Pilas Bloques: a Scenario-Based Children Learning Platform

Alfredo Sanzo
Fundación Dr. Manuel Sadosky and
Universidad de Buenos Aires, FCEyN
Buenos Aires, Argentina
Email: asanzo@fundacionsadosky.org.ar

Fernando Schapachnik
Partially supported by UBACyT 20020130200032BA.
Fundación Dr. Manuel Sadosky and
Universidad de Buenos Aires, FCEyN
Buenos Aires, Argentina
Email: fschapachnik@fundacionsadosky.org.ar

Pablo Factorovich
Fundación Dr. Manuel Sadosky
Buenos Aires, Argentina and
Universidad Nacional de Quilmes
Bernal, Argentina
Email: pfactorovich@fundacionsadosky.org.ar

Federico Sawady O'Connor
Universidad Nacional de Quilmes
Bernal, Argentina
Email: sawady.faso@gmail.com

***Abstract*—This article introduces Pilas Bloques, a scenario-based children learning platform, built to support Argentina's nation-wide Computer Science at school initiative. Besides a number of technical features like working offline and being light on resource consumption, it proposes a pedagogical strategy based on inquiry-based learning, decomposition, short and focused challenges and abstraction building.**

May, 2017

***Index Terms*—Computer Science at school; introductory programming teaching; learning platform; pedagogical strategy**

I. Introduction

If you had to support a nation-wide programming teaching initiative, would you develop yet another "programming for kids" platform? The only reasonable answer to that question is with another question: what are your requirements? This article describes why and how one of the teams in charge of supporting the national deployment of Computer Science (CS) education in Argentina, Program.AR, decided to build a tool to support a very specific set of requirements that, at the time, no other tool could fulfil.

The CS teaching effort had to be built on top of Conectar Igualdad[1] [1], the Argentine nationwide one-to-one computer program, with more than 5.5 million netbooks delivered to high-school students, and many local initiatives also making netbooks available at primary schools. Although these 1 to 4 GB netbooks can be considered widely available in Argentine schools, Internet connectivity cannot, meaning that an offline mode was mandatory. Also, the tool had to be in Spanish, and be available free of charge (being open-source was a plus). More importantly, it had to support the pedagogical methodology developed by the team over the years.

[1]www.conectarigualdad.gob.ar

With the clear intent of developing high-order thinking abilities (abstraction, critical-thinking, problem solving, etc. [2]) and helping students understand how the technological world we live in works, the Program.AR team developed a pedagogical strategy based on prior work by national universities [3] that is based on inquiry-based learning, decomposition, short and focused challenges, and where code quality is more important than mere goal-achiving code. As a difference with more well-known approaches, it pursues a *procedures first* approach, where task subdivision is presented before control structures and other programming constructs.

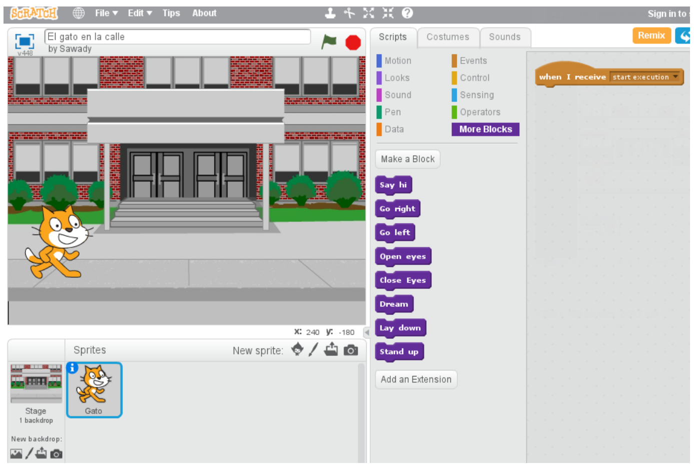


Fig. 1. Original scenario in Scratch. Shortcomings: ability to modify scenario and multiple distracting block categories and options.
Available at bit.ly/2ihqzCZ (in Spanish).

It was tested with 9-to-15 years old students in pilot courses through the country, and the compiled experience was used to develop teaching sequences that were combined in a teacher manual [4], using Scratch [5] as a supporting tool. As can be seen in the example challenge in Fig. 1, we offered a previously built scenario, with a very specific objective (in this case task division: *make the cat go to sleep and then wake up*), where the student had a limited set of domain-specific primitives (only *lay down*, *close eyes*, *open eyes*, *dream*, *stand up*, etc.)[2].

