Contextualized Spiral Learning of Computer Programming in Brazilian Vocational Secondary Education

Luis Gustavo J. Araujo
State University of
Feira de Santana
Feira de Santana, Bahia
Brazil 44036-900
luisaraujo.ifba@gmail.com

Roberto A. Bittencourt
State University of
Feira de Santana
Feira de Santana, Bahia
Brazil 44036-900
roberto@uefs.br

David M. B. Santos
State University of
Feira de Santana
Feira de Santana, Bahia
Brazil 44036-900
davidmbs@uefs.br

***Abstract*—This innovative practice full paper presents an experience report of an approach carried out with students of a vocational secondary program in Informatics, in an introductory course on Computer Programming. In Brazil, in addition to the undergraduate education, an important part of professional education is based on technical vocational secondary programs, geared towards faster training and practice. In information technology (IT) vocational programs, high levels of dropout and failure are frequent. Much of this failure happens in computer programming courses. Various factors contribute to the lack of student motivation and, thus, to the high professional deficit in the field of IT. This work presents an experience report of an approach used with students of a technical vocational IT program. The approach is based on activity contextualization, environments for novices and a spiral learning approach. The experience was split into three contextualized blocks: Figures, Games and Images. In each block, concepts are (re)introduced and deepened in a spiral way. Lessons learned suggest that: context change impacts student motivation, the approach is partially adequate to the audience, the Scratch tool contributes to learning programming concepts, and appropriate tools help transitioning between block-based and text-based programming languages.**

I. Introduction

In Brazil, the modality of technical vocational secondary education faces several problems related to both student permanence in the programs and graduate permanence in the field, especially in the field of information technology (IT) [1]. Instead of working as technicians, graduates of technical vocational programs usually continue their studies, frequently in different fields, leading to a deficit of IT professionals.

In addition to this issue, there are high failure and dropout rates in those programs [2], [3]. Among the various causes, we highlight those related to school and teaching staff, such as teaching approaches and teacher training issues [4]. In the area of computer programming, various approaches and teachers disregard student profile in course design. According to Guzdial, issues presented in computing courses are, for the most part, dissociated from students' lives, which leads them to find programming boring and excessively technical [5].

Curriculum designers and teachers need to consider student profile, adopting, in class, contexts or domains that interest them. Technical vocational students growingly interact in social network platforms, play complex video games and use image and audio applications. As Prensky asserts, such students were born in the midst of a technological whirlwind and interact daily with those media [6].

To account for student profile, several tools and languages have been developed, allowing the adoption of contexts closer to youngsters. Those contexts are present in tools to teach programming that meet the needs of novice programmers. Examples of these tools are Scratch, Logo, KTurtle, App Inventor, Lego Mindstorms, Jython Environment for Students (JES) and PPlay [5], [7], [8]. Associated with the development of these tools, several studies have been carried out using contexts such as stories, simulations, robotics, games [9], [10] and media [11], [12].

In addition to tools and contexts, several researchers try to apply and evaluate different pedagogical approaches, such as problem-based learning (PBL), flipped classroom, spiral learning and others. These approaches contribute to student motivation and learning and may impact on dropout and failure rates, while encouraging students to continue in their field of training.

Despite the variety of initiatives, little has been done on the use of media computation in Brazilian technical vocational education. In this scenario, we decided to explore the use of contexts, such as media and games, associated with spiral learning, an educational approach that aims to reduce cognitive overload and present concepts gradually [13].

This paper presents an experience report of a teaching and learning approach to computer programming performed in the eleventh grade of a technical vocational secondary program in Information Technology. Aiming at students' motivation and learning, the approach is based on activity contextualization, suitable environments for beginners and a spiral learning approach. In this approach, the same content is presented more than once, at increasingly complex levels, through contextual-

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

ized blocks that associate a context with a tool.

II. Background

Adapting the typical introductory programming course to a particular context or domain is a recurrent theme in the literature. Papers range from the creation of environments and languages adequate to novices (e.g., Logo, Blockly, Scratch, Alice and CodeCombat), to the use of contexts such as Figures, 2D/3D Games, Robotics and Media [14], [15]. The visual results generated by those tools and contexts minimize testing or debugging, reducing the cognitive overload [16], and facilitate learning.

The context of Figures, where learners create geometric figures, is usually implemented with the use of *turtle graphics*. The Turtle was idealized by Papert in the 1970s to introduce computing concepts to children. In his book "Twenty Things to Do with a Computer", he presents examples of using the computer through the physical version of the Turtle [17]. In the 1980s, Papert devised a digital version of Turtle that is still used nowadays, present in some languages and environments, e.g., Scratch and Python. Ranade presents a turtle-based library called "simpleCPP" to learn C++ through minimizing the focus on the language [18]. The library allows reducing the number of introduced topics at once, making it easy to learn repeat structures in the initial classes. In addition, the use of graphics supports teaching more complex concepts such as recursion, and promotes motivation for learning concepts such as inheritance, for instance. Several projects have used turtle graphics principles, such as Pencil Code [19] and The Hour of Code [20].

