Increasing Motivation of CS1 Non-Majors through an Approach Contextualized by Games and Media

Bianca L. Santana
State University
of Feira de Santana
Feira de Santana, Bahia
Brazil 44036–900
biancasantana.ls@gmail.com

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

***Abstract*—This innovative practice full paper reports our attempts to increase motivation of CS1 non-major students through an approach contextualized by games and media. We report our experience to conceive and deliver an introductory programming course for CS non-majors. We used a mixed approach which combines Scratch with game creation, and Python with both turtle graphics and image manipulation in order reduce initial barriers to learning and increase student motivation by using different contexts. The course was given to university-level, freshman, civil engineering students who take programming as a required first-term course. The main lessons learned were: a spiral approach over programming content reduces cognitive load and facilitates learning; careful design for a transition between tools/languages is crucial to minimize difficulties; games and media bring freshmen closer to the programming world; and playful contexts improve motivation, but only to a certain extent.**

I. Introduction

Computer programming is usually seen as something difficult to learn, regardless of audience. To support this idea, one may look at the high dropout and failure rates in CS1 courses, which have not improved over the years [1], [2]. Several factors may influence difficulties faced by beginners, such as teaching approach, choice of programming language, support tools, motivation, and learning styles, to name a few [3], [4]. Among those factors, motivation plays a fundamental role, for learning to code demands practice time, which seems more natural when apprentices are motivated. Non-major students may experience these difficulties at higher levels because, apart from being beginners in CS1 courses, they do not usually have the typical intrinsic motivation of majors. To face those issues, CS1 courses need to be remodeled to meet the needs of non-majors and offer a motivating context [5].

For more than 20 years, most science and engineering students of our university take CS1 as a compulsory course. The goal of this course is to enable students to solve algorithmic problems by building small programs in a current programming language. Traditionally, this course is offered using a lecture-lab approach, supported by languages such as Pascal or C. According to the informal reports by faculty involved, lack of motivation is evident in most students taking this course. Even for students who begin motivated, engagement vanishes when difficulties arise while they are developing their reasoning and coding skills. We believe this happens because these courses are not framed to provide a more adequate and meaningful context to non-majors, contributing to decreased student motivation. Other authors assert that high failure and dropout rates of non-majors in CS1, especially female students, are symptoms of a failure to communicate the value of computing to students [6]. They also state that the generic computer science course is not able to meet the needs of a diverse student body [7].

In order to soften initial difficulties faced by CS non-major students from our institution and improve their motivation, we created a spiral approach to teach programming that combines Scratch, Python and contexts of games, geometric drawings and image manipulation. We applied our approach over two consecutive terms in classes of freshman Civil Engineering students. The main lessons learned in our experience were: the spiral approach to programming content reduces cognitive load and facilitates learning; careful design for transition between tools/languages is crucial to minimize difficulties; games and media bring freshmen closer to the programming world; and playful contexts improve motivation, but only to a certain extent.

II. Background

Over the years, numerous initiatives have been proposed with the aim of reducing difficulties faced by newcomers in CS1 courses. Proposed solutions usually change one or more features of CS1 courses such as curriculum, pedagogy, language, or tools [8]–[10]. In our approach, we use a mix of languages and tools already tried and well assessed in CS1 teaching.

Previous work has already shown that Scratch can be used in the first weeks of CS1 courses to level the class and offer an initial motivating context, before proceeding with other tools and languages. For instance, de Kereki [11] and Mishra et al. [12] present approaches that use Scratch in the first weeks of CS1 courses for engineering students. In these papers, authors realize that using Scratch at the beginning of the course promotes a high level of motivation and a positive perception of students about programming.

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

In addition to Scratch, previous research and experience reports suggest the use of Python for teaching programming to a diverse audience. Shannon [13] describes a change in the CS1 curriculum for IT majors and non-majors, using Python as programming language. The course addresses topics related to the Internet, social computing issues and introduces students to programmable robots and SQL databases. Hromkovic et al. [14] describe an approach to CS1 for non-majors using Logo in the first weeks and Python in subsequent weeks, addressing a wide range of scientific data and problems. Results from these initiatives show that feasibility of using Python to more efficiently introduce programming.

Contextualizing CS1 courses according to the audience is pointed out by various researchers as capable of improving motivation and increasing non-major engagement. Procedures for tailoring a CS1 course to a particular audience involve the recognition of interests and reassessment of learning objectives [5]. Nikula et al. [15] report how a programming course with high dropout and failure rates had its rates improved by more than 50%, by promoting changes aimed to cut out demotivating and general dissatisfaction factors. Joyce [16] describes a course for non-majors focused on presenting the computer as a useful tool to solve problems, especially through programming. Brunvand and McCurdy [17] present a course on "technological fluency", involving electronic and programming content, exploring a context of sound-art circuits, experimental and electronic music, noise-making circuits, hardware hacking, and circuit bending. The authors alter the scenario that was usually contextualized to areas such as chemistry and biology in order to position the course as an interesting choice for a wide variety of non-major students. They realize that technological fluency can be taught in a way that technological content can be seen as a natural component of a broader artistic context.

