Motivation of Engineering Students with a Mixed-Contexts Approach to Introductory Programming

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

Jose Solenir L. Figueredo
State University
of Feira de Santana
Feira de Santana, Bahia
Brazil 44036–900
solenir.figueredo@gmail.com

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

***Abstract*—This research to practice full paper investigates motivation of engineering students with a mixed-contexts approach to introductory programming. Non-major CS1 students usually have more difficulties learning computer programming than CS majors, for reasons such as lack of interest in computing or courses formatted for CS majors. Student motivation is vital in the learning process, and fostering it is important to promote more effective learning. Various initiatives have been proposed to increase engagement and reduce the difficulties faced by non-majors in CS1 courses. We believe that motivation can be improved by using a mix of languages and tools already tested in CS1 teaching. In this paper, we evaluate a teaching approach for non-majors, which combines the use of Scratch in a context of game creation, and Python with both turtle graphics and image manipulation. We performed an exploratory case study with CS1 civil engineering students from our institution to analyze the impact of this approach on student motivation. Our results describe motivation arisen during the course in terms of attention, relevance, confidence and satisfaction (ARCS), and as practical factors that contribute to either increase or decrease motivation.**

I. Introduction

Computing plays an ever-growing role in people's personal and professional lives. This raises a demand for general understanding of basic principles of computer science and how to apply them. As information technology fluency and computational thinking become more relevant to modern life, CS educators need to meet the demand of diverse audiences by creating courses for CS non-majors. Various authors state that programming is a fundamental component of any proposal of introduction to computer science appropriate to undergraduates, for its impact on problem-solving skills, precision of thinking, accuracy in exposition, and consideration of details [1], [2]. Regardless of audience, however, programming is usually hard to learn, and non-major students can experience these difficulties at higher levels, as they are usually both novices and more interested in a different field.

Several factors may aid in understanding difficulties faced by non-majors CS1 courses. Among them, motivation plays a key role. Learning how to code demands practice, which is more natural when apprentices are motivated [3], [4]. To be successful, CS1 courses should address the needs of non-majors and provide a motivating context [5]. Over the years, various initiatives have been proposed to reduce difficulties faced by non-majors, and to improve their engagement. Results show that this audience takes more advantage of courses tailored to their needs than the traditional CS1 course [6].

In our institution, science and engineering students must take a compulsory CS1 course. Our instructors' perceive a failure of the traditional CS1 course, with a more formal introduction to languages such as Pascal or C, to engage non-major students. Most of our engineering students take CS1 while they are freshmen. To increase their engagement, we promoted changes to this course to reduce their initial difficulties. We try to bring computer science closer to their likes, and, consequently, improve their motivation. Our approach proposes a mix of languages and tools already tested in CS1 teaching: the Scratch tool in a context of game development, and the Python language aided by a screen turtle library and an environment for image manipulation.

Our goal was to understand how the proposed approach influences student motivation through a case study carried out with a CS1 class of Civil Engineering freshmen from our institution. From this goal, we raised two research questions: *i) How does student motivation arise in this approach in terms of its different dimensions of attention, relevance, confidence and satisfaction?*, and *ii) Which are the various likely influences of this approach on student motivation?*. Since this was an exploratory case study, we had no previous hypotheses to test. Instead, we focused on an in-depth analysis of motivation, which is the main contribution of this work. Results suggest that introducing programming concepts through Scratch in a context of game creation enhances student motivation and facilitates later understanding of Python. Nonetheless, as the complications of a "real-life" programming language arise, motivation levels fall. We also identify aspects of our approach that positively and negatively influence student motivation, and describe them in a preliminary model.

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

II. Background

Motivation is a word that describes processes that can awaken, give purpose, and continue to allow behavior to persist, and to lead to choose a particular behavior [7]. Motivation to learn may be related to the skills of self-regulation, learner control and meta-cognitive behavior [8]–[11]. The theory is value-expectation assumes that people are motivated to participate in an activity if they realize that it is related to the satisfaction of their personal needs, and if there is a positive expectation of success [12], [13].

Por outro lado, outros trabalhos mensuram motivação em CS1 de maneiras diferentes. Citar outrso trabalhos como jenkins etc....

Several motivational techniques and strategies may be used by teachers to strengthen student performance and reinforce positive attitudes toward learning [7]. Moreover, one needs to know how to assess motivational factors in class. Keller [14] presents the Attention, Relevance, Confidence and Satisfaction (ARCS) model that includes sets of strategies to increase the motivational appeal of instruction. ARCS is based on the theory of value-expectation and contains four conceptual categories that supply many of the concepts and variables that characterize human motivation. The attention category provides guidelines for obtaining, maintaining, and directing students' attention to the appropriate stimuli. The relevance category offers guidelines so that instruction seems relevant to the student in many ways, not only in present or future career opportunities. The confidence category provides guidelines for building student confidence by helping to shape the impression that some level of success is possible if effort is exercised. The satisfaction category embodies practices that help people to feel good about their accomplishments.

ARCS incorporates a systematic design process, called motivational design, that may be used in material creation and lesson planning [15]. To use ARCS, one needs to classify the motivational problem to be solved and analyze the audience to identify motivational gaps and to prepare motivational goals.