Games have been present in various societies and cultures throughout the times. Nowadays, youngsters interact with digital games through various devices such as computers, tablets and cell phones. Some studies explore the popularity and the playful aspect of games to teach programming. Some papers explore the use of games for teaching programming concepts. For instance, one paper introduces an approach named PlayIT, using the LightBot game to teach concepts such as select and repeat structures, functions and recursion [21]. Results suggest that students think that the game contributes to learn programming, and that their perceptions of programming as something interesting and their thoughts about finding the game fun are strongly related. Other papers use the game context by having students develop the games themselves. For example, one paper presents an approach using pyGame, introducing concepts incrementally to motivate students [22]. The approach initially presents a problem, its solution, incorporates new requirements and, finally, students implement it, showing a strong interest in game construction.

Media, such as images, videos and sounds, are also thoroughly present in young people's lives. A full-fledged approach to teach introductory computing through media was idealized by Guzdial, initially for Liberal Arts majors at Georgia Tech (CS1 non-majors) [5], [23]. However, experiences with this context are not restricted to non-majors. Simon presents an approach to teach programming to Computer Science majors, showing that 83% of media computing tasks requires the use of two or more programming concepts, thus, being more challenging [24]. In the traditional approach, only 37% of the tasks met this requirement, leading authors to conclude that Computer Science students in the media computation course learned similarly to students from a traditional approach.

In Brazil, games are frequently used to teach programming. Rebouças et. al. present game development as a context in a programming course. The authors note that the Python language, combined with the pyGame library, offers an effective combination for using games to teach programming [9].

The use of tools and contexts is also the focus of several papers on teaching programming. As Bordini et al. point out in a survey of Brazilian K-12 computing education, several papers use games, robotics, automation and simulation, among other contexts [7]. However, as that review shows, the use of media, such as sounds and images, is not thoroughly explored, unlike the international scenario.

We have previously worked with image and game contexts to teach programming in technical vocational education [11]. Our previous study also uses a spiral approach, divided into two contextualized blocks, using Images with the JES environment and Games with PPlay, both with the Python language. Results point out the adequacy of Python as first language, and the importance of contextualized activities and challenges.

Media are also used in other educational modalities in Brazil, such as a CS1 course to Civil Engineering majors in higher education [12]. Another study in middle school uses two contextualized blocks, i.e., Figures with the Turtle library and Images with the JES environment, both with the Python language [25]. Both studies aim to investigate student motivation and learning.

In addition to tools and contexts used in a teaching approach, one needs to reflect on the learning theories that support the approach. Among various learning theories, we stand for the use of spiral learning, because it uses a gradual introduction of content to students, reducing cognitive overload. Although Bruner's original concept of spiral has a focus on the curriculum, this premise can be adapted to isolated courses as well, presenting contents repeatedly and with gradual increase in complexity [26], [27].

III. Methodology

This section describes the used language and tools, participants, course planning and the procedures for data collection and analysis.

A. Language and Tools

Python is a language suited to the needs of novice programmers as it is well-suited to education, popular and used in industry. According to Grandell et al. (2006), Python is a high level language, originally created to facilitate learning, and initially suggested as an alternative to teaching programming by its creator, Guido Van Rossum [28]. Python has many features of a language suitable for teaching programming,

such as clear and reduced syntax, dynamic types, expressive semantics, immediate feedback, structure reinforcement, and relevant open source projects.

Scratch is an environment for learning programming developed by the MIT Media Lab. Scratch uses a block-based programming language that allows novice programmers to build and apply programming logic concepts without knowing how to code. This feature allows Scratch to be tinkerable. In it, users organize blocks in different ways and see how they behave by learning through practice. In addition, it allows users to focus on problem understanding, rather than struggling to get a "compilable" program [29].

JES – Jython Environment for Students is an environment for teaching programming through media manipulation developed by Georgia Tech [5]. JES has modules for manipulating images, sounds, videos, websites and a Turtle module, all in Python. Its functions are simple, allowing media manipulation with a few lines of code. That allows programming novices to develop relevant activities, e.g., effects in images, creation of figures, creation and sound effects, edition of videos and website creation, with little effort.

B. Participants

This study was carried out in a technical vocational high school in the city of Feira de Santana, Bahia, in the eleventh grade class of an Information Technology program. The approach was deployed within the course of Programming Language 1 (LP1), which proposes to teach the fundamentals of programming and algorithms. Participants were 9 students: 6 girls and 3 boys, with a mean age of 15 years (standard deviation of 0.53), and with no prior programming experience.

It is important to highlight that all students participated in the study. Nonetheless, the number of students is still small from the start, pointing to a common problem in Brazilian vocational courses: high failure and dropout rates.

The researcher acted in this study as both teacher and observer; an undergraduate assistant also supported students in class; and a third researcher acted as observer in class. The original teacher watched the classes, performing very few interventions.

C. Course Planning

The proposed approach aims to teach programming concepts, seeking to facilitate learning and increase student motivation. Therefore, we used tools and languages appropriate to these students through a spiral learning approach [13]. We planned the use of the following tools: Scratch, for creation of geometric figures and drawings and game development; and JES, for editing images using the Python language. We split the course into three blocks. Table I gives an overview of the approach.

