

Didactics of the Fundamentals of Computer Programming with a Computational Thinking Approach Based on Turtle Graphs*

Jesús Insuasti^a ■ Edwin Insuasty^b ■ Alexander Baron^c

Abstract: Computer programming fundamentals often present significant challenges, particularly in developing computational thinking and problem-solving skills. In Colombia, several higher education institutions have reported high dropout rates in introductory programming courses, according to government data. In response to this issue, a qualitative study with quantitative components is proposed, adopting a case study approach aimed at enhancing the teaching of the “Computer Programming Fundamentals” course at the University of Nariño, in Pasto, southern Colombia. The research introduces the use of Turtle computational graphics through Flowgorithm, a tool originally proposed by Sacramento State University in California. Widely used in the 1980s, Turtle graphics provide a unique and engaging method for teaching computational thinking through programming. By translating code into visual movements and drawings, this approach helps bridge the gap between abstract programming concepts and tangible understanding—particularly beneficial for beginners. Flowgorithm’s implementation of Turtle graphics introduces essential programming concepts in a fun and interactive environment. This method fosters greater confidence in novice programmers and nurtures deeper interest in the subject, thereby significantly improving learning outcomes. In a study involving sixty-six students enrolled in the course, the approach yielded promising results. Students were able to immediately observe the effects of their code, promoting experimentation, creativity, and visual evaluation of their algorithms—key factors in developing strong problem-solving skills.

Keywords: Computational Thinking; Turtle Graphics; Didactics; Fundamentals; Programming

* Research Article

- a PhD in Engineering – Systems and Informatics. National University of Colombia, Medellín campus). Email: insuasti@udenar.edu.co; ORCID: <https://orcid.org/0000-0002-3233-3577>
- b PhD in Educational Processes in Virtual Environments. University of Nariño, Pasto, Colombia. Email: edwin@udenar.edu.co; ORCID: <https://orcid.org/0000-0002-5844-9949>
- c PhD in Engineering – Systems and Informatics. National University of Colombia, Medellín campus. Email: abaron_98@udenar.edu.co; ORCID: <https://orcid.org/0000-0001-6425-8041>

Recibido: 21/08/2024 **Aceptado:** día, mes y año **Disponible en línea:** 09/08/2025

Cómo citar: Insuasti, J. Insuasty, E. y Baron, A. (2025). Didactics of the Fundamentals of Computer Programming with a Computational Thinking Approach Based on Turtle Graphs. *Academia y Virtualidad*, 18(2), 65–80. <https://doi.org/10.18359/ravi.7533>

Didáctica de los fundamentos de la programación de computadoras con un enfoque de pensamiento computacional basado en Grafos de Tortuga

Resumen: Los fundamentos de programación de computadoras a menudo presentan desafíos significativos, especialmente en el desarrollo del pensamiento computacional y las habilidades para resolver problemas. En Colombia, algunas instituciones de educación superior han experimentado altas tasas de deserción en cursos introductorios de programación, según datos gubernamentales. Por lo tanto, se propone una investigación cualitativa con complementos cuantitativos, adoptando un enfoque de estudio de caso, que tiene como objetivo contribuir a la enseñanza del curso de fundamentos de programación de computadoras en la Universidad de Nariño en Pasto, al sur de Colombia. La investigación propone utilizar gráficos computacionales de Tortuga usando Flowgorithm como herramienta base, propuesto por la Universidad Estatal de Sacramento en California. Los gráficos de Tortuga, ampliamente utilizados en la década de 1980, ofrecen una forma única y atractiva de enseñar el pensamiento computacional basado en la programación de computadoras, lo cual es particularmente beneficioso para principiantes. Al traducir el código en movimientos y dibujos visuales, los gráficos de Tortuga se reconocen con un enfoque actualizado para acercar y simplificar los conceptos abstractos de programación. Esta representación visual ayuda a desarrollar habilidades de pensamiento computacional y resolución de problemas. Los gráficos de Tortuga en Flowgorithm también introducen brevemente conceptos esenciales de programación en un entorno divertido e interactivo. Este enfoque aumenta la confianza en los nuevos programadores y fomenta un interés más profundo en el campo, mejorando significativamente el aprendizaje. Esta enseñanza ha mostrado resultados prometedores en un estudio que involucró a sesenta y seis estudiantes que asistieron al curso, destacando que pueden ver inmediatamente los efectos de su código mediante la experimentación y la creatividad, lo que les permite evaluar y refinar sus algoritmos de manera visual.