Using Scratch, however, presented several shortcomings for our approach. Although it fit perfectly fine for parts of the curricula where students are given the opportunity to pursue small-to-medium sized projects using procedures to achieve task division, Scratch's wide availability of characters, scenarios, and primitives could be distracting when the focus is put on particular concepts.

As a workaround, scenarios were built as incomplete projects hiding pieces of code deep down in the canvas (in Fig. 1 the purple blocks inner code is out of sight), yet students were still able to manipulate the scenario or focus away from the intended topics. In addition to that, there were features hard to implement, such as "resetting" the scenario state in each execution and providing automated feedback on the student's solution. This is mainly because Scratch has an open-ended activity philosophy.

All things considered, the decision was to build a specific tool to cope with the above mentioned requirements, and thus Pilas Bloques was born. Fig. 2 shows the same scenario of Fig. 1, but in the new tool.

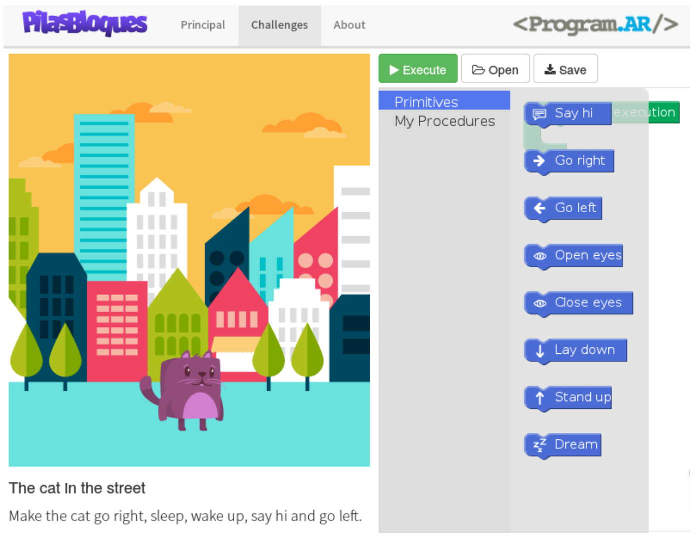


Fig. 2. Current scenario in Pilas Bloques. Predesigned animated scenario, simplified interface.
Available at bit.ly/2jINGqj (in Spanish).

This article describes the tool, the rationale for its design decisions and some early adopters statistics. Next section describes the pedagogical strategy used, emphasizing the concept of *procedures first*. Pilas Bloques is presented in more detail in Section III, while its usage data can be found in Section IV. Section V compares it with the most similar programming learning tools available. Finally, Section VI concludes the article with a brief outline of the tool's roadmap.

[2]All the images appearing in this article were specifically translated to English from the original tool's language (Spanish).

II. Pedagogical Strategy

Pilas Bloques' target audience is late-primary and early-secondary school students (aged 9 to 15), aiming mainly at public schools.

Some of the pillars of our pedagogical strategy are well-known criteria like inquiry based learning and native language use. In our case that means that whatever tool is used needs to be available in Argentine Spanish, so that language does not become a learning barrier. It also has to have a visually welcoming and children-suited design, which includes graphics and animations, and has to allow for programming through blocks [6], an increasingly popular strategy for avoiding syntax-related and conceptually shallow mistakes.

The rest of our pedagogical strategy is based on previous work by local universities [3] that has proven to be successful to engage students from impoverished backgrounds into programming in last year high school courses and first year university courses. These students usually lack abstraction capabilities, lack taste for math and logic, and usually have weak mathematical skills. The method hides mathematical concepts were it can, teaches concepts and abilities that are usually considered "failed" prerequisite (e.g., fluency in logic or self-motivation for study) and mainly focus on building abstraction capabilities.

Abstraction capabilities are pursued by placing emphasis on the representation of complex data as a combination of a small set of primitive alternatives and by requiring to program based on procedures and functions with proper names. As an example, consider a matrix of numbers with a read/write head that is placed on an cell at time and can only move in one of four directions at a time. Students are first taught how different numbers can be used to represent different objects. For instance, the matrix can be thought as a garden and the number 1 can encode a rose while 2 encodes a lily.