The three blocks that make up this study repeatedly present concepts of programming and algorithms, at ever deeper and more complex levels. For example, a simple select structure (*if*) is learned first; then, a chained select structure (*if – else, if – elif – else*); later, a nested select structure and more complex forms combining relational and logical operators.

In Block I, we presented programming concepts such as select and repeat structures, functions and relational operators, using Scratch and the context of geometric figures. Table II presents the planning for Block I.

During Block II, we kept the Scratch tool, changed the context to games, presented new concepts such as variables and parameters, and reviewed and deepened concepts previously seen. Some activities were based on previous work from our group [30], [31]. Planning for Block II is in Table III.

In Block III, we used the context of image manipulation, started using the Python language and the JES tool, reviewed all the concepts seen from the previous blocks, and discussed issues of language syntax. Planning for Block III is in Table IV. The materials used in the approach can be found in the support website[1].

D. Data Collection and Analysis

In parallel to this intervention, we carried out an exploratory research. Quantitative data collection procedures used were: i) pre-intervention survey; ii) post-block surveys; iii) post-intervention survey. Thus, at the end of each block, we surveyed students with the purpose of measuring the changes of perceptions of programming concepts and contexts used, as well as their motivation. After all the blocks, we used a post-intervention questionnaire that included questions to assess motivation during the course. Quantitative results were analyzed through descriptive and inferential statistics. We also used data from the students' written exams. All participants answered the surveys.

Qualitative data collection procedures were: i) semi-structured interviews with students; ii) semi-structured interview with the teacher; iii) reflective journals and iv) in-class observations. Qualitative analysis was based on content analysis. We started with open coding, followed by axial coding, and we finally wrote analytical memos to facilitate the interpretation of results [32].

Qualitative data were separately analyzed by block. Thus, we could perform an analysis to identify particular elements of each block, such as tools and contexts. In block I, open coding revealed 42 codes. In block II, it revealed 33 codes. In block III, initial coding revealed 41 codes. All codes for each block were separately grouped into four categories in the axial coding: Approach, Learning, Tool and Motivation.

IV. Our Experience

This session presents our experience, through the description of the activities performed in the three blocks presented as a temporal narrative.

A. Block I – Scratch and Figures

In the first block, we presented notions of programming and the Scratch tool, examples of use and its GUI. We introduced students to the basic concepts of programming

[1] http://luisaraujo.github.io/programacao_com_midias/

Table I
SYNTHESIS OF COURSE PLANNING – LP1

Place	Centro Territorial de Educação Profissional do Portal do Sertão, Feira de Santana, Bahia, Brazil
Participants	9 Students
Blocks	Figures (Drawing geometric figures)
	Games (Creating games)
	Images (Manipulating images)
Contents	Algorithms and programming, data types, Input and Output, Functions, Parameters, Select and Repeat Structures, Variables, Constants, Arithmetic, Logical and Relational Operators
Tools	Scratch, JES and the Python language
Duration	Five months (three hours per week)
Workload	54 hours
Period	April 20 to September 28, 2017

Table II
COURSE PLANNING – BLOCK I – LP1

Dia	Goals	Contents	Activities
01	Review knowledge about geometric figures and apply it in Scratch	Basic commands of the Scratch pen (Pen up/Pen down, Move and Turn)	Draw lines and squares using Scratch commands
02	Deepen knowledge about angles and apply it	Basic commands of the Scratch pen (Pen up/Pen down, Move and Turn)	Draw triangles and stars
03	Identify the need and apply loops to draw figures	Decompose sequences in simple and nested loops	Create squares with simple and nested loops
04	Use loops to create more complex figures	Scratch commands (Change color effect by, Change x/y by) and loops	Draw figures using squares
05	Apply select and repear structures to create figures	Select structure (if and if–else), Relational operators and variables	Create a colored circle with select structures
06	Apply knowledge acquired in the block	Apply knowledge acquired in the block	Draw an image with various figures
07	Assess practical knowledge learned during the block	All presented concepts and contents	Answer written exam

Table III
COURSE PLANNING – BLOCK II – LP1

Day	Goals	Contents	Activities
01	Know the concepts and development process of games.	Concepts about Game Design and Game Development	Understand how games work and the development process
02	Apply movement commands in game mechanics	Variables, Specific Scratch commands	Create the Pong game using the learned concepts
03	Apply knowledge about select structures in game mechanics	Select structures, Relational Operators and already presented Scratch commands	Create the Pong game using the learned concepts
04	Apply programming knowledge in game development	All presented concepts and contents	Create the Pong game using the learned concepts
05	Understand new Scratch commands and the mechanics of Space Invaders	Apply knowledge about select structures and relational operators in game mechanics	Create the Space Invaders game using the learned concepts
06	Apply knowledge about select structures in game mechanics	Select structures and Scratch commands (Motion and Looks)	Create the Space Invaders game using the learned concepts
07	Apply knowledge about repeat structures in game development	Select and repeat structures, Scratch commands	Create the Space Invaders game using the learned concepts

through creation of geometric figures. We used some figure variations to compose more entertaining drawings.