Most studies on motivation in learning focus on the description of motivational strategies applied with no support of empirical data [16]. However, there is a measurement tool that can be used in conjunction with the ARCS model to diagnose motivational problems within instructional materials. The instrument is called Instructional Material Motivation Survey (IMMS), and has 36 Likert-scale declarations. All of them are developed according to individual ARCS components [15].

The ARCS model and IMMS have been widely applied in studies that aim to promote and evaluate motivation strategies [16]. Various studies adapt the original instrument to its own context. Hamada [17] introduces an integrated web-based environment for teaching topics of computer theory, whose design and evaluation take into account the ARCS model guidelines, and consider the active and collaborative learning preferences of computer engineering students. Savi et al. [18] propose a model, based on ARCS and other theories, to evaluate the quality of games for software engineering education. The proposed model measures whether the game captures students' attention, shows relevance and motivation, and uses the knowledge learned. von Wangenheim et al. [19] present an educational board game to teach and reinforce the application of value added management concepts in the context of undergraduate courses in computing. Among other factors, motivation is evaluated according to the categories of the ARCS model. Such work demonstrates the flexibility of the ARCS model and a trend to care about student motivation when preparing courses. In our work, we rely on the ARCS model to identify motivational problems, propose changes and later evaluate them.

III. Methodology

In this work, we used a methodology of mixed-methods case studies, with qualitative and quantitative data collection, as it allows an exploratory analysis of the learning environment.

A. Intervention

The intervention was conducted during the second half of 2016, in a CS1 class for non-majors, offered to civil engineering freshmen in our institution. This course has a load of 60 hours evenly split between lectures and lab sessions. Traditionally, it hes been offered using languages like Pascal or C, and following a sequence of textbooks and language manuals. We changed both language and tools as well as teaching practices and assessment instruments.

The course is divided into three units. The first unit used the Scratch tool to soften the introduction of basic programming concepts to beginners. We based our choice on previous reports, which show the potential of Scratch as an introductory tool [20], [21]. The teaching of basic programming concepts through Scratch, a visual tool with a block-based programming language, facilitates a smooth transition to Python, reinforcing the learned concepts [22]. In the second unit, we introduced Python and reinforced the same basic concepts from Unit I by using the Turtle library to draw geometric figures. In the first classes of this unit, the examples developed with the Scratch pen were placed side by side with similar examples implemented with Python and Turtle. This correspondence was used to highlight syntax details. The use of the Logo-based Turtle graphics library has been suggested as a good solution to ease difficulties of non-majors [23]. In the third unit, we kept Python, but changed the context to image manipulation, similarly to Forte and Guzdial's approach [5], with the JES environment. At the beginning of the unit, examples work by manipulating color. At the end of the unit, we move to examples that change the structure of an image.In addition to reinforcing basic concepts previously used, additional concepts such as vectors and matrices are introduced.

We adopted a less formal take on lectures. Instructors built examples incrementally, frequently asking students to help them. Assessment was practical and performed in the lab in Units I and II, followed by a practical project in Unit III. Grading of the practical exam occurs as soon as the student

finishes it and the instructor gives the student feedback on the mistakes made.The practical project required students to build their own image editor.

B. Research Participants

The class was composed of 40 civil engineering freshmen, plus 14 food engineering sophomores who had previously failed the course. To reduce variations in student profile, we only used data of the freshmen. Lectures were taught by three different instructors, each one responsible for one unit. During the whole course, students were split into three lab groups, with two separate freshmen groups of 20 students each. Both lab groups were conducted by the same instructor. In the labs, the instructor was aided by an undergraduate assistant. The authors of this paper were not instructors in this course, neither in lectures, nor in labs. Students who wished to participate in the research signed an informed consent form. Only 36 freshmen agreed to participate: 27 (75%) were male and 9 (25%) were female. The mean participants' age was 20.75 ± 2.91 years.

C. Data collection and analysis

To collect data, students answered three surveys to evaluate the relationship between the approach and its impact on learning and motivation. The first survey was answered at the beginning of the course, to gather participants' profile. Two additional survey questionnaires were adapted from the Instructional Material Motivation Survey (IMMS) [16]. One was answered at the end of the second unit, and it aimed to identify the motivational aspects of the game context with Scratch and to better understand the transition to the Python language. The other, also based on IMMS, was answered at the end of the course, and it sought to identify the motivational aspects of the media manipulation context. We also observed lectures and lab sessions, and performed two semi-structured interviews. The first interview was held with seven students, and the second, with six students. Interviews sought to identify traits of motivation and collect students' opinions on the tools and activities. We encoded the qualitative data through content analysis and later described the central themes uncovered. We used descriptive and inferential statistics to analyze quantitative results. Finally, we triangulated qualitative and quantitative data to strengthen the interpretation of results.

IV. Student Motivation

Here we try to answer our first research question: *How does student motivation arise in this approach in terms of its different dimensions of attention, relevance, confidence and satisfaction?* We assessed students' motivation levels in the four categories of the ARCS model. We analyze the individual results for each unit according to the categories.

A. Attention

In Unit I, the results of the IMMS-based survey (Figure 1) show that most students agreed that: the variety of exercises helped them to maintain attention; they learned surprising or unexpected things; and they believed that the use of games motivated the learning process. From observations during lectures, we noticed that students kept a participatory behavior. We found that using Scratch in a context of creating games from small challenges stimulates creativity, which helps them to keep attention on the tool. All interviewed students positively evaluated the use of games, asserting that Scratch facilitates learning and makes classes more fun: *"Scratch was interesting, [because] it arouses creativity, curiosity too, because as we solve the challenges, it encourages us to want to improve, to put more stuff."* (P35).

