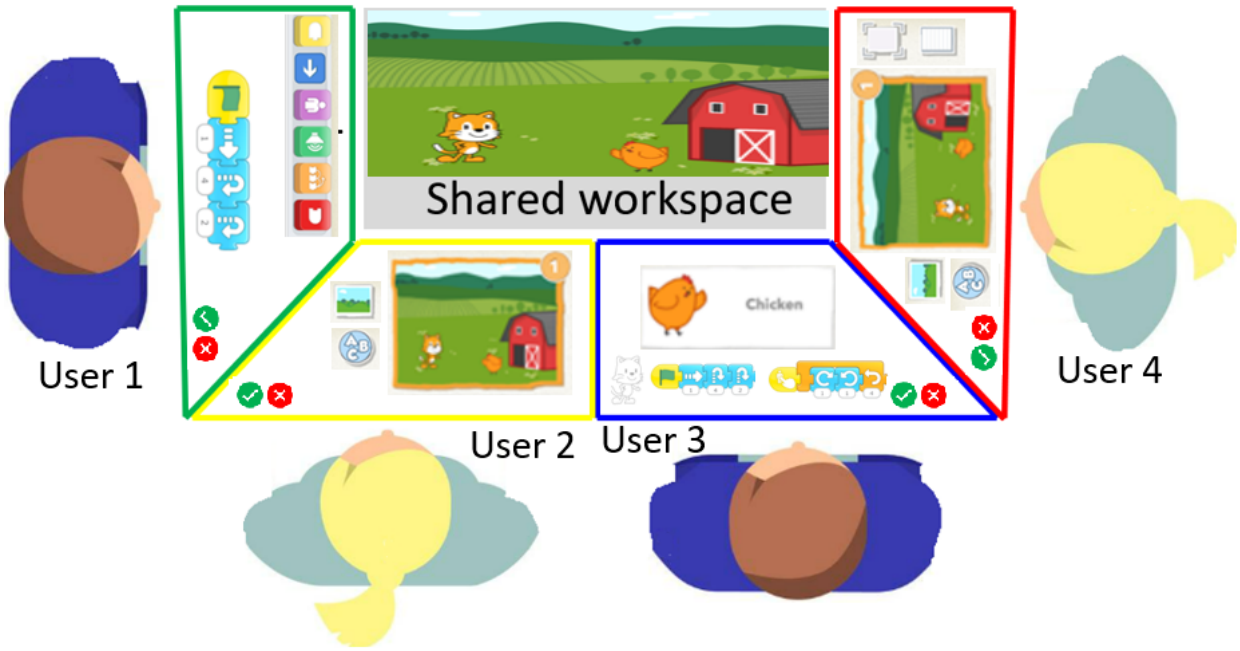

Fig. 6. Global view of the environment with four UWs.

features can be adjusted through the settings button: volume, brightness, and notifications. The collaborative space features a principal area to display the execution of scripts, which occupies the most space. Within this area, several buttons from ScratchJr are displayed to manage script execution and design scenes, including starting execution, restarting execution, and accessing the scene gallery (see Figure 7, marked as A). Secondly, in the middle of the user interface, two buttons manage connected users. Figure 7, marked as B, shows these buttons labeled with the symbols "+" and "-", which allow users to be added or removed from the session. Additionally, an emoticon representing each connected user is displayed in this area. Finally, several individual spaces are shown (see Figure 7, marked as C). These spaces offer a block panel to create scripts individually. Once again, the blocks have a similar look and feel to those of ScratchJr, leveraging users' previous experience. The blocks are grouped into several categories such as movement, sound, messages, etc. When a user selects a category, the corresponding blocks are displayed. The high-fidelity prototype implements only some of these blocks: four basic movements (up, down, right, and left), start and end blocks. Additionally, the prototype supports adding characters. The high-fidelity prototype implements a notification system to coordinate user proposals within the group. When a user proposes a change to the script, a notification is generated for all users. This notification must be accepted or rejected by the rest of the users in order for the proposed change to be applied in the collaborative space.