The first day, students created geometric figures such as lines and squares, using functions of the Scratch pen. Students did not show difficulties at this time, creating figures with ease.

The second day, from concepts of the previous class, students created figures such as triangles and stars. In this activity, the use of angles was a barrier for some. After creating a triangle, by trial and error, some students still had difficulties to create the star, because they could not adapt the algorithm to create stars from triangles. However, with the help of the teacher and assistant, students completed these activities.

The third day, students learned repeat structures and applied those concepts to previous exercises to build squares. Students reported that the use of loops is simpler and facilitates square creation. For them, the number of Scratch blocks used makes the activity more complex. However, it is the introduction of the new concept that is the most complex aspect of this lesson.

The fourth day, students applied the concept of loops in exercises to create square variations: overlapping, changing

Table IV
COURSE PLANNING – BLOCK III – LP1

Day	Goals	Contents	Activities
01	Build knowledge about digital images, Objecss, Classes and apply it inside JES	RGB model, Digital Images, Classes, Objects, JES functions (get/setRed, get/setGreen, get/setBlue, pickAFile, getPixel, setColor), JES Classes (Color, Picture, FileChooser)	Create geometric figures as pixels (Square, Triangle and Circle).
02	Build knowledge about arrays and Repeat Structures	Code decomposition, Arrays, for loops, Functions in, range and JES write function	Create geometric figures (Square, Triangle and Circle) and gradient effect with loops
03	Apply the concepts of arrays and Repeat structures to image manipulation	Arrays, for loops, Arithmetic operators, Previously presented JES functions and getPixels.	Create hue effects (change hue), Negative and greyscale
04	Build knowledge about Select structures and apply it to image manipulation	Select structures (if), Relations operators, Previously presented JES functions and getWidth, getHeight	Create effect in half an image and Black and white effect

their color, and generating playful images. Activities of student modifying drawings and the variety of figures were more relevant as they exercised creativity.

The fifth day, students learned the concept of select structures and applied it to create circles, by using loops. The teacher mediated the creation of simple circles by a theoretical presentation, where students should find the right measures, i.e., steps and angle, to create a circle.

This activity is more complex for associating circle replication with loops and conditionals, regardless of being similar to the previous lesson. However, creativity was exercised, since students were able to modify colors, pen thickness, and create new figures, applying variations with conditionals.

The sixth day, students created a figure composed of several geometric figures by applying previous concepts: the figure of a house presented by the teacher. As in the previous classes, they modified the activity according to their tastes.

B. Block II – Scratch and Games

In the second block, we kept the Scratch tool, but we changed the context. In this block, we aimed to create classic games such as Pong and Space Invaders, deepening concepts previously learned, and adding new concepts.

The first day, students learned about the process of game development and saw examples with Scratch. Through the presentation of the examples, they were encouraged to create games. Before starting, the teacher showed gameplay videos.

The second, third and fourth days, students were reintroduced to concepts such as variables, select and repeat structures, using more complex relational operators to create the *Pong* game. As students already knew Scratch, activities focused on programming logic. However, some tried to replicate knowledge from the previous block by using pen commands to move sprites. In general, students remained engaged in the activity and completed the game with the help of the assistant and classmates. In this activity, there was less exercise of creativity, since students had to follow the mechanics of the game.

The fifth, sixth and seventh days, students created the *Space Invaders* game, applying the concepts already seen on game mechanics. Because of the greater complexity of this game compared to the former, students needed to reuse concepts in more complex ways, since Space Invaders has richer mechanics and more objects.

C. Block III – Python, JES and Images

In the third contextualized block, we presented the JES environment and a text-based programming language (Python). The aim was to apply programming concepts to image manipulation. We presented the main items of the JES GUI and some of the elementary functions (e.g., to get an image and display it). This initial moment was quite expository, with students repeating the commands used by the teacher.

Although the image concept is close to students' lives, we needed to present concepts of digital image representation. Thus, we addressed the RGB standard and images as arrays of pixels. Students showed ease with both these elements and elementary functions (e.g., get and set pixels and colors).

As a first activity, students modified some pixels of an image, by drawing geometric figures such as squares, triangles and circles. Even though it was their first experience with a text-based language, students had no difficulties.

The second day, students reused the code from the first class to decompose it into functions and to add repeat structures, as well as to create gradient shapes using mathematical formulas. Although most students found math difficult, they liked to develop these activities.

The third and fourth days, students went from creating images on an initially white screen to manipulating previously constructed images. During this class, they created effects such as greyscale, black and white, and negative. Although we guided these activities more than those of previous classes, we noticed that students understood well the concepts used.

D. Summary

Scratch helps to understand some programming concepts that are used by students with little effort. The activities also contributes to their use of both functions, select and repeat structures. However, students still have some difficulties to understand concepts such as variables and relational operators. On the other hand, Python allows better understanding of the

latter. Despite the transition from Scratch to Python in Block III, there are no major conceptual difficulties when compared to the previous blocks.

Spiral learning allows to deepen some concepts with very few cognitive overload, contributing to the ease of learning. It also contributes to student motivation.