One context that has proved effective to several audiences is the so-called media computing, a term used to describe approaches that contextualize programming learning through the creation and manipulation of digital media. Typically, the goal of CS1 courses using media computing is not to make programming learning easier, but rather to motivate and increase student engagement to reduce failure and dropout rates.

In addition to contextualizing to an audience that is often poorly served by traditional CS1 courses, the creation of digital media can be listed among the new literacy skills for the computer age [6]. These authors describe the experience of using media computing in a CS1 class for liberal arts students, where they implement PhotoShop-style filters in images, and video and sound special effects using the Python language. The course achieved lower failure and dropout rates compared to the traditional CS1 course offered to the same audience. Most students showed an interest in taking an advanced version of the course. The media approach has positive and wide-ranging effects, is effective for non-majors, promotes more engagement and learning, and higher retention rates in various universities [7].

Contextualizing is important but needs to be combined with a pedagogical approach. In our work, we used a spiral learning approach based on the ideas of spiral curriculum of Bruner [18], [19]. In the spiral curriculum, topics are revisited with increasing level of difficulty. New learning relates to previous learning and students' competences increase with each visit. Each time a topic is revisited, it is applied in a more advanced way. Our approach is not a full curriculum, but is strongly based on the values of the spiral curriculum: reinforcement, moving from simple to complex, and logical sequence [19].

To measure students' motivation, we used the ARCS model of motivation [20]. ARCS uses the conceptual categories of Attention, Relevance, Confidence and Satisfaction to characterize human motivation, and is based on the theory of value-expectation. The attention category relates to obtaining, maintaining, and directing students' attention to the appropriate stimuli. Relevance measures whether instruction seems relevant to the student in various ways, not only in future career opportunities. The confidence category relates to building student confidence by helping to shape the impression that some level of success is possible if effort is exercised. Finally, the satisfaction category relates to practices that help people to feel good about their accomplishments.

A previous experience with measuring motivation in a media computing course for non-majors is described elsewhere [21]. The results, however, are based on a different instrument (IMMS), which is also based on the ARCS model. In a different track, another experience analyzes student motivation with media computing in a Brazilian middle school, both quantitative and qualitatively [22]. They describe factors that affect motivation in their approach by means of concept maps.

III. Course Organization

Our approach keeps the same subject matter but changes languages, tools, teacher's attitudes and assessment. We developed a first version of our approach and applied it to a CS1 class of freshman students in civil engineering and food engineering in the second half of 2016. This was a pilot case study that allowed us to identify the strengths and weaknesses of our approach. Based on these results, we promoted changes in the approach and applied it in another case study, conducted in the first half of 2017, with a group of freshman students of civil engineering.

Our CS1 course has 60-hour workload, comprised of 30 hours of lectures and 30 hours of computer lab sessions. The course is divided into three units, using a spiral learning approach, with programming content repeated with increasing level of difficulty. Table I presents details of each unit. Our spiral approach consists of repeating, for each subsequent unit, part of the concepts of the previous unit before presenting new concepts. This type of approach offers an iterative review of content and, in each iteration, along with the review, content and skills are deepened compared to the previous phase [19].

During lectures, we reduce the presentation of formalisms, and the instructor builds step-by-step examples with the help of students. Assessment is no longer purely theoretical and

Table I
COURSE ORGANIZATION

Unit	Learning Objectives	Content Covered	Performed Activities
I	• Understand what programming is and its importance within CS; • Know the main foundations of programming; • Implement games in Scratch; • Think algorithmically.	• Program execution flow; • Conditional structures; • Iteration structures; • Logic Operations; • Variables; • Functions.	• 3 lectures with tutorials on building games in Scratch; • 3 labs with Scratch exercises; • Homework assessment through two challenges; • Practical exam in computer lab
II	• Know the basic syntax of the Python language; • Create simple programs for drawing figures through the Turtle library; • Implement and use functions consistently.	• Program execution flow; • Conditional structures; • Iteration structures; • Logic Operations; • Variables and data types; • Functions with parameters.	• 4 lectures with tutorials on drawing geometric figures; • 3 labs with exercises for drawing geometric figures; • Homework assessment through two challenges; • Practical exam in computer lab.
III	• Know the structure of an image and associate it with the concept of matrix; • Implement functions for manipulating elements of color and structure of an image; • Build programs that input and output data, and read and store files.	• Data input and output; • Reading and storing media files; • Conditional structures; • Iteration structures; • Logic Operations; • Variables and data types; • Functions with parameters and return values; • Vectors and Matrices; • Encoding images.	• 5 lectures with image manipulation tutorials in Python; • Final Project: image editor

carried out in a classroom, but, instead a practical exam carried out inside the lab in Units I and II, and a project, in the third unit. Grading exams is performed as soon as the student finishes the exam and the instructor gives the student feedback on his/her mistakes. In Unit III, each student builds his/her own image editor, which is assessed by the instructor.