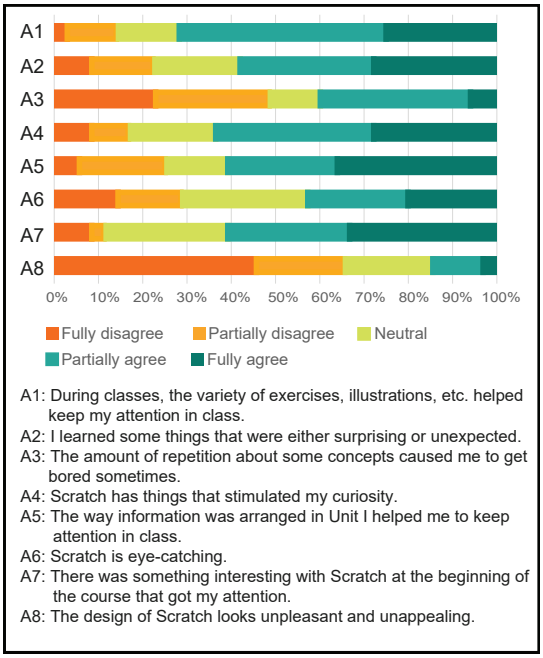

Figure 1. Attention in Unit I.

In Unit II, during the first classes, which aimed to translate concepts from Scratch to Python, using side-by-side code with the same goals implemented in both tools, we observed high student participation. But participation frequency decreased as classes showed a greater content load. Most students agreed that they have not been able to keep pace with classes in this unit.

In Unit III, the results of the IMMS-based survey (Figure 2) show that most students found that the variety of exercises and how information was organized did not help to keep attention in class. In contrast, most students agreed that they learned surprising or unexpected things and that there was something in the first few classes that caught their attention. Observations in class showed that lectures with curiosities about image encoding and more significant examples, such as Chroma Key, were able to keep students' attention, even when dealing with more complex concepts. When lectures did not use this approach, using a more descriptive style and presenting lots of code at once, most students were distracted. Often, they took advantage of class time to solve exercises from other courses.

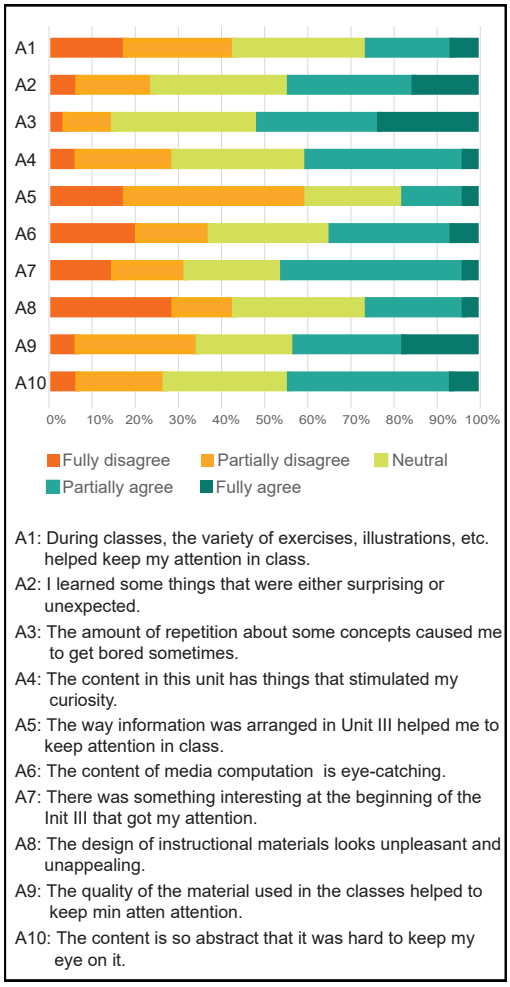

Figure 2. Attention in Unit III.

B. Relevance

Although classes in Unit I have caught students' attention, the results of the survey (Figure 3) show that most participants disagreed that the Scratch content was relevant to their interests. Scratch did not give the sense of practical use in students' lives, which reduces their perceived relevance to future career opportunities. During interviews, when asked what they expected from the course, students pointed out that they wished to learn how to use software for civil engineering: *"I thought it would be things related to civil engineering, for example, how to tinker with engineering software, like AutoCad, some of these [systems]"* (P18) or just a computer class: *"I imagined that I would even play with normal computer programs... so that would be a very basic thing, but not introducing computer science itself, right?"* (P03).