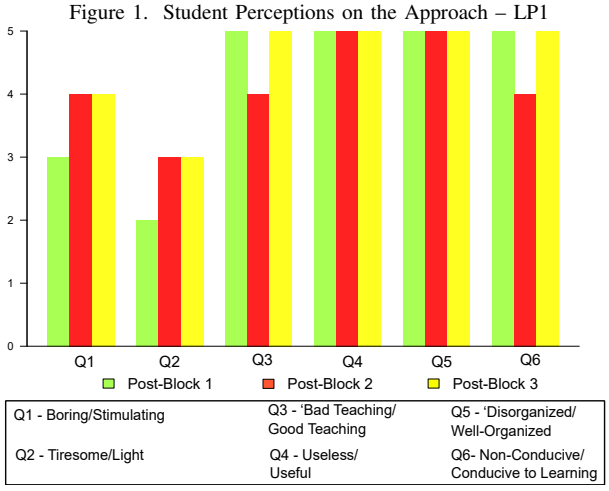
V. Results

This section presents students' perceptions of our approach, the evaluation of motivation through the CIS instrument, and the main themes that emerged with the qualitative data.

A. Students' Perceptions

During the course, we tried to assess aspects related to the approach through a dichotomous scale between 1 and 5. Figure 1 shows the average values in this scale for each aspect in each of the blocks. We notice that most aspects have positive results, partially in the first block, and predominantly in the last two blocks. Only the 'Tiresome vs. Light' aspect had results below the average of three for Block I. This aspect improved as well as the 'Boring vs. Stimulating' aspect in the transition between Blocks I to II, maintaining the results in block III. Block II had two negative changes in assessment – in terms of 'Bad Teaching vs. Good Teaching' and 'Non-Conducive vs. Conducive to Learning' – compared to Block I. Those aspects improved in Block III, having maximum ratings together with 'Useless vs. Useful' and 'Disorganized vs. Well-Organized'.

Figure 1. Student Perceptions on the Approach – LP1



The worst rated aspect was 'Tiresome vs. Light'. This indicator can be explained by the volume of activities planned during the blocks, since all the classes had practical activities to be carried out. However, aspects of teaching, organization, conducing to learning, and usefulness show how appropriate the approach is to students' profiles.

B. Post-Intervention Survey (CIS)

At the end of the course, participants answered the Course Interest Survey (CIS)[2] [33]. CIS is an instrument based on the Attention, Relevance, Confidence and Satisfaction (ARCS) model that aims to measure student motivation. It was designed to measure students' reactions to instructor-led instruction in classroom education.

Results on *Attention* show that students agree that the instructor does unusual or surprising things that are interesting (44%), and that the instructor creates suspense when developing content (33%). On the other hand, 44% of the students agree that classes have very little that captures their attention (44%) and that curiosity is not stimulated by questions or problems in the course (77%).

Results on *Relevance* show that only 11% disagree that the instructor makes the subject look interesting. Students agree that they actively participate in class (33%). Only 22% of students agree that they will not benefit from this course. Contrarily, students agree that personal benefits are not clear (88%), and 66% do not see how course content relates to something they already know.

Results on *Confidence* show that students agree that success in the course depends on them (44%), that content is very difficult (55%). 33% agree that they fell confident during the course. No student agrees that it is difficult to predict their grades in the activities. Students disagree that it takes luck to get good grades (44%). However, students did not find the level of the course appropriate (66%).

Results on *Satisfaction* show that no student disagrees that the amount of work is appropriate. 55% of the students disagree that they need to work hard to suceed in the course. 33% of the students agree that the course gives a lot of satisfaction, and that their grades are fair compared to other students. However, 66% do not enjoy working for this course.

Analyzing CIS results, we notice that although the instructor creates suspense and does unexpected things, the course does not catch the attention of the majority of the students, nor does it stimulate their curiosity. Although the instructor makes the subject seem important and the students understand the benefits of the course, they do not realize the personal benefits and the relationship with things they already know.

Finally, students understand that success in the course depends on them, and that luck is not a determining factor. Although they find the content difficult, they feel confident and find it easy to predict their performance. Students find, despite the approach evaluation pointing to the contrary, that the amount of activities is appropriate. This suggests that it may not be the amount of activities that causes the tiresome aspect, but their level. However, grades are fair for them, and the course is satisfactory.

C. Qualitative Data

Besides the CIS data, we also collected qualitative data from interviews and in-class observations. Here we present the most relevant themes that emerged from our analysis.

Mediation. We noticed that mediation is an important element in the teaching process. During the activities in the three blocks, students reported on teacher and assistant mediation. Some students, upon finding a barrier, asked the teacher for

[2]Survey results in https://sites.google.com/site/fie2018cs1vocational/

help:: "*It's stressful! You want to give up and leave it all. Then I would ask for help from the teachers or from a classmate.*" (S1)[3]. "*I got desperate, called the teacher, then he explained it, and then I did it. I also tried to look at the classmate near me.*" (S2). "A student (S4) points the issue of teacher mediation when asked on what could make the course harder". Thus, the mediation process adopted in the experience seems imperative in this teaching and learning process.