A. Unit I – Scratch and Games

Unit I aims to soften the initial difficulties of beginners. We use Scratch for this tool uses block-based visual programming to facilitate learning, allowing students to master programming constructs and focus on logic problems prior to syntax [23]. Through Scratch, we introduce logic and basic programming concepts such as control structures, variables and procedures.

The first lessons use the Scratch pen. Subsequent lessons approach the concepts in the context of building classic games such as Pong, Space Invaders, Bow and Arrow, and Interlagos, and are based on previous work of our research group with programming novices [24], [25].

During lectures, very few explanatory slides are used. Instead, the instructor uses computer and projector to show how to code the examples. In each lab session, students are challenged to build a different game, with greater difficulty each week. Students are assessed through solving two homework challenges and performing a practical exam at the end of the unit, held in the lab. No particular textbook was adopted for the course. All materials needed for classes were developed together with the instructor. As lessons occur, slides and examples are posted to the course support site [1].

B. Unit II – Python and Turtle

In Unit II, we reinforce acquired knowledge and facilitate the transition from a block language to a textual language. We chose Python, a popular programming language for teaching, because of its versatility, offering support for modular and object-oriented design [26]. In addition, Python offers fewer cognitive obstacles for beginners to overcome, since fewer concepts should be taught before coding a program; it is interactive, allowing the use of the prompt to test program fragments; runs code until finding an error; and it forbids incorrect indentation [27].

We introduce Python, and we use the same programming concepts from Unit I, now reinforced by the use of the Turtle library in Python, and challenging students to face textual language syntax and their associated errors. We use the standard IDLE development environment.

In the first lesson, we show how drawings with the Scratch pen are equivalently coded with Python and Turtle. The other lessons are dedicated to explore the same concepts discussed in Unit I and, additionally, the concept of parameters in functions. Before assessment, we review concepts, approaching them from a more formal point of view. For instance, we explain the different variable types and how each type is encoded by the computer. This class also reinforces the concept of function definition and use, and the use of parameters.

During lectures, the instructor explains the concepts by building examples that make more complex drawings, such as kaleidoscopes and chessboards. In labs, students receive a challenge guide with activities of similar difficulty to the examples explored in lectures. Student are assessed by solving two homework challenges and an exam, performed in the lab.

C. Unit III – Python and Media Computation

In Unit III, we keep the Python language, but change the context to media computing, where students manipulate images in an approach similar to that of [28], using the JES environment. The goal is to provide a context more creative than the usual, where students can understand how media

[1]Web site (in Portuguese): https://sites.google.com/site/uefsicc20171/aulas.

are encoded by the computer and create their own media while learning concepts from introductory programming and computer science. We seek to reinforce the concepts already learned and, additionally, we present the concepts of vectors, matrices and more complex constructs such as nested loops.

The first three lessons (i.e., two lectures and one lab) aim to introduce the Jython Environment for Novices (JES) and work with general purpose examples such as textual games and computing areas of geometric figures. This promotes a lighter and beneficial context transition to students. In addition, the use of general purpose programs at the beginning of this unit reinforces the use of Python as a more professional language. The other lessons cover topics on creating and manipulating images. Examples built in lectures range from simple image filters (e.g., negative, grey scale) to more complex effects like Chroma Key.

Students' assessment is carried out through a three-milestone image editor project. This editor used some of the filters implemented in the classroom, but also adds particular requirements to ensure that students explore the taught concepts. Lab practices in Unit III follow the development of students' projects.

IV. Our Experience

The current version of our approach was applied in a CS1 class for freshmen of Civil Engineering, during the first half of 2017. Class consisted of 37 freshmen, of whom 25 (67.6%) were male and 12 (32.4%) were female. Lectures were taught in a classroom with all students, while computer lab sessions split students into two groups of up to 20 students, each group working in different times. Labs were also assisted by two undergrad Computer Engineering students. All lessons were taught by the same faculty, who is also one of the authors of this work. The instructor adopted the materials and procedures defined in the approach.

We conducted observations throughout all classes. Observations were drawn only from students who agreed to sign an informed consent form. We also asked students to answer a course interest survey (CIS) at the end of the course to measure their motivation according to the previously described ARCS model of motivation.

A. Unit I – Scratch and Games

During Unit I, students were very enthusiastic about the course. In the first class, the instructor presented the course syllabus, explained the previous scenario of CS1, and the changes to the teaching approach during that term to make it more engaging. Initial activities in classroom showed how to use the Scratch pen with sequential instructions. Gradually, the instructor introduced concepts of loops, conditions, input and output, and procedures in Scratch. Students were asked to code a program for drawing geometric figures according to user choices as a homework.