Next, procedures are taught, and the following assignment is given: "Plant a garden, which is a row of 4 bouquets, each one represented by a rose sorrounded by 4 lilies". This can be written as:

```
program {
  // First bouquet.
  put(2);
  move(down);
  put(1);
  move(left);
  put(2);
  move(right); move(right);
  put(2);
  move(left); move(down);
  put(2);
  move(down); move(down);
  // Second bouquet.
  put(2);
  ...
}
```

The teacher then explains that this type of solution is hardly descriptive and it lacks abstraction. An example of the type of code that is favored is:

```
program {
  PlantGarden();
}

procedure PlantGarden {
  PlantBouquet(); move(down); move(down);
  PlantBouquet(); move(down); move(down);
  PlantBouquet(); move(down); move(down);
  PlantBouquet();
}

procedure PlantBouquet {
  PlantLily();
  move(down);
  PlantRose();
  move(left);
  PlantLily();
  move(right); move(right);
  PlantLily();
  move(left); move(down);
  PlantLily();
}

procedure PlantRose {
  put(1);
}

procedure PlantLily {
  put(2);
}
```

Later on, an assignment to plant 200 bouquets can be posed, serving as a motivation to teach repetition control structures. Although the example shown in this article is textual, we make beginners start with block-based programming, even for our *procedures first* strategy.

For the concrete implementation of our inquiry-based strategy, we provide a set of fixed scenarios, each one with a defined goal. In order to complete each goal the student needs not only to apply previous knowledge, but also needs an ideally-small new concept that she does not still possess. Once the student reaches the problem, *tries to solve it with her current knowledge*, and formulates the question, then –and not before– the teacher introduces the concept.

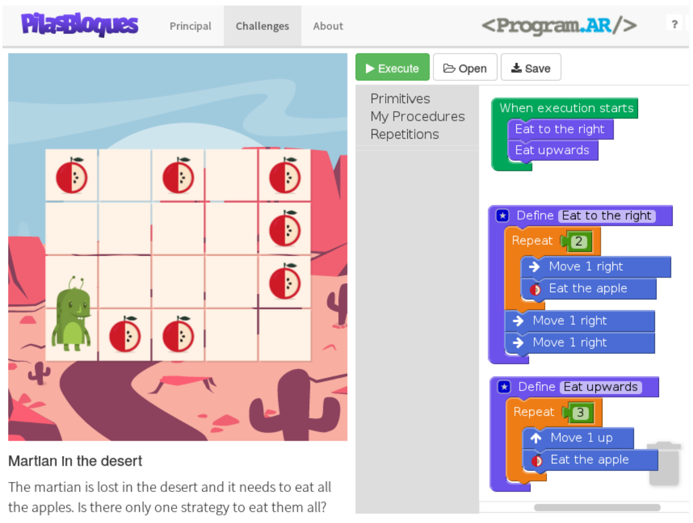
Our proposed learning path is presented as a collection of predefined scenarios with clear goals. Each one builds cumulative knowledge over the previous one, and each one represents a small, reachable step towards the understanding of the desired concepts. Open ended projects –like those traditionally used in "pure" Scratch– are often a very rich complement to these activities with predefined scenarios, yet in this work we focus on the latter.

For the initial scenarios, as it is customary with children programming pedagogical strategies, we also use children-tangible base programming elements: objects like apples, martians or cats, in opposition to traditional integers, strings, and structs. Accordingly, we use domain-specific primitives like *eat apple* or *is there an apple?*, in opposition to traditional string or integer operations.

Task-completion, i.e., the tool notifying the student that she finished a challenge, is key. In the case of Pilas Bloques, to stress that achieving the goal might not mean that the program is fully correct (i.e., the student could have used a sentence many times instead of using a repetition control structure), we output a carefully crafted message. It reads: "Goal achieved! However, your program might or might not be right. Discuss it with your classmates and teachers to know if you really found an idea that is useful for other tasks.". We plan to use program analysis techniques to provide more accurate feedback in the future.