Contexts. A key element in the approach is the contexts used in the experience. They aimed to turn classroom tasks closer to students' everyday activities, making them more meaningful. According to two students, contexts help in learning: "*It helped us because we were using our reasoning to do something different. It's better than doing calculations ... no, we burned our brains to try to make figures, try to make some shapes.*" (S9). "*It motivated me, because it was more fun than calculating... we made the images, the commands. I think it was more fun. So it motivated me. In the beginning, we used to do activities on computations, so it's better.*" (S6). Another student highlights the visual element of contexts. "*When you visualize it, the thing gets easier for you to perceive. I think the human being is more for the visual. In the case of pure calculations, you get the results, but it's more complicated. Math is weird!*" (S5).

For students, some contexts are more complex than others, as in the case of games. However, there is a sense of accomplishment, both for the difficulty and for the interactive result. One student asserts: "*I think it's much better. It's hard, I know, but only in comparison. At first I found it easy that it was just an algorithm, but after that, I moved on to that part, I found it harder. But it's kind of... you know... to make the game, when it's over, it's something else. When you see it running, it's good too.*" (S4). "*The change made it harder, because it involves ... I mean there was more code. But it motivated me because it was learning something new.*" (S7).

Two students point to the difference between the approach used and a traditional approach. "*It was cool, because, in addition to learning how to edit images, I was not doing that basic stuff every teacher goes through, it was a different stuff.*" (S9); "*Editing images is better, because it gives you more incentive, and the traditional way tires you up. At first it's good, but then it gets tiring, because it's only numbers, there's no visuals.*" (S8).

Thus, we noticed that games are more challenging because they are more complex. Nevertheless, they can motivate students. Contexts, i.e., figures, games, and images, provide visual feedback that is important from the point of view of learning and motivation. We also perceived that students prefer more contextualized approaches than traditional approaches.

Student relationships. Collaboration among students was always present. Students helped each other during the activities, creating a cooperation environment. Some reports show this: "*I looked at someone else's code, and others came to look at my code.*" (S9); "*It was good. I asked for help from S4, S1 and S3 ... whoever was around me. I would get up and look at their code.*" (S5). "*When we knew it, we would go there and help our classmates. One helping the other. S8 helped me, he showed me how to do it... When I was going to help someone, I didn't give her the answer, just what she had to do, what she needed to do.*" (S4). We noticed that asking for help from the classmate is the first option. Only when stuck, students request help from the teacher. During the course, we did not prevent students from helping each other, since the interaction between them also enabled knowledge construction.

Tools. About Scratch and JES, we realized that they were generally adequate for students. Scratch interface helps students: "*The way you tinker with the blocks, to make commands fit Lego style, and it has colors. The colors help me to associate!*" (S1). '*Because when I think about programming, I think like this: a lot of text, and you trying to do the stuff there... don't know what. Here the design is different, there are commands on the side, there are some little drawings that you can add, it's really cool.*" (S9). As well as its visual feedback: "*Scratch helped me ... like that, when you do it on paper, you only know the theory, but with Scratch, you see it happening, understand? Like that stuff of repeating, on paper we just put it there, but in Scratch it goes... and you see, if you're wrong, you'll know.*" (S6).

For some students, JES initially seemed easy: "*Despite being the first contact of the students, they are motivated with JES, they said it has an easy to use interface.*" (O1)[4]. And its use after Scratch is motivating: "*When showing the print with operators, E6 said: 'I see that it's going to be better than the other'*" (O1); "*Students are celebrating the fact of using JES, even with results on the console.*" (O2). However, some students negatively assessed the tool: "*I would put a better graphical interface on JES, with the commands on the side ... and explain what they did.*" (E8). It is noticeable that the student has the Scratch commands as reference, the way they are displayed for use. "*It was a surprise, because in Scratch I saw those colorful things there, the blocks, and in JES, I had to type, do the whole thing.*" (E9).

Motivation. We highlight in the following the four categories present in the ARCS model as they showed in the qualitative data.

About **attention**, students showed interest in several aspects of the approach, both the content, the repeat command and the possibilities that this structure offers, since students have started the activities by repeating codes to generate figures. We perceived, in the games block, that context is the most evident factor emerging from the qualitative data. The features of games catch students' attention and arouse curiosity. Changing context aroused an initial excitement in the students. An important element for attention in the context of images is the possibility of varying examples and activities. Since the image effects require just a few lines of code, enough to complete in some minutes, it is possible to present new things in each class. This was pointed out by a student.

[3]Students are identified with the letter S and a number for each student.

[4]Observations are identified with the letter O and a number for each class.

Regarding **relevance**, we noticed that the activities and the context are relevant, since they allow students to change the proposed activities. Students were surprised by the fact that they could edit images, relating it to app use, something common in their lives. The visual element and the possibility of applying the knowledge learned outside the classroom is an issue raised by students. "*Because it's more fun for me with something that gives results later. We'll be able to use it!*" (S8). Another student, similarly, highlights the possibility of applying the knowledge in internships or at work after the course: "*I think that when we finish the course, if an internship or something similar appears, we will have an easy time.*" (E2).