In the second lesson, we started building games. The instructor showed a video with the gameplay of the first proposed game, Bow and Arrow. Students were enthusiastic, and they sang a chorus *"eeeeeee"* (LE02). The instructor created a new Scratch project and started to implement the game. Before coding each feature, he asked for suggestions from students. Students were very interested, giving frequent suggestions and pointing out possible implementation errors. Whenever the instructor tested the game, students clapped their hands and celebrated as the archer hit the balloons inside the game. This student behavior was also maintained during the classes where the Space Invaders game was built. The instructor kept the same attitude of waiting for students' suggestions before coding each feature. At each opportunity, the instructor explained how the programs are gradually implemented, always using a context of civil engineering as example. One relevant observation is that the motivation of the female and male audience were similar. After building the Space Invaders Game, the instructor asked students to add two new phases, as homework. All students did this task, and many implemented features beyond requested or completely reframed the game, changing rules and theme.

Lab sessions took place each week in alternation with lectures. In each lab session, students were asked to implement a game with difficulty level similar to the game implemented in the previous lecture, but with a different theme. Initially, the instructor presented a gameplay video of the original game, and students were given a challenge guide, where each challenge was a feature of the requested game. In the first lab session, students were asked to develop the Pong game, and, in the second lab, the Interlagos racing game. Each student had a different pace: various could finish the game before session end, without much help of the assistants; others, on the other hand, needed assistence, nonetheless were able to finish the challenges before session end.

Exam assessment involved three questions that asked to fix a bug or implement a new feature in a Scratch project. In each question, students chose between two projects with different themes: one project had a more feminine aspect and the other more masculine. Students had one hour to answer questions and the next hour was used for grading and feedback. They received their results as soon as they finished the exam, along with feedback on the mistakes made. In Unit 1, only 6 students performed below passing grade, and the average grade of this unit was 8.5 ± 1.9 points (out of 10).

B. Unit II – Python and Turtle

During Unit II, students remained enthusiastic about the course. In the first lesson of this unit, the instructor stated that, from that day, they would learn to code in a professional language and that, if they committed themselves, they would finish the course knowing how to use programming to solve various problems, including engineering problems. An introduction to the Python language and commands of the Turtle library followed, rewriting the same programs made with the Scratch pen. From the second lesson, lectures brought examples of increasingly complex designs, with each example introducing new programming concepts. The instructor took a stand of reminding how they implemented those concepts

in Scratch before introducing new commands. This ensured a smoother transition. On the other hand, as the examples became more complex, the classroom showed slightly less engagement.

Two homework challenges were requested. The first required that, given a stylized letter font template, each student should draw his or her own name. The second challenge called for the creation of a function to draw chess-type boards, but with variable size. On the answers delivered by the students, we noticed that various showed difficulties, both in the use of functions with parameters and in the use of variables. Another perception is that students implemented only the features requested in the challenges, different from the Unit I. From the difficulties identified, we prepared a review class reinforcing those themes. The concept of variable received more attention: different variable types were discussed and the way each is encoded by the computer, plus the difference between use and assignment.

Lab sessions took place in alternation with lectures. During labs, each student received a challenge guide and could ask for the help of the teaching assistants. We noticed that students were more in demand for assistance than in unit I, and that not everyone was able to finish all the planned activities during session time.

An exam was carried out in the computer lab, and was comprised of four questions organized in increasing level of difficulty. Assessment was planned to last for an hour, but students demanded more time, which delayed grading and feedback a bit. Students' performances in assessment activities was similar to Unit I, with only five students performing below passing grade. Average grade of Unit II was 8.8 ± 1.9 points (out of 10).

C. Unit III – Python and Media Computation

Initial lectures of Unit III aimed to present the JES environment and to review all programming concepts employed in Unit II through the construction of general purpose examples. Examples in these first three lessons used variables, select and repeat structures, and functions. The only new concepts presented in introductory classes was of functions with return values and while loops. During these lessons, students presented more inattentive and careless behavior, participating less in class. Whenever they participated, it was to clear doubts and not to suggest strategies for coding, as was the case in Units I and II. In spite of this more passive behavior, comments were often heard emphasizing the usefulness of the content being taught. For instance, in the lecture where a program was implemented to compute student grades, many emphasized its utility with comments such as: *"Wow! I will use this a lot ..."* (LE10).

Student participation was increased when the instructor started to discuss image manipulation. In the first lecture, he showed how the computer encodes an image, relating to the concept of vectors and matrices. The first lab session asked students to create a program to edit images for posting on social networks. Various students showed satisfaction with this: "How cool! Let's become Zuckerberg and post on Facebook!" (LE08). We also noticed that students still were in doubt about function calls, parameter passing, and mainly the use of returning types in functions.

The second lecture presented how an image is composed in terms of color systems and how to create filters such as black-and-white and negative. The third lecture brought examples that manipulate the structure of an image, and presented code to manipulate images producing mirrored images and collages. From this lesson onwards, students appeared to have many doubts related to the understanding of matrix manipulation. In the fourth lecture, the instructor presented the effects of background removal and Chroma Key. All examples were built step-by-step during class, which enhanced student participation. The lessons in Unit III involved a lot of curiosities about image manipulation, and this caught students' attention. As the filter results are immediately seen, various manifestations of astonishment and satisfaction were observed: "Ownnn... How I wanted to do this!" (LE14).