IV. Conclusions and Further Steps

Tangible interfaces seems to be a promising line of work, although more research is needed to clarify its impact on programming learning experience [11]. This study leverages a multi-touch tabletop as the primary tangible interface to facilitate collaborative learning of block-based programming. It introduces a high-fidelity prototype aimed at teaching ScratchJr to young students. A co-design process involving 23 students from diverse disciplines was conducted in the classroom to develop several low-fidelity prototypes. These prototypes were subsequently analyzed for usability, culminating in the development of the high-fidelity prototype. The prototype has two main spaces: one for collaborative work and another for individual tasks such as programming scripts and character creation. The following steps of this project will be, firstly, developing a software that implements the high-fidelity prototype. This prototype will be empirically evaluated with pre-service teachers and young students. Finally, its feedback will guide the development of the final application.

Acknowledgment

The authors acknowledge the collaboration of María del Carmen Lancho Martín and Iván Ramírez Díaz, the teachers of the HCI course, for their support during the co-design session. This work has been co-funded by the following grants: Ayuda Puente 2023, URJC, Ref. M3035; Proyectos de Innovación Educativa URJC Ref. PIE23-157; and PID2022-137849OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU.

Statement of use of generative AI and AI-assisted technologies in the drafting process. During the preparation of this study, the authors used MS Bing Chat to improve the readability of the language in English. After using this tool, the authors have reviewed and edited the content as necessary and take full responsibility for the content of the publication.

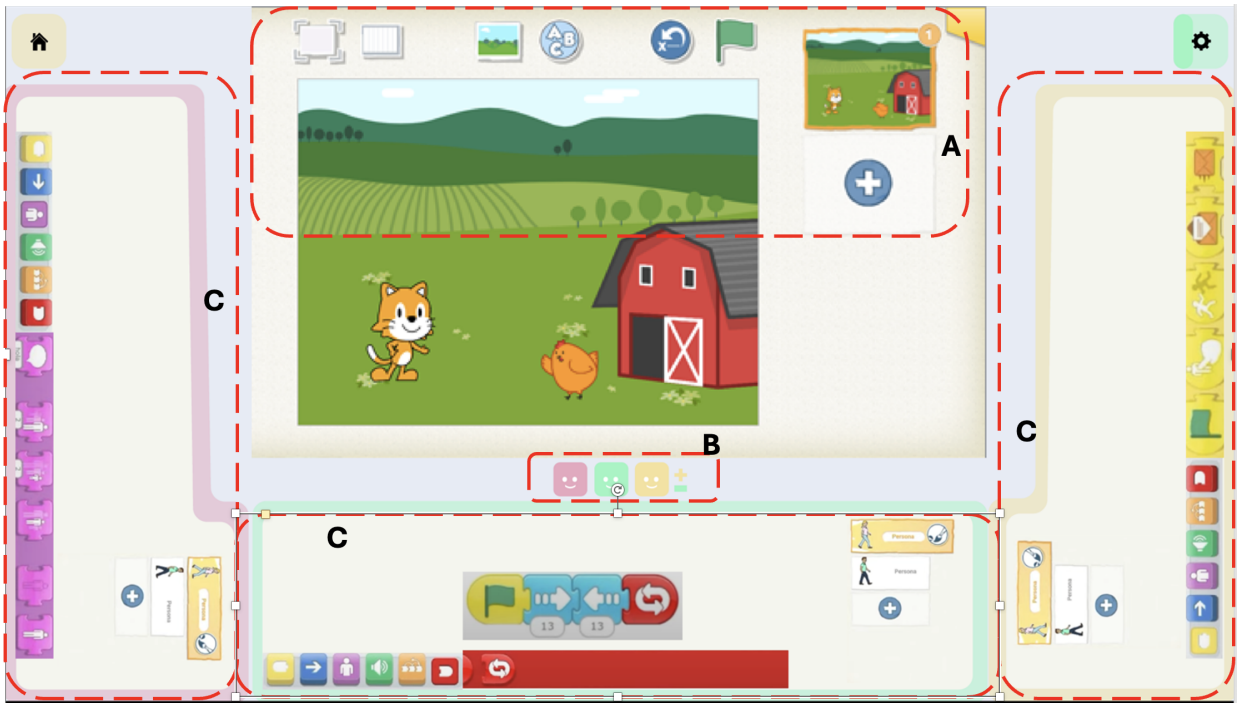

Fig. 7. High-fidelity prototype.