Adapting a CS1 course to a particular audience involves understanding students' interests [6]. Thus, we knew in advance that the tools chosen for the three units would not arouse relevance in a sense of practical utility, but we sought to meet other relevance criteria. According to Keller [14], the feeling of relevance may arise for the way something is taught,

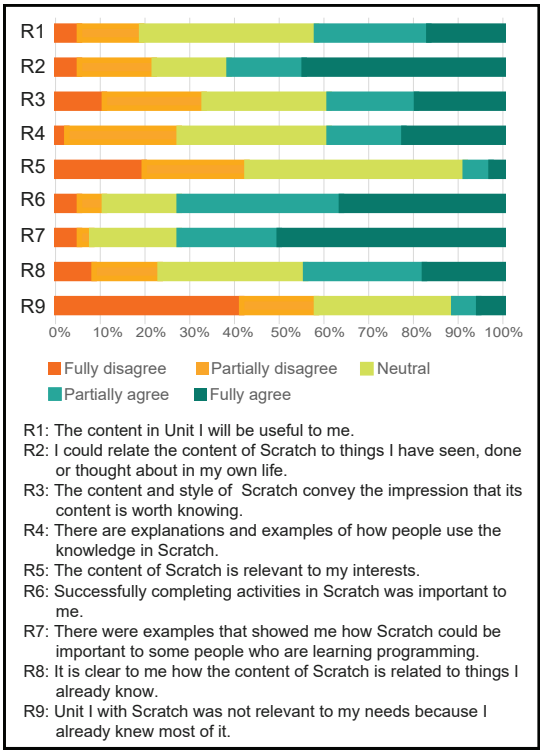

Figure 3. Relevance in Unit I.

satisfying certain personal needs of the student, such as giving people with the "need of achievement" the opportunity to set moderately challenging goals and take personal responsibility to achieve them. In this sense, Scratch got positive ratings because most students agreed that there have been examples that show how Scratch can be important to people who are learning programming and to people who value successfully completing activities. Moreover, most students agreed that they were able to associate the content from Unit I with things they have already seen or known.

Subsequent units using the Python Language resulted in a greater sense of relevance to students. Most agreed that Python is more challenging than Scratch. During interviews, various students assigned a more technical character to the Python language: *"At first, there was a link from Scratch to Python, and I could see that there was something more technical in Python than in Scratch."* (P35). Some interviewed students preferred Python because this language increased the sense of usefulness of what was taught, pointing out possibilities of this language as a reason: *"It is... no. Like... the unit on Python itself, I've been tripping because I know how to do amazing things. It's the Python stuff, for me, it was the best. The second unit, about practice, was the best."* (P20).

In Unit III, the results of the survey (Figure 4) show that most students disagreed that the content, contextualized with image manipulation, was relevant to their interests. This result was already expected, since the choice of tools and approach considered relevance in terms of "personal needs". Despite that, the number of students who considered Python more

useful than Scratch was large. Exploring Python in the context of media has shown flexibility of language. Some students have seen explicit value on the JES predefined functions for manipulating images: *"I liked Jython [JES] more because I'm even thinking of using Jython [JES] to manipulate images. (...) And I'm even thinking in some way of using it."* (P01).

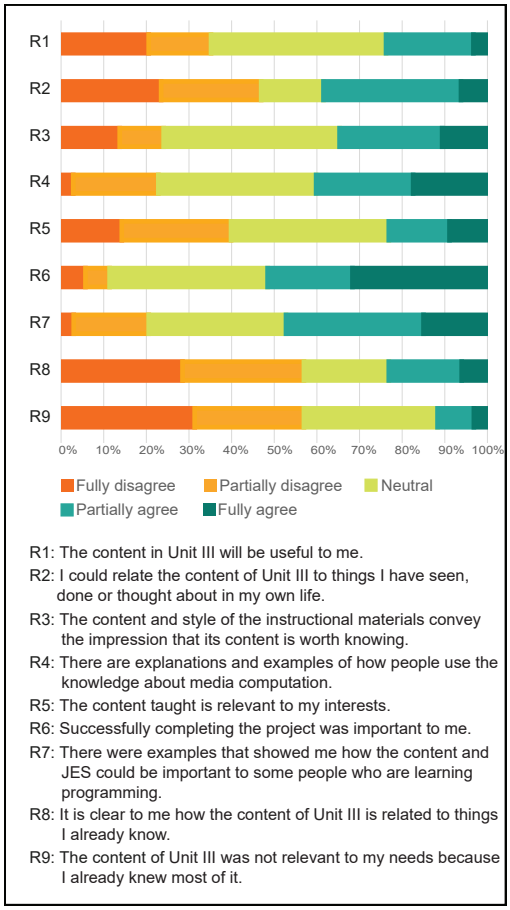

Figure 4. Relevance in Unit III.

C. Confidence

During Unit I, beginners' difficulties were minimized. As a result, confidence was enhanced. The results of the survey (Figure 5) show that most students agreed that introductory classes helped them understand what they would learn from the Scratch tool, and disagreed that the content covered and the exercises were difficult to understand. The fact that students do not feel overwhelmed during the Unit I shows Scratch makes initial programming content more accessible. Another aspect about confidence is that students need some autonomy to feel that they are capable of implementing the challenges. When interviewed about the best game built, students justify their project choices: *"it had more challenges."* (P02) or *"For having challenges ... challenging, and you can make your own game"* (P18). Furthermore, 75% of the students agreed that they have had difficulty in doing homework challenges. In the classes where examples had a greater degree of complexity, observations showed an often dispersed group.