For assessment in Unit III, each student developed a project of an image editor with requirements different from programs used in lectures but with similar level of difficulty. Implementation of editor functions was started during lab sessions and students could pose questions to the instructor and assistants to clear their doubts. In general, students had questions about the filter implementation, and they had difficulty managing the size of the programs, since they were larger when compared to the programs implemented in Unit II. Despite the support offered during lab sessions, not all students were able to implement all the requirements of the image editor. The difficulties with English became even more evident as students had difficulty interpreting help screens and the error messages in the JES environment. We also had issues of plagiarism, which resulted in a larger number of students who did not reach passing grade in this unit, 10 students, in total. However, the average performance of the class remained satisfactory, with an average of 8.2 ± 2.7 points (out of 10). Looking at the final results, only one student quit the course, and among those who remained, three students failed.

D. Survey Results

A synthesis of our survey results on students' motivation is shown in Figure 1. Values are based on a Likert scale for each motivation category from ARCS, as proposed by Keller for the Course Interest Survey (CIS) questionnaires [20]. Results show, for each category, mean values between moderately true (3) to mostly true (4). Higher average values arose for both relevance (3.40) and confidence (3.46), while attention and satisfaction both had average of 3.12. Although we do not compare with other scenarios, absolute results between 3 and 4 in a scale from 1 to 5 point to good student motivation with our approach.

Analyzing the questions in the CIS questionnaire, we noticed particular issues. Attention, as can be seen in Figure 2, was raised by the variety of teaching techniques and for stimulating student curiosity. Most students considered that

their attention was captured in the course. Relevance, shown in Figure 3, had better results, especially for the feeling that students had that the course was useful to them, and that is was related to things they already know, which is probably aided by the familiar contexts of games and media. Regarding confidence, as shown in Figure 4, students' perceptions suggest that they believed being in control of their own success, ignoring luck and valuing effort. Finally, as can be seen in Figure 5, student satisfaction was not so high, because few students felt the course gave them a lot of satisfaction, although some satisfaction was present because of appropriate feedback, fair results and grading, and adequate rewards for their effort. We also computed a t-test to check for motivation differences between male and female students for each category of the ARCS model, but we have found no significant differences. Those results agree with our observations in class.

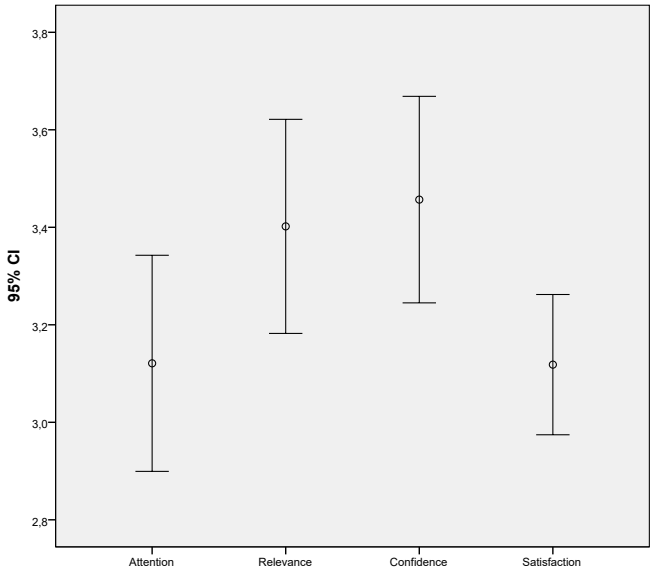

Figure 1. Error Bar Diagram – Motivation Categories

V. Lessons Learned

After two semesters applying our approach, some challenges and lessons learned are evident, such as the advantages of the spiral approach to content, the issue of transitioning between contexts, and the context chosen itself. However, we recognize that our approach does not fully solve the issues of lack of motivation of non-major students. The following is a discussion of each of these lessons and issues.

A spiral approach to introduce programming content. In our approach, each unit repeated all the concepts of the previous unit before presenting new concepts. On some occasions, the same concept was approached more than once within the same unit. In Unit I, for instance, games to be implemented employed the same commands, albeit in a different way. Such repetitions require a lot of time and, given the limited time frame of a course, may narrow the range of new content that can be addressed. However, this approach gives students with more difficulties the opportunity to review explanations and level up to other colleagues, besides reducing the cognitive load to learn something new. When repeating content matter, care must be taken not to bore students who are at a more advanced pace. As our approach makes use of different contexts and tools in each unit, this problem has been mitigated.