III. Pilas Bloques

This section describes Pilas Bloques, which was build to support the pedagogical strategy described in Section II. The emphasis here is put on the technological aspects.



```
When execution starts
  Eat to the right
  Eat upwards

Define Eat to the right
  Repeat 2
    Move 1 right
    Eat the apple
  Move 1 right
  Move 1 right

Define Eat upwards
  Repeat 3
    Move 1 up
    Eat the apple
```

Martian in the desert

The martian is lost in the desert and it needs to eat all the apples. Is there only one strategy to eat them all?

Fig. 3. A Pilas Bloques activity. Left: the predefined animated scenario. Right: place to drag and drop blocks to put together a program (working area), where a program has been partially coded to illustrate the idea. Available at bit.ly/2jo1lzw (in Spanish).

In addition to the requirements of Section II, a number of technical (or rather, non-pedagogical) requirements needed to be fulfilled:

- The tool had to be available free of charge, and preferably, had to be open source.

- The tool had to be available both online and **offline**, because not every school in Argentina has stable Internet connectivity.
- The tool had to run in low-resource netbooks, because, as said before, that is the profile of the computer many students in Argentina had (and still have) access to.

Pilas Bloques was built using ember/javascript and blockly [6], and uses NodeJs/Electron to make it offline and available for Windows, Mac, and Linux. Pilas Bloques is available at http://pilasbloques.program.ar and the source code at http://github.com/Program-AR/pilas-bloques.

Fig. 3 shows a typical Pilas Bloques activity, with the left side panel containing the predefined animated scenario and the right side panel the canvas to drag and drop blocks to put together a program. Note that blocks are grouped by type and that primitives are domain-specific.

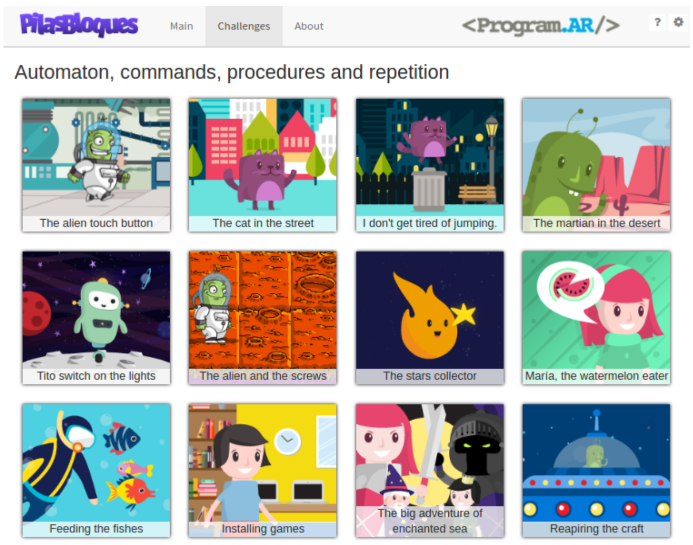

PilasBloques
Main
Challenges
About
<Program.AR/>
Automaton, commands, procedures and repetition
The alien touch button
The cat in the street
I don't get tired of jumping.
The martian in the desert
Tito switch on the lights
The alien and the screws
The stars collector
María, the watermelon eater
Feeding the fishes
Installing games
The big adventure of enchanted sea
Reapiring the craft

Fig. 4. The Pilas Bloques challenges screen let's the student choose one of the many predefined challenges.
Available at bit.ly/2jIQBz2 (in Spanish).

As can be seen in Fig. 4, in Pilas Bloques the student can choose from more than 40 predefined challenges that cover (1) basics, sentences, top down task division, repetitions and processing of fixed length linear data structures, (2) conditional alternatives, (3) conditional repetition, and processing of variable length linear data structures, (4) functions that sense the scene, and (5) parameters.

IV. Deployment

Pilas Bloques is the tool of choice for Program.AR CS teaching activities and teacher professional development. Some of its usage statistics are:

- **Availability**: Argentina's one-to-one netbook program, Conectar Igualdad, dual boots Windows/Linux. Pilas Bloques comes pre-installed as part of Huayra Linux[3], the Linux OS that ships with the netbooks since 2013. That means Pilas Bloques is pre-installed in approximately 3 million netbooks.
- **Teaching Material**: A teachers' manual for a one year-course using Pilas Bloques is available free of charge (in Spanish) at http://bit.ly/2l0nuEw.
- **Downloads**: The tool was downloaded 6000 times between March 2016 and April 2017 (63% corresponds to version 1.0.0 released on July 27^{th}, 2016).
- **Online Uses**: 200000 activities have been solved online in the same time span, by 26000 unique users.