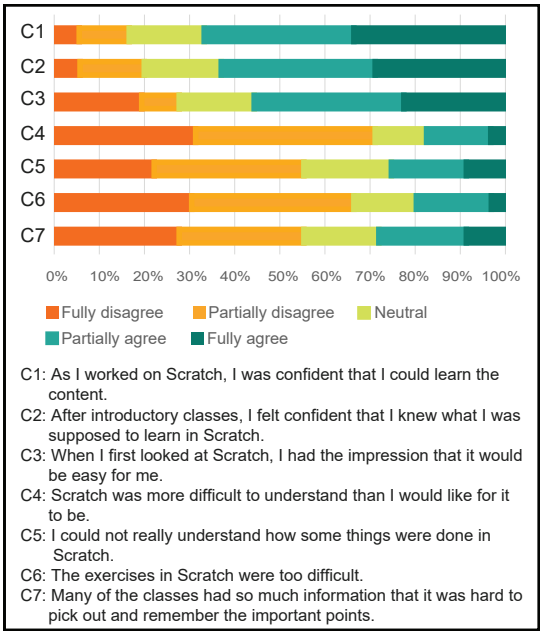

Figure 5. Confidence in Unit I.

In subsequent units, various students experienced difficulties with Python. Most students stated that they could not solve all the challenges proposed in lab sessions. During interviews, various students point out that Python is a complicated language: *"Hmmm... I found the language very complicated and I don't know... I don't know if it's for lack of practice. The challenges... like... I can't do them easily. It takes a lot to do them."* (P03). The fact that Python commands are expressed only in English, unlike Scratch, may also have contributed to increasing difficulties of various students that were not proficient in English as a second language. 45% considered English a hindrance. Difficulties with English were also mentioned: *"I found the language more difficult, especially for using the dominant language, English. That was a ... at least for me, it was a bit ... added difficulty."* (P02). Despite this negative perception, most students agreed that using a challenge guide was essential to complete the exercises.

In Unit III, the results of the survey (Figure 6) show that there was an even sharper drop in student confidence. For most, content was harder to understand than they wanted it to be: "This third unit was even worse for me. So much to learn, like... like... to study more. And it was more tiresome." (P20).

One possible explanation for the low rates of attention and confidence during units II and III is the lecture format, which had larger code examples as well as a more descriptive style. Because code was not built in class, students showed signs of feeling lost. When asked exclusively about the lab sessions, results remained positive, as in previous units. Most agreed that the project guide helped them to understand what was required, and that the project they carried out motivated the learning process. The style of lab sessions did not change from

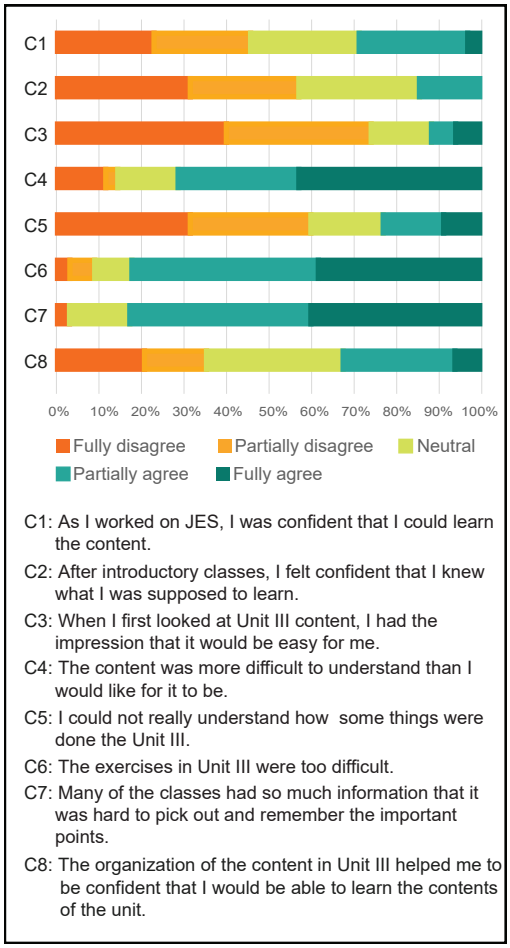

Figure 6. Confidence in Unit III.

the other units'. As students implemented the final project during labs, backed by teaching assistants and with a detailed specification guide, confidence that they would be able to finish the project was enhanced.

D. Satisfaction

Most students showed high satisfaction rates during Unit I. Results of the survey (Figure 7) show that, for most, feedback after exercises helped them feel rewarded for their effort. Moreover, most students admitted that they mentioned, outside class, that they were learning to use a tool to build games and animations. We identified that satisfaction was related to the other categories, especially with perceived relevance. The same way most students did not agree that Scratch had practical value to their lives and future expectations, few students would like to learn more about Scratch.

In subsequent units, we note that the approach potentialized satisfaction in a way similar to the unit with Scratch. Survey results (Figure 8) show that most students were satisfied with successfully completing given exercises and agreed that instructor feedback has helped them feel rewarded for their effort. About media, most students also admitted having mentioned outside class that they were learning to manipulate images, but showed no interest in learning more about images.

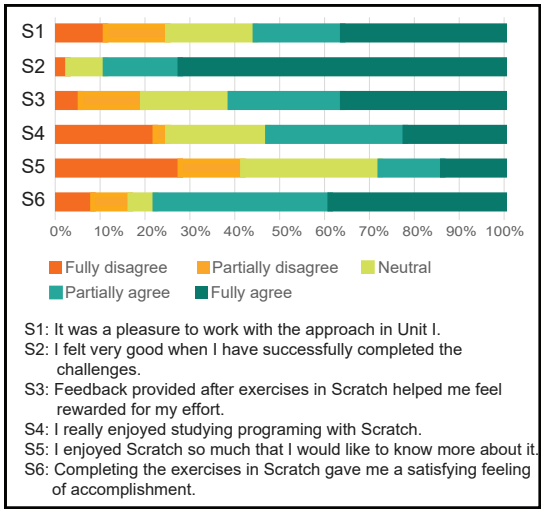

Figure 7. Satisfaction in Unit I.