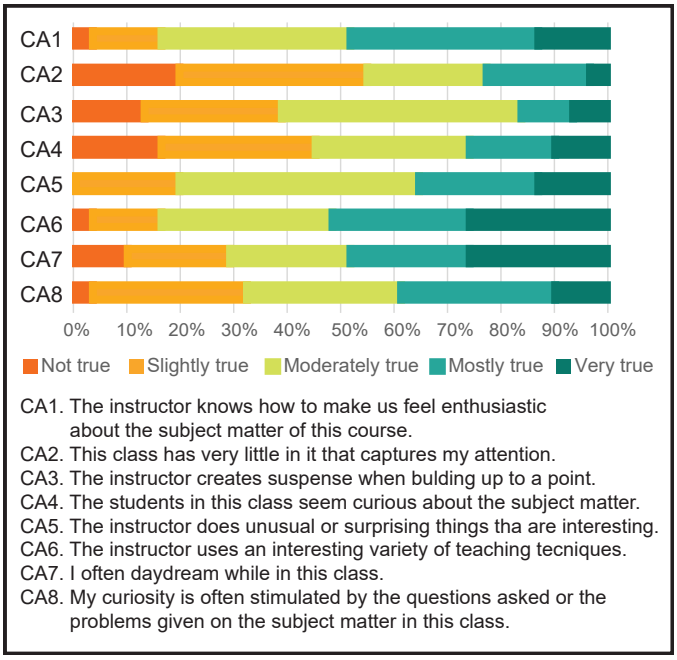

Figure 2. Results for Attention Category

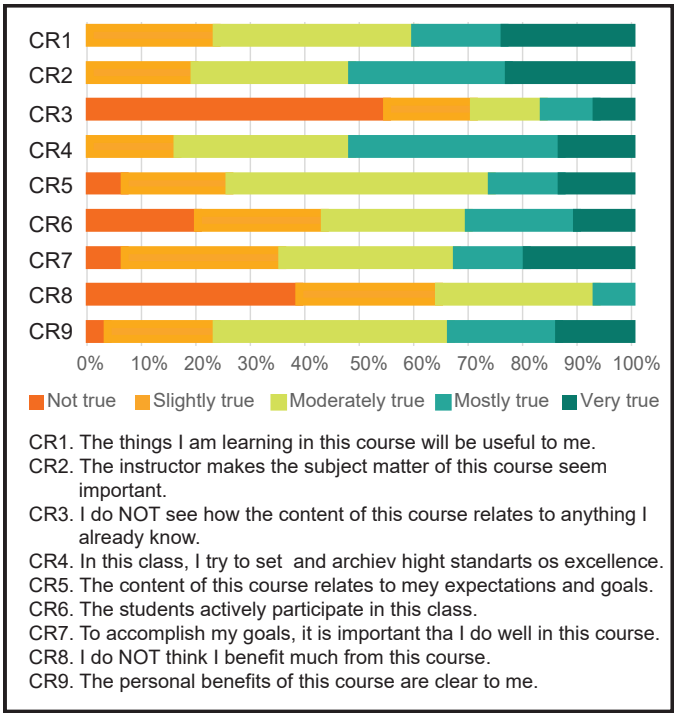

Figure 3. Results for Relevance Category

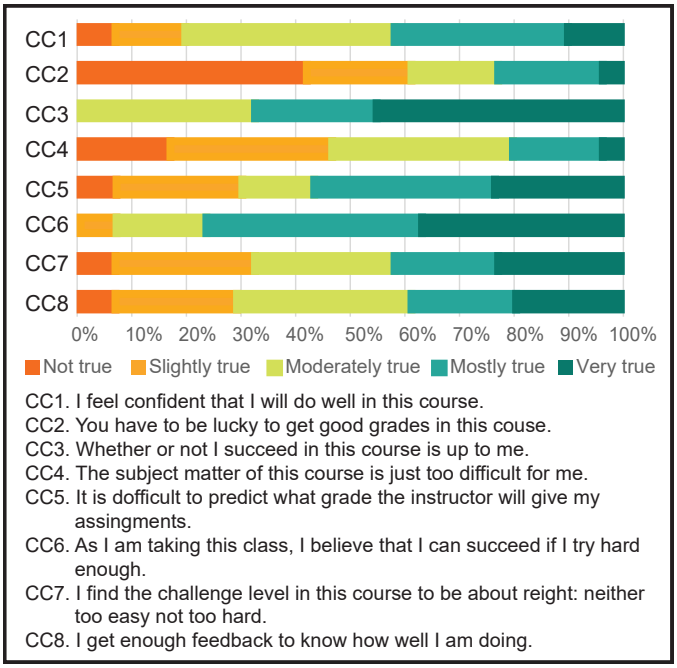

Figure 4. Results for Confidence Category

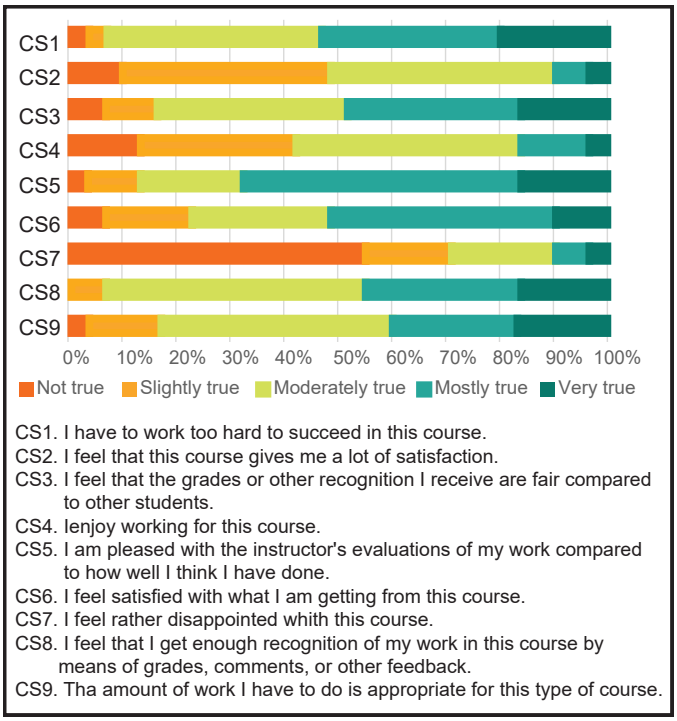

Figure 5. Results for Satisfaction Category

Beware of the transition between tools and languages. We realize that the transition between contexts takes time but enhances our spiral approach. The choice of Scratch for Unit I was intended to promote a softer introduction to programming concepts. Since it is a block-based and visual language, students do not have to worry about syntax details. Since the environment is internationalized, students are able to interact in their mother tongue, and their commands are more closely related to their human language, which makes practices cognitively less demanding. Using the Scratch pen in the first lessons of Unit I facilitated the transition to the Python language with the Turtle library in Unit II, since source code becomes highly comparable. We took similar care in promoting the context transition between Units II and III. Before dealing with media manipulation, we booked a few classes to explore exercises with Python without the Turtle library, where we implemented, for instance, some textual games and programs for computing areas of geometric figures. Nevertheless, it is undeniable that as syntax details arise and code complexity grows, students present more difficulties. However, we believe that students' difficulties with understanding Python have been softened by our approach. Another issue worth mentioning is that the diversity of exercises using the same programming concepts in different contexts has reinforced the usefulness of programming for problem solving.

Immediate feedback on practical exams. We consider the immediate feedback offered after the practical exams one of the strengths of our approach. We believe that students feel benefited by this practice, especially those with greater difficulties, because they are explained on the causes of their mistakes and can individually discuss them with the instructor. However, one must take into account that this model requires some more preparation time, because the questions must require short time to answer and, at the same time, allow assessing the acquisition of various skills.