During 2016 Pilas Bloques was used to train 900 teachers (http://bit.ly/2jssfGH) and 400 school principals and superintendents (http://bit.ly/2jsoctH) into teaching programming at school.

[3]http://huayra.conectarigualdad.gob.ar/

V. Related Work

There are many tools available to teach introductory programming to children and teenagers, and many reviews available (see, for instance [7] or [8]). For brevity we survey only the ones that are more popular and have similarities with our approach.

As was mentioned in Section I, **Scratch** [5] is a very powerful tool for open-ended, exploratory-type projects. It lets the user define a scenario and several characters to participate in a scene/game, using block-based code from a very large set of generic primitives. It also includes *procedure creation*. It has two modes: the "play" mode which hides the block code and offers the animation/game as a final product, and it has an "edit" mode, which was the one we used for our initial experiences. However, it does not directly support the pedagogical strategy we are interested in, which is based on features such as fixed-scenarios, ad-hoc domain-specific primitives and feedback on task completion (see Section II).

Alice [9] is another well-known tool, very similar in approach to Scratch: it also allows the building of complex scenarios and animations (although in this case those are 3D), and experiment using programming with blocks. Consequently its similarities and differences with Pilas Bloques are equivalent to those with Scratch. Besides the 3D scenarios, the main differences with other tools is that Alice is leaned towards teaching object oriented programming and allows a migration path from blocks to textual code.

Lightbot [10] is a teaching tool which had its first version as a browser flash game, and now it has developed a more user-friendly mobile version. It also uses graphical elements very similar to blocks to represent the primitives of the language, it has the advantage of defining a set of goal-oriented scenarios -levels- and every scenario has the same domain and specific primitives. Another similarity to Pilas Bloques is that it introduces *procedures* from a very early moment.

An important difference with other tools is that the working area where the primitives should be dragged on has a predefined amount of spaces for them. This has an impact on each activity's objective, which very quickly become a question of how to solve the problem *with a program that limits the number of instructions*. In fact, procedures are presented first as a way to *add extra instructions* to the program, and later as a way to solve *pattern repetition* problems (for which discovering the largest pattern helps reduce final instruction amount even more).

Lightbot has fixed names (f_1 and f_2) on the usable functions, so it does not emphasize the idea of using procedures or functions as way to structure task-division as we do.

Code.org [11] is a non-profit organization from the United States founded in 2013 whose objective is to divulge and promote education in CS. As a part of its campaign, it has developed a tool accessible via http://studio.code.org. This tool has several points in common with Pilas Bloques, and several differences. The main points in common are block-oriented programming, and scenario-based activities with primitives at a children acceptable level of abstraction.

The main difference is that *task division*, which is achieved by *procedure creation* is not introduced early in Code.org's curriculua. Nevertheless, Code.org recently introduced the idea of function creation and usage as a way of reusing code in it's last course, in a way that does not have a central role in their pedagogical strategy. Also, to the best of our knowledge, the tool does not have a downloadable standalone multi-OS version, something that Pilas Bloques features as a central requirement.

VI. Conclusions and Future Work

This article presented Pilas Bloques, a scenario-based children learning platform built to support Argentina's nationwide Computer Science at school initiative. It proposes a pedagogical strategy based on inquiry-based learning, decomposition, short and focused challenges and abstraction building. It also includes a number of features like availability of full-fledged offline version, being light on resource consumption, native-language interface, clean look and animated characters designed for children.

The tool comes with more than 40 predesigned scenarios which are suitable for a year-long introductory programming course, and more scenarios are being programmed. It has been well received online, with more 6000 downloads and more than 200,000 attempts to solve online activities over the last nine months.

Both Pilas Bloques and its pedagogical strategy need to be more thoroughly tested to assess its strong points and weak areas.