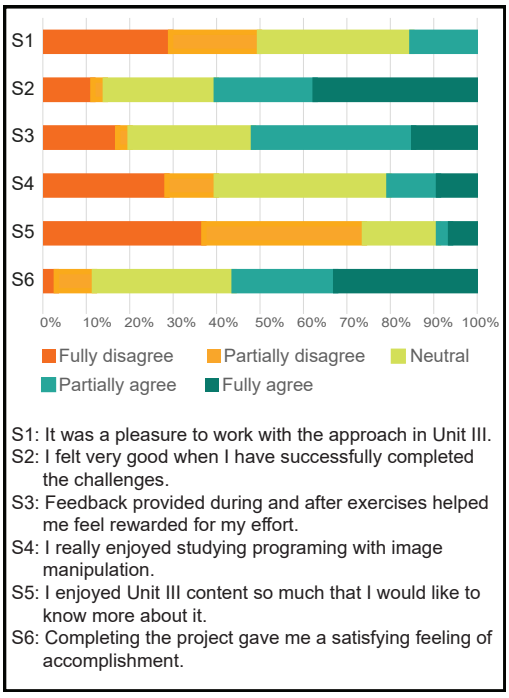

Figure 8. Satisfaction in Unit III.

Observations and interviews lead us to believe that it was less enjoyable to study with the approach of Units II and III.

V. Influences on Student Motivation

Here we try to answer our second research question: *Which are the various likely influences of this approach on student motivation?* From the results, we stress some aspects that directly influenced student motivation and, as a consequence, learning. From our observations and interviews, we coded the

positive and negative aspects of our approach and how they interfere with student motivation, describing them in Figure 9.

Languages and Tools. Results showed that using Scratch enhances students' confidence, reducing the fear of error, and this directly influences students' persistence and achievement. Cutting out the syntax errors, all the effort to learn to code focuses on developing basic logic skills and understanding basic concepts. We also noted that the introduction with Scratch facilitates the introduction to the Python language. As students already know some basic concepts, they focus their initial efforts in learning syntax details of the language. All the interviewed students were able to relate Scratch and Python: *"it helped to understand parameters, variables, functions, a bit of logical reasoning, step by step, to arrive at the solution of the problem. (...) It is ... for example, when we have to draw some figure, it is the same function, like that, just change the name, for example, move in x and backward, forward, what else ..."* (P02). On the other hand, the "childish" aspect of Scratch does not give the feeling that the content learned is relevant, while Python brings students closer to professional practice, which increases students' perceptions of relevance. However, when faced with syntax errors and other difficulties, students' attention and confidence decrease.

Contextualization. By formatting a CS1 course with the contexts of games and media, we hoped to leverage students' motivation in every aspect. Overall, students were much happier in learning how to create games than in manipulating images. Part of this impression arose because games were created with a simple tool (Scratch), while media were explored using a professional programming language. The use of games further enhanced students' confidence and satisfaction while the use of media further enhanced relevance, as it showed that the language learned has a broader application context. Even though there is a difference in how tools influence motivation, it became clear that both contexts are good because they bring programming closer to things they like to do. Some interviewees said that creating games is good because they like to play games and that it was good to learn how games are made: *''I had no idea, I had no idea what it was like to make a game. And from Scratch I saw more or less how they did it, like, I had pretty basic notions of how the little games we play are made."* (P14). The use of images is also noted as interesting: *"It's interesting because we can even work with our own photos as well. Editing them if you want to."* (P30).

Lectures. Our approach envisaged lectures that combine the use of explanations in the whiteboard, building step-by-step examples, and slide presentations. For an interviewee, *"when you are building them [during class], you will understand step by step [of example]"* (P30). During Units I and II, we were able to reach a good balance of these elements. During Unit III, we produced various slides describing curiosities and effect previews. On the other hand, due to the greater complexity of examples, the instructor decided not to build them step-by-step in class. Furthermore, when classes have a more formal character, showing implementation details in code snippets, students feel easily bored. However, we noticed that when classes show some curiosities about the topic covered (e.g., gameplay videos or image filter applications), even if the content is complex, attention is maintained because curiosity was aroused. The real challenge lies in sustaining students 'attention. To do so, instructors need to respond to students' needs of looking for new sensations, and to arouse their curiosity to search for knowledge without overestimating them [14]. The sense of relevance can also be stimulated if explanations are contextualized with useful everyday applications, such as the use of images on social networks.

Labs. We noted that our lab format enhances learning and respects students' own paces without underestimating them. Since lab sessions allowed practicing contents from the lectures, students could train their skills with support of the instructors. Some students felt confident during lab sessions when they were given prior instructions: *"I feel confident ... when I'm prepared, based on a previous lecture, right? Oh, I think I'm ready to do the lab."* (P30). In a challenge-based lab format, students are encouraged with the assurance that, when they face difficulties, they are supported by instructors. This format was interesting because it supports both those who are more dependent on instruction and students who like to be challenged: *"So ... in every lab, I defied myself. I rarely asked, asked the assistants or the prof how to do it, because I challenged myself by trying to do it myself. (...)"* (P06).