Benefits of the context. Although students present greater engagement during the Unit I, we realize that all contexts are good and bring students closer to activities they enjoy doing. The use of Scratch in a context of game creation, and of Python in the context of media, helped create a more relaxed atmosphere during class. As the procedure to test the game is by playing it, students pay attention and participate actively. In Unit III, where the content load was greater, image manipulation created the expectation of viewing correctly applied effects. That gave them purpose, avoiding students falling into boredom. Another positive feature of using these contexts is that students were more interested in completing activities, and used more creativity, at times even implementing unsolicited features. This is evident in comparison to our perception of previous experiences in a more traditional approach, where students appeared to be engaged only to get good grades. We felt that students could take more advantage of the classes if we offered a context more related to civil engineering. However, when we consider that the course is taught to freshmen, who know little about the universe of civil engineering, contextualizing the course with games and media manipulation is apparently more significant. Since students become familiar with the basic foundations of programming, later on, in courses such as Numerical Methods, usually offered when students have more mastery of the engineering field, students will be able to use programming to solve problems more related to their professional choice. Another positive aspect of the

program-neutral contextualization of this experience is that we can use the same approach with non-majors from other programs. which is often the case for department decisions that mix students from different programs in the same class.

No silver bullet. Our new approach does not solve all the issues related to the lack of motivation and the difficulties of non-major students in our institution. A playful context encourages student participation, but they still perceive programming as difficult to learn. This was evident in Unit III, where although classes were interesting and kept a good deal of students' attention, many could not properly apply the concepts and finish all the features of the image editor project, showing that concepts were not easy to understand. Furthermore, we still need to improve the offered learning experience, including aspects related to regional factors that negatively affected learning such as difficulties with English.

VI. Conclusions

In this paper, we report our experience of promoting a change of approach to teach programming to CS non-major students of our university. Our approach combines the use of Scratch in a context of game creation, Python with the Turtle library in a context of figure drawing, and Python in a context of image manipulation.