Palabras clave: pensamiento computacional; gráficos de tortuga; didácticas; fundamentos; programación

Didática dos fundamentos da programação de computadores com uma abordagem de pensamento computacional baseada em Gráficos de Tartaruga

Resumo: Os fundamentos da programação de computadores frequentemente apresentam desafios significativos, especialmente no desenvolvimento do pensamento computacional e das habilidades de resolução de problemas. Na Colômbia, algumas instituições de ensino superior têm enfrentado altas taxas de evasão em cursos introdutórios de programação, segundo dados governamentais. Por isso, propõe-se uma pesquisa qualitativa com complementos quantitativos, adotando uma abordagem de estudo de caso, com o objetivo de contribuir para o ensino do curso de fundamentos da programação de computadores na Universidade de Nariño, em Pasto, no sul da Colômbia. A pesquisa propõe utilizar gráficos computacionais de Tartaruga com o uso do Flowgorithm como ferramenta base, proposto pela Universidade Estadual de Sacramento, na Califórnia. Os gráficos de Tartaruga, amplamente utilizados na década de 1980, oferecem uma forma única e atrativa de ensinar o pensamento computacional com base na programação de computadores, sendo particularmente benéficos para iniciantes. Ao traduzir o código em movimentos e desenhos visuais, os gráficos de Tartaruga são reconhecidos como uma abordagem atualizada para aproximar e simplificar conceitos abstratos da programação. Essa representação visual ajuda no desenvolvimento de habilidades de pensamento computacional e resolução de problemas. Os gráficos de Tartaruga no Flowgorithm também introduzem brevemente conceitos essenciais da programação em um ambiente divertido e interativo. Essa abordagem aumenta a confiança dos novos programadores e estimula um

interesse mais profundo na área, melhorando significativamente o aprendizado. Esse método de ensino mostrou resultados promissores em um estudo que envolveu sessenta e seis estudantes participantes do curso, destacando que eles podem ver imediatamente os efeitos de seu código por meio da experimentação e criatividade, o que lhes permite avaliar e refinar seus algoritmos visualmente.

Palavras-chave: pensamento computacional; gráficos de tartaruga; didática; fundamentos; programação