Instructional Materials. An emergent aspect of our observations is that the attention level was higher in classes that used visually organized and flashy slides, with curiosities on the subject. About the materials used, we highlight the challenge guide given to students during labs. In it, each game or exercise was split into small challenges ordered by difficulty level. For students, it directed implementation: *"Excellent. (...) Because, willing or not, it gives a way. Like... you don't start from scratch."* (P20); *"I think it's cool, because the exercise guide starts from the most basic to the most advanced. And with that, like... you evolve over the course of the lesson."* (P09).

Practical Assessment. Lab exams were the main way to assess acquired competences during the Units I and II. Grading was done right after the end of the exam, and students immediately received feedback on their performance. Operationally, this form of assessment takes time and is unfeasible when there are no assistants helping the instructor. Nonetheless, interviewees' opinions were unanimous: *"Grading immediately, for me, it's wonderful. Especially when you, like... learn ... She'll tell you your mistakes. Then you will learn more. If everyone did it like that, it would be much better."* (P20).

Instructor's Attitudes. In our approach, classes should have a less formal character and the instructor should be closer to students, showing interest in their feedback. From our observations and interviews, we identified both positive and negative instructor's attitudes that influence motivation. We noticed that students are more attentive in lectures when the instructor periodically reinforces the learning goals and the importance of certain concepts. By taking an encouraging stance, fostering students to challenge themselves, the instructor encourages their confidence without underestimating them.

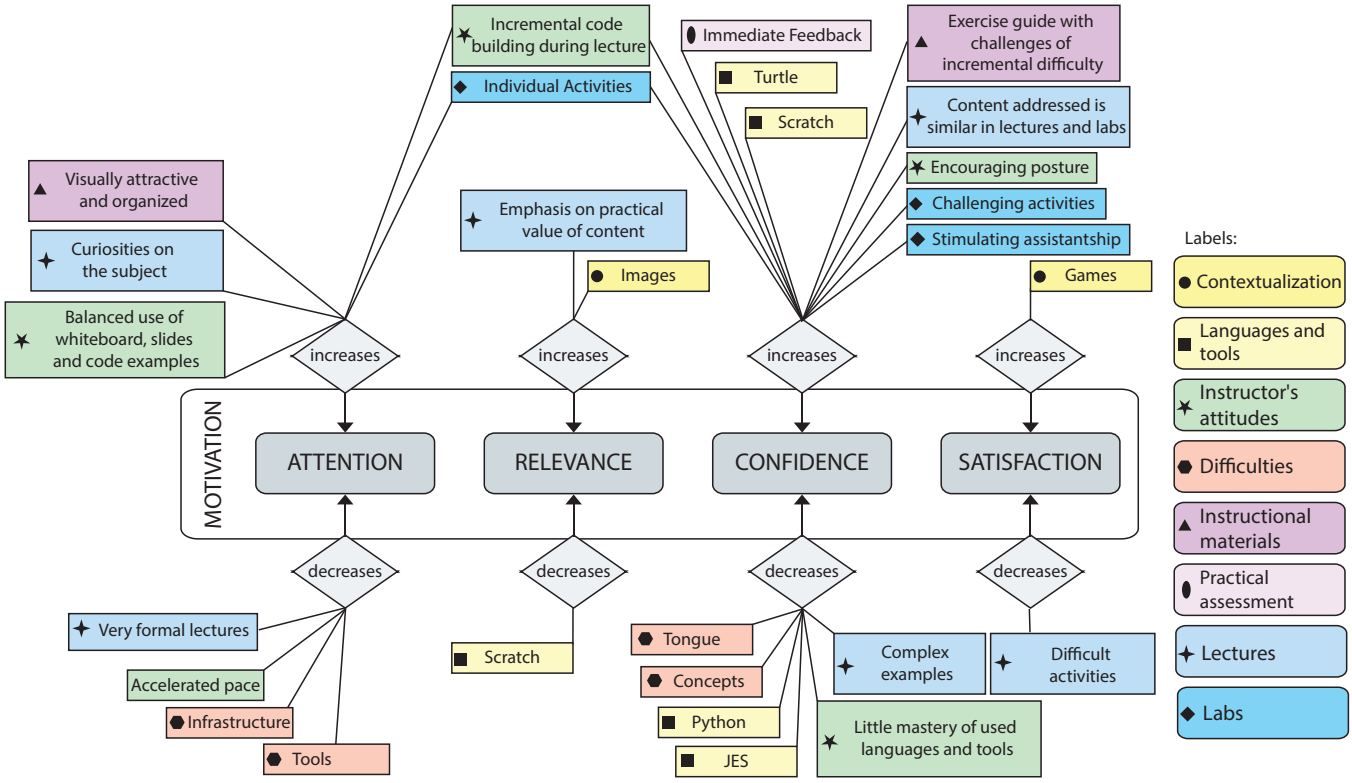

Figure 9. Aspects of our approach and how they interfere with student motivation

When the instructor explores different instructional practices, alternating between explanations and coding, class attention is captured. On the other hand, when the instructor has a fast pace of instruction, or does not build examples in class, it is likely that students with greater difficulties will not pay attention and neither will make an effort to implement challenges.