References

[1] C. Solomon, B. Harvey, K. Kahn, H. Lieberman, M. L. Miller, M. Minsky, A. Papert, and B. Silverman, "History of logo," *Proceedings of the ACM on Programming Languages*, vol. 4, no. HOPL, jun 2020. [Online]. Available: https://doi.org/10.1145/3386329

[2] J. Hallinen, "s.v. stem," in *Encyclopedia Britannica*, 2024. [Online]. Available: https://www.britannica.com/topic/STEM-education

[3] N. A. of Sciences, N. A. of Engineering, and I. of Medicine, *Rising Above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future*. Washington, DC: The National Academies Press, 2007. [Online]. Available: https://nap.nationalacademies.org/catalog/11463/rising-above-the-gathering-storm-energizing-and-employing-america-for

[4] T. O. W. House, "President obama's 2016 state of the union address," 2016. [Online]. Available: https://medium.com/@ObamaWhiteHouse/president-obama-s-2016-state-of-the-union-address-7c06300f9726.fky3fqa7g

[5] Y. Lin and D. Weintrop, "The landscape of block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming," *Journal of Computer Languages*, vol. 67, p. 101075, 2021. [Online]. Available: https://doi.org/10.1016/j.cola.2021.101075

[6] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, p. 60–67, nov 2009. [Online]. Available: https://doi.org/10.1145/1592761.1592779

[7] L. P. Flannery, B. Silverman, E. R. Kazakoff, M. U. Bers, P. Bontá, and M. Resnick, "Designing scratchjr: Support for early childhood learning through computer programming," in *Proceedings of the 12th International Conference on Interaction Design and Children*, ser. IDC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 1–10.

[8] J. Clarke-Midura, V. Lee, J. Shumway, and M. Hamilton, "The building blocks of coding: a comparison of early childhood coding toys," *Information and Learning Sciences*, vol. 120, no. 7/8, pp. 505–518, 2019. [Online]. Available: https://doi.org/10.1108/ILS-06-2019-0059

[9] H. Jeong, C. E. Hmelo-Silver, and K. Jo, "Ten years of computer-supported collaborative learning: A meta-analysis of cscl in stem education during 2005–2014," *Educational Research Review*, vol. 28, p. 100284, 2019.

[10] A. O'Donnell and C. Hmelo-Silver, "Introduction: What is collaborative learning?" in *The International Handbook of Collaborative Learning*, C. Hmelo-Silver, C. Chinn, c. Chan, and A. O'Donnell, Eds. Routledge, 2013, ch. 1, pp. 1–15.

[11] L. Silva, A. Mendes, and A. Gomes, "Computer-supported collaborative learning in programming education: A systematic literature review," in *2020 IEEE Global Engineering Education Conference (EDUCON)*, 2020, pp. 1086–1095.

[12] M. Colleen, "Is pair programming more effective than other forms of collaboration for young students?" *Computer Science Education*, vol. 21, no. 2, pp. 105–134, 2011. [Online]. Available: https://doi.org/10.1080/08993408.2011.579805

[13] M. Mateescu, C. Pimmer, C. Zahn, D. Klinkhammer, and H. Reiterer, "Collaboration on large interactive displays: a systematic review," *Human–Computer Interaction*, vol. 36, no. 3, pp. 243–277, 2021.

[14] B. Selwyn-Smith, C. Anslow, M. Homer, and J. R. Wallace, "Co-located collaborative block-based programming," in *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2019, pp. 107–116.

[15] B. Selwyn-Smith, C. Anslow, and M. Homer, "Blocks, blocks, and more blocks-based programming," in *Proceedings of the 1st ACM SIGPLAN International Workshop on Programming Abstractions and Interactive Notations, Tools, and Environments*, ser. PAINT 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 35–47.

[16] S. D. Scott, M. S. T. Carpendale, and K. Inkpen, "Territoriality in collaborative tabletop workspaces," in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 294–303.