We noticed that student **confidence** may present itself in class in several ways. Various proposed activities had to be completed at the end of the class. Thus, the students who could not finish them, remained in classroom, confident that they would finish the activity soon, since they would have to leave for the break. Another issue is related to student collaboration, when they move around in classroom or are called by classmates to help them. That shows a degree of confidence by some students.

Something that generates **satisfaction** for students is to finish the activities successfully. According to one student, the motivation is "*because of the final results, I felt more motivated to see the results. I put an effort to do it!*" (E1). That generates a sense of confidence: "*S1 finishes the movement of both the ball and the bar and calls: 'Done!'; the teacher sees it and asks: 'What's the feeling?'; He says: 'Being a good programmer!'*" (O2). Another important issue is feedback: "*I think what motivates me to learn more is that when I show the code to the teacher, he says: 'Congratulations!'*". In addition, satisfaction in completing a difficult activity and satisfaction of obtaining visual results both stand out. For one student, the challenge is a motivating element.

VI. Learned Lessons

The approach is partially appropriate to the public. Some students find that the approach is organized and helpful, provides good teaching and facilitates learning, which is on the positive side of the assessment. On the other hand, some students have assessed the course negatively: they consider it tiresome, they think it does not stimulate curiosity, and its benefits to students' lives are not clear.

Contextualized activities are important for student motivation, and influence the learning of the contexts themselves, since each context has a group of most used concepts. Some contexts have higher levels of complexity, such as games, and that should be taken into account when deciding for their use. Teachers should analyze activities and use them to deepen concepts at particular times within their courses.

Introduction of new contexts, contents, environments and languages is important, because it avoids boredom, facilitates learning new concepts and their use in different scenarios, in addition to deepening concept understanding. Thus, besides the contextualized activities, it is important to find a good mix of them, varying environments and languages, if needed.

Scratch is suitable for learning introductory programming. Students have built complex figures and games with no prior experience in programming. Thus, with little instruction, students are able to manipulate Scratch, exercise their creativity during the execution of activities, and master programming concepts.

The transition between Scratch and Python is effective. We noticed that the transition to a text-based language, after two blocks using Scratch to create figures and games, proved effective. Students did not show initial astonishment, nor did they have their confidence shaken when they had to use a new language. However, the use of Scratch first causes some side effects when we move to a tool with a less rich user interface, like JES.

Our teenage students rarely study at home. We noticed that most students participating in the experience did not practice programming out of class, nor did they study the concepts presented in class. Although some mentioned that they were researching at home, this was not confirmed in class. Hence, one needs to create strategies for building knowledge in class. It is important to emphasize that this issue negatively influences the idea of making the course lighter, with fewer activities, since it is usually the only time they have to practice programming.

VII. Conclusions

This work presented an experience report of a teaching and learning approach in an introductory course on Programming Languages with students of the eleventh grade of a technical vocational program in Information Technology. Our approach combines the use of environments (Scratch and JES) and language (Python) suitable for programming novices, contexts relevant to these students (figures, games and images) and gradual content introduction with increasing complexity, based on spiral learning.

About the structure of our learning blocks, we learned that the repeated use of tools allows greater exploration by students. However, contexts with more complex activities may reduce the exercise of student creativity, as it was the case with games. Nonetheless, the spiral approach associated with several contexts supports deeper understanding of basic programming concepts.

The present study suggests that the block-based Scratch language and the text-based Python language are suitable for novice students, being learned with very few difficulties, in contexts such as geometric figures and image manipulation. We also noticed that the transition between Scratch and Python minimizes the difficulties of introducing a text-based programming language.

Future work includes a thorough analysis of the approach presented here to better understand, through both qualitative and quantitative data, how details of the approach impact student motivation and learning in technical vocational education.

REFERENCES

[1] Observatório Softex, *Formação e Capacitação para a Indústria Brasileira de Software e Serviços de TI*. Softex, 2010.

[2] M. R. Machado, R. Moreira, and R. Priscila, "Educação Profissional no Brasil, Evasão Escolar e Transição para o Mundo do Trabalho," in *I Colóquio Internacional sobre Educação Profissional e Evasão Escolar*, 2006.

[3] A. C. Cravo, "Análise das Causas da Evasão Escolar do Curso Técnico de Informática em uma Faculdade de Tecnologia de Florianópolis," *Revista Gestão Universitária na América Latina – GUAL*, vol. 5, no. 2, pp. 238–250, 2012.

[4] C. R. da Silva, B. R. Pimentel, and K. R. Finardi, "Refletindo sobre a evasão em um curso técnico do Pronatec," *Revista de Ensino, Educação e Ciências Humanas*, vol. 15, no. 3, 2015.

[5] M. Guzdial, "A Media Computation Course for Non-Majors," *ACM SIGCSE Bulletin*, vol. 35, no. 3, pp. 104–108, 2003.

[6] M. Prensky, "Digital Natives, Digital Immigrants Part 1," *On the Horizon*, vol. 9, no. 5, pp. 1–6, 2001.

[7] A. Bordini, C. M. O. Avila, Y. Weisshahn, M. M. da Cunha, S. A. C. Cavalheiro, L. Foss, M. S. Aguiar, and R. H. S. Reiser, "Computação na Educação Básica no Brasil: O Estado da Arte," *Revista de Informática Teórica e Aplicada*, vol. 23, no. 2, pp. 210–238, 2016.