Difficulties. From observations, we identified four types of most frequent difficulties: concepts, tongue, tools and infrastructure. Conceptual difficulties refer to programming concepts. Difficulties with English as a second language (most students were not proficient in English), quoted at some point in the course by more than 40% of the students, were the most unexpected discovery. When facing this type of difficulty, students' confidence is tested. Confident people tend to attribute the causes of success to aspects like skills and effort rather than luck or task difficulty [14]. Insecure apprentices, on the other hand, worry a lot about failure. The greater the difficulties, the weaker is the impression that some level of success is possible if the effort is performed, which gives the impression that *"this is not for me, I wasn't born for this"* (P14). Difficulties with tools are those where students perceive some bug or tool limitation, while infrastructure difficulties are related to computer failure or organizational issues that hinder the progress of the lesson. In our observations, we noted that whenever one of these difficulties arises, attention in class is lost, and it is hardly resumed afterwards.

VI. Conclusions

Here we presented a teaching approach to introductory programming framed to motivate students and lessen initial difficulties of CS non-majors. Our approach uses of a mix of languages and tools already tested in CS1 teaching, combining the use of Scratch with creation of games, and Python with a screen turtle and image manipulation. To evaluate our approach, we conducted an exploratory, mixed-methods, case study with a a CS1 class for civil engineering freshmen.

From the analysis of each category of the ARCS model of motivation, we identified how general aspects of our approach influence student motivation, both positively and negatively. We found that, by using Scratch in the first classes of the course, student confidence is enhanced. By introducing Python using examples that can be implemented in both Scratch and Python, students can feel that they already have some knowledge of programming concepts and focus their efforts on learning Python syntax. We found that our lab session format, with challenge-led activities of incremental difficulty and immediate feedback positively influence student motivation.

Those results are exploratory, but provide preliminary evidence suggesting the effectiveness of the approach. Although we do not mean to generalize, we believe several findings are common to other approaches different from ours. We intend to report another case study with a second group of students with background similar to the ones in this work, measuring both motivation and learning of programming skills.

ACKNOWLEDGMENT

This project was supported by FAPESB – Foundation for Research Aid of the State of Bahia, in the form of a scholarship for master studies.

The authors would like to thank both the students ans faculty members from the State University of Feira de Santana (UEFS) that voluntarily took part in this research.

REFERENCES

[1] H. M. Walker, "Computational thinking in a non-majors cs course requires a programming component," *ACM Inroads*, vol. 6, no. 1, pp. 58–61, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2727126

[2] S. Cooper and W. Dann, "Programming: A key component of computational thinking in cs courses for non-majors," *ACM Inroads*, vol. 6, no. 1, pp. 50–54, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2723169

[3] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4. Citeseer, 2002, pp. 53–58.

[4] A. 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, "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.

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

[7] R. J. Wlodkowski, *Motivation and teaching: A practical guide.* ERIC, 1978.

[8] J. R. Baird and R. T. White, "Promoting self-control of learning," *Instructional Science*, vol. 11, no. 3, pp. 227–247, Dec 1982. [Online]. Available: https://doi.org/10.1007/BF00414281

[9] J. Kuhl, *Volitional Mediators of Cognition-Behavior Consistency: Self-Regulatory Processes and Action Versus State Orientation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 101–128. [Online]. Available: https://doi.org/10.1007/978-3-642-69746-3_6

[10] M. J. Lee, "Effects of different loci of instructional control on students' metacognition and cognition: Learner vs. program control," 02 1990, unpublished report. [Online]. Available: https://eric.ed.gov/?id=ED323938

[11] B. J. Zimmerman, "A social cognitive view of self-regulated academic learning." *Journal of educational psychology*, vol. 81, no. 3, p. 329, 1989.

[12] R. M. Steers and L. W. Porter, *Motivation and work behavior*, 3rd ed. New York : McGraw-Hill, 1983, includes bibliographies.

[13] V. H. Vroom, *Work and motivation*, 1st ed. San Francisco : Jossey-Bass Publishers, 1995, originally published: New York : Wiley, 1964.

[14] J. M. Keller, "Development and use of the arcs model of instructional design," *Journal of instructional development*, vol. 10, no. 3, pp. 2–10, 1987.

[15] 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

[16] W. Huang, W. Huang, H. Diefes-Dux, and P. K. Imbrie, "A preliminary validation of attention, relevance, confidence and satisfaction model-based instructional material motivational survey in a computer-based tutorial setting," *British Journal of Educational Technology*, vol. 37, no. 2, pp. 243–259, 2006.

[17] M. Hamada, "An integrated virtual environment for active and collaborative e-learning in theory of computation," *IEEE Transactions on Learning Technologies*, vol. 1, no. 2, pp. 117–130, 2008.

[18] R. Savi, C. G. von Wangenheim, and A. F. Borgatto, "A model for the evaluation of educational games for teaching software engineering," in *Software Engineering (SBES), 2011 25th Brazilian Symposium on*. IEEE, 2011, pp. 194–203.

[19] C. G. von Wangenheim, R. Savi, and A. F. Borgatto, "Deliver!–an educational game for teaching earned value management in computing courses," *Information and software Technology*, vol. 54, no. 3, pp. 286–298, 2012.

[20] 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.

[21] 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 *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*. IEEE, 2015, pp. 1–9.

[22] M. Dorling and D. White, "Scratch: A way to logo and python," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 2015, pp. 191–196.

[23] 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.