The changed course was given to freshmen of civil engineering, a program that has CS1 as a compulsory subject in the first semester. Our initial perceptions reveal that student interest in the course has greatly improved when compared to past experiences. From our observations, we perceive that each unit, language, and tool has potentialized learning and motivation in different ways. In general, the first unit with Scratch had more positive results in terms of student performance than the following units, followed by the second unit with Turtle. It became clear that using a spiral-based approach to programming content and carefully transitioning between tools/languages are beneficial to minimizing difficulties and facilitating student learning. In addition, contextualization with games and media brings newcomers to the programming world.

We are presently carrying on a more in-depth analysis of the data obtained with our approach and plan to report results in a research paper. We also intend to apply and observe the effects of our approach with other non-major audience of our institution, such as food engineering and math and science teaching programs. We also intend to deepen the studies on the spiral approach to content, which proved to be one of the main strengths offered by our approach in both case studies.

Acknowledgment

The authors would like to thank the students of the State University of Feira de Santana that took part in this experience.

References

[1] J. Bennedsen and M. E. Caspersen, "Failure Rates in Introductory Programming," *ACM SIGCSE Bulletin*, vol. 39, no. 2, pp. 32–36, jun 2007.

[2] C. Watson and F. Li, "Failure rates in introductory programming revisited," *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pp. 39–44, 2014.

[3] T. Jenkins, "On the Difficulty of Learning to Program," *Language*, 2002.

[4] A. M. Azzam and Y. Career, "Why students drop out CS1 course?" in *Proceedings of the Second International Workshop on Computing Education Research*, vol. 64, 2006, pp. 97–108.

[5] A. Forte and M. Guzdial, "Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses," *IEEE Transactions on Education*, vol. 48, no. 2, pp. 248–253, 2005.

[6] ——, "Computers for communication, not calculation: Media as a motivation and context for learning," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. IEEE, 2004, pp. 10–pp.

[7] M. Guzdial, "Exploring hypotheses about media computation," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ser. ICER '13. New York, NY, USA: ACM, 2013, pp. 19–26.

[8] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming," *ACM SIGCSE Bulletin*, vol. 39, no. 4, pp. 204–223, 2007.

[9] A. Vihavainen, J. Airaksinen, and C. Watson, "A systematic review of approaches for teaching introductory programming and their influence on success," in *Proceedings of the tenth annual conference on International computing education research*. ACM, 2014, pp. 19–26.

[10] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[11] I. F. de Kereki, "Scratch: Applications in computer science 1," in *38th Annual Frontiers in Education Conference*. IEEE, 2008, pp. T3B–7.

[12] S. Mishra, S. Balan, S. Iyer, and S. Murthy, "Effect of a 2-week scratch intervention in cs1 on learners with varying prior knowledge," in *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 2014, pp. 45–50.

[13] C. Shannon, "Another breadth-first approach to cs i using python," in *ACM SIGCSE Bulletin*, vol. 35, no. 1. ACM, 2003, pp. 248–251.

[14] J. Hromkovič, T. Kohn, D. Komm, and G. Serafini, "Combining the power of python with the simplicity of logo for a sustainable computer science education," in *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Springer, 2016, pp. 155–166.

[15] U. Nikula, O. Gotel, and J. Kasurinen, "A motivation guided holistic rehabilitation of the first programming course," *ACM Transactions on Computing Education (TOCE)*, vol. 11, no. 4, p. 24, 2011.

[16] D. Joyce, "The computer as a problem solving tool: a unifying view for a non-majors course," in *ACM SIGCSE Bulletin*, vol. 30, no. 1. ACM, 1998, pp. 63–67.

[17] E. Brunvand and N. McCurdy, "Making noise: Using sound-art to explore technological fluency," in *Proc. of the 2017 ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '17. New York, NY, USA: ACM, 2017, pp. 87–92.

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

[19] R. M. Harden, "What is a spiral curriculum?" *Medical teacher*, vol. 21, no. 2, pp. 141–143, 1999.

[20] J. Keller, *Motivational Design for Learning and Performance: The ARCS Model Approach*. Springer US, 2010.

[21] B. L. Santana, J. S. L. Figueredo, and R. A. Bittencourt, "Motivação de estudantes non-majors em uma disciplina de programação," in *WEI 2017 – XXV Workshop sobre Educação em Computação, São Paulo. Anais do XXXVII Congresso da SBC*, 2017.

[22] 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.

[23] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," in *ACM SIGCSE Bulletin*, vol. 39, no. 1. ACM, 2007, pp. 223–227.

[24] 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.

[25] 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.

[26] T. Koulouri, S. Lauria, and R. D. Macredie, "Teaching introductory programming: A quantitative evaluation of different approaches," *Trans. Comput. Educ.*, vol. 14, no. 4, pp. 26:1–26:28, Dec. 2014.

[27] T. Jenkins, "The first language-a case for python?" *Innovation in Teaching and Learning in Information and Computer Sciences*, vol. 3, no. 2, pp. 1–9, 2004.

[28] M. Guzdial, *Introduction to media computation: A multimedia cookbook in Python*. Citeseer, 2004.