To that aim we plan to conduct an experimental study where same-age school students with no previous background on programming are randomly assigned to a Pilas Bloques group or to a Scratch (or Code.org curricula) group. After a one-semester course, we plan to measure mainly engagement, ability to properly use some coding constructs (e.g., control structures and variables) and abstraction building capacity. We define engagement as cognitive investment in learning and completing the task ([12]) and consequently it can be measured through several indicators like task completion, enthusiasm, participation, self reported interest and easiness and willingness to learn more among others (like in [13]), while coding construct domain can be evaluated via MCQs (like in [14]). Abstraction building capacity, on the other hand, might require to pose tasks that rank at a higher-level on Bloom's taxonomy [2], such as ranking peer solutions or criticizing a set of possible solutions to a given challenge.

It should be noted that such a comparison would follow a somehow "black box" approach, in the sense that it would say nothing about whether the benefits (or shortcomings) are responsibility of the respective tool or its pedagogical strategy. However, we argue that each tool was built with an intended way of using it, and such pair is what needs to be assessed in the first place.

Regarding the intended growth of Pilas Bloques, our current feedback on task completion is rather rudimentary. We plan to explore automatic code analysis techniques (e.g. [15]) to more accurately detect and explain the student when a solution reaches the goal yet it does not use some key concept (for instance, a repetition exercise that it solved by copying and pasting the instruction numerous times), or to more precisely differentiate, and properly hint, between solutions that do not reach the goal.

The other most salient aspect of our todo list is letting non professional programming teachers create their own activities by means of an automated tool, while clearly differentiating the ones that come pre-shipped from the contributed ones.

References

[1] S. L. Martínez, "Inclusión digital en la educación pública argentina. El programa conectar igualdad/digital inclusion in argentinean public education. the Conectar Igualdad program," *Revista Educación y Pedagogía*, vol. 24, no. 62, p. 205, 2012.

[2] L. W. Anderson, D. R. Krathwohl, and B. S. Bloom, *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon, 2001.

[3] P. E. Martínez López, *Las Bases Conceptuales de la Programación. Una nueva forma de aprender a programar*. E-Book, 2013, in Spanish. ISBN: 978-987-33-4081-9. Ebook URL: http://www.gobstones.org/download/bases-conceptuales-de-la-programacion/.

[4] P. M. Factorovich and F. A. Sawady O'Connor, *Actividades para aprender a Program.AR*. Fundación Sadosky, 2015, in Spanish. ISBN: 978-987-27-4161-7. Ebook URL: http://programar.gob.ar/descargas/manual-docente-descarga-web.pdf.

[5] 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," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.

[6] N. Fraser *et al.*, "Blockly: A visual programming editor," *URL: https://code.google.com/p/blockly*, 2013.

[7] S. Papadakis, M. Kalogiannakis, V. Orfanakis, and N. Zaranis, "Novice programming environments. scratch & app inventor: A first comparison," in *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments*, ser. IDEE '14. New York, NY, USA: ACM, 2014, pp. 1:1–1:7. [Online]. Available: http://doi.acm.org/10.1145/2643604.2643613

[8] I. Utting, S. Cooper, M. Kölling, J. Maloney, and M. Resnick, "Alice, Greenfoot, and Scratch – a discussion," *Transactions on Computing Education*, vol. 10, no. 4, pp. 17:1–17:11, Nov. 2010.

[9] W. Dann, S. Cooper, and D. Slater, "Alice 3.1 (abstract only)," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 757–757.

[10] L. A. Gouws, K. Bradshaw, and P. Wentworth, "Computational thinking in educational activities: an evaluation of the educational game light-bot," in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. ACM, 2013, pp. 10–15.

[11] F. Kalelioğlu, “A new way of teaching programming skills to k-12 students: Code. org,” *Computers in Human Behavior*, vol. 52, pp. 200–210, 2015.

[12] F. Newmann, *Student Engagement and Achievement in American Secondary Schools*. 1234 Amsterdam Avenue, New York: Teachers College Press, 1992.

[13] C. M. Lewis, “How programming environment shapes perception, learning and goals: Logo vs. scratch,” in *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’10. New York, NY, USA: ACM, 2010, pp. 346–350. [Online]. Available: http://doi.acm.org/10.1145/1734263.1734383

[14] J. L. Whalley, “Csed research instrument design: The localisation problem,” in *In S. Mann & N. Bridgeman (Eds.), Proceedings of The Nineteenth Annual NACCQ Conference*, 2006, pp. 307–312.

[15] S. Gulwani, I. Radiček, and F. Zuleger, “Feedback generation for performance problems in introductory programming assignments,” in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 41–51. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635912