[8] IC-UFF, "PPLay: Aprenda a Programar Desenvolvendo Jogos," 2017, http://www2.ic.uff.br/pplay/. Acesso em 18 ago. 2017. [Online]. Available: http://www2.ic.uff.br/pplay/

[9] A. D. D. S. Rebouças, D. L. Marques, L. F. S. Costa, and M. A. de Azevedo Silva, "Aprendendo a Ensinar Programação Combinando Jogos e Python," in *Simpósio Brasileiro de Informática na Educação*, 2010.

[10] D. L. Marques, L. F. S. Costa, M. A. de Azevedo Silva, and A. D. D. S. Rebouças, "Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução à Programação Utilizando Jogos e Python," in *Anais do Workshop de Informática na Escola*, 2011, pp. 1138–1147.

[11] L. G. J. Araujo, R. A. Bittencourt, and D. M. B. Santos, "Ensino de programação na educação básica através da manipulação de mídias," in *COBENGE 2017 - XLV Congresso Brasileiro de Educação em Engenharia*, Joinville, 2017.

[12] B. L. Santana, J. S. L. Figuerêdo, and R. A. Bittencourt, "Motivação de Estudantes Non-Majors em uma Disciplina de Programação," in *WEI 2017 – XXV Workshop sobre Educação em Computação*, 2017.

[13] J. S. Bruner, *Toward a Theory of Instruction*. Harvard University Press, 1966, vol. 59.

[14] A. Tartaro and H. Cottingham, "A problem-based, survey introduction to computer science for majors and non-majors," *Journal of Computing Sciences in Colleges*, vol. 30, no. 2, pp. 164–170, 2014.

[15] D. Saito, A. Sasaki, H. Washizaki, Y. Fukazawa, and Y. Muto, "Program learning for beginners: Survey and taxonomy of programming learning tools," in *Engineering Education (ICEED), 2017 IEEE 9th International Conference on*. IEEE, 2017, pp. 137–142.

[16] S. Y. Lye and J. H. L. Koh, "Review on teaching and learning of computational thinking through programming: What is next for k-12?" *Computers in Human Behavior*, vol. 41, pp. 51–61, 2014.

[17] S. Papert and C. Solomon, *Twenty things to do with a computer*. Massachusetts: June, 1971.

[18] A. G. Ranade, "Introductory programming: Let us cut through the clutter!" in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2016, pp. 278–283.

[19] D. Bau, D. A. Bau, M. Dawson, and C. Pickens, "Pencil code: block code for a text world," in *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 2015, pp. 445–448.

[20] Code.org, "Hour of code," 2018. [Online]. Available: https://hourofcode.com/br

[21] A. Mathrani, S. Christian, and A. Ponder-Sutton, "Playit: Game based learning approach for teaching programming concepts." *Educational Technology & Society*, vol. 19, no. 2, pp. 5–17, 2016.

[22] C. Yang and J. Yu, "Using incremental worked examples for teaching Python and Game programming," in *International Conference on (CSEIT 2011)*, 2011.

[23] M. Guzdial, "Exploring Hypotheses about Media Computation," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ACM, 2013, pp. 19–26.

[24] B. Simon, P. Kinnunen, L. Porter, and D. Zazkis, "Experience Report: CS1 for Majors with Media Computation," in *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*. ACM, 2010, pp. 214–218.

[25] L. G. J. Araujo, R. A. Bittencourt, and D. M. Santos, "An analysis of a media-based approach to teach programming to middle school students," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: ACM, 2018, pp. 1005–1010.

[26] J. S. Bruner, *Toward a theory of instruction*. Harvard University Press, 1966, vol. 59.

[27] ——, *The process of education*. Harvard University Press, 2009.

[28] L. Grandell, M. Peltomäki, R.-J. Back, and T. Salakoski, "Why Complicate Things?: Introducing Programming in High School Using Python," in *Proceedings of the 8th Australasian Conference on Computing Education*, 2006, pp. 71–80.

[29] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 4, p. 16, 2010.

[30] R. A. Bittencourt, A. S. Rocha, B. L. Santana, C. S. Santana, D. A. Carneiro, G. A. Borges, H. S. Chalegre, J. F. J. Silva, J. M. J. Santos, L. A. Silva *et al.*, "Aprendizagem de programação através de ambientes lúdicos em um curso de engenharia de computação: Uma primeira incursão," in *XXI Workshop sobre Educação em Computação*, 2013.

[31] R. A. Bittencourt, D. M. B. dos Santos, C. A. Rodrigues, W. P. Batista, and H. S. Chalegre, "Learning programming with peer support, games, challenges and scratch," in *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2015, pp. 1–9.

[32] J. W. Creswell, *Projeto de Pesquisa: Métodos Qualitativo, Quantitativo e Misto*. Artmed, 2010.

[33] J. Keller, *Motivational Design for Learning and Performance: The ARCS Model Approach*. Springer US, 2010. [Online]. Available: https://books.google.com.br/books?id=y289SAAACAAJ