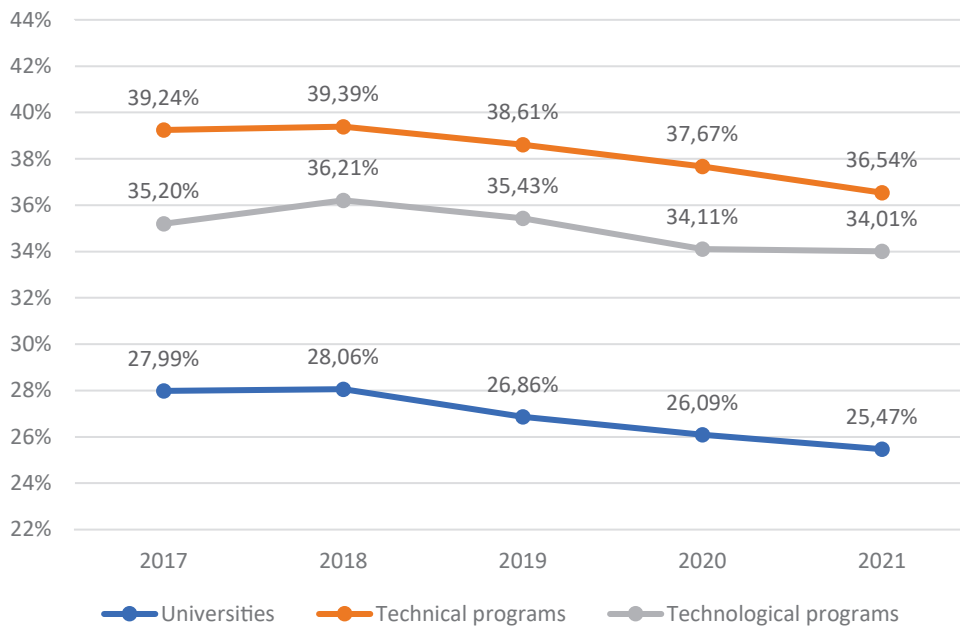
Introduction

Learning computer programming can be particularly challenging, especially for beginners. The process demands strong computational thinking and problem-solving skills, which are essential for effective coding. In Colombia—much like in many parts of the world—educational and resource limitations often intensify these challenges, making it

especially difficult for students who are just beginning to explore the field.

Introductory programming courses, or courses in computer programming fundamentals, have historically faced high attrition rates, particularly in regions such as Colombia. Government data from Colombia underscores this issue, revealing a concerning trend across higher education institutions, as illustrated in *Figure 1*.

Figure 1. Dropout rate in technical, technological, and university careers (MEN, 2022)



Source: Own elaboration.

Learning computer programming can be a significant challenge, especially for beginners. The process requires strong computational thinking and problem-solving skills, both of which are critical for successful coding. In Colombia, as in many other parts of the world, limitations in educational resources often exacerbate these challenges, creating substantial barriers for students just beginning their journey into programming.

Introductory programming courses, or courses focused on the fundamentals of computer programming, have historically shown high attrition rates—particularly in regions like Colombia. Government data highlights this issue, revealing a concerning trend across higher education institutions, as illustrated in *Figure 1*.

These high dropout rates suggest that traditional teaching methods may not effectively address students' learning needs in these contexts. At the University of Nariño, our research adopts a case study approach to explore alternative teaching strategies aimed at improving students' comprehension of programming fundamentals. Through a mixed-methods research design, we provide in-depth insights into how pedagogical changes can influence student engagement and retention.

Turtle graphics—a tool originally popularized in the 1980s—have been revitalized in our study as a means of supporting programming instruction. Using Flowgorithm, a visual programming environment developed at California State University, Sacramento, we have reintroduced this

graphical tool to help beginners visualize coding concepts through interactive and engaging visual representations.

Turtle graphics offer a unique approach to learning programming by converting abstract code into immediate, visible outcomes. This visual feedback makes complex concepts more accessible and easier to grasp, helping students develop a stronger understanding of programming logic. The visual and interactive nature of Turtle graphics enhances the learning experience, making it both tangible and enjoyable.

Our case study involved 66 students who experienced the benefits of this alternative method firsthand. By allowing them to experiment and observe the outcomes of their code, Turtle graphics increased students' confidence and fostered a deeper interest in computing. Feedback from participants confirmed that this approach made learning more engaging and effective.

This article is structured as follows: *Section 2* reviews the relevant literature. *Section 3* describes the methods and materials used. *Section 4* presents the results and discussion. *Section 5* concludes with key findings and suggests directions for future research.

Literature review

In higher education, dropout rates represent a major challenge, influencing not only individual outcomes but also broader socioeconomic development. Recent studies indicate that demographic factors play a key role in shaping these trends. For example, data from the American Community Survey (U.S. Department of Commerce, Census Bureau, 2021) show that individuals with disabilities have a dropout rate of 10.4%, compared to 4.8% among those without disabilities.

Additionally, race and ethnicity are significant predictors of dropout rates. Pacific Islanders and Hispanics, for instance, show higher-than-average rates of educational attrition. These disparities reflect a complex interplay of social, economic, and cultural factors, emphasizing the importance of tailored interventions that address specific community needs. Understanding these nuances is

crucial for developing effective education policies aimed at reducing dropout rates and improving academic success.

In Colombia, trends in higher education dropout rates mirror those in the United States. While there has been a gradual decline, the rates remain high. According to Colombia's Ministry of National Education (MEN, 2022), the average dropout rate in universities stands at 25.47%, while technical and technological programs report even higher rates—36.54% and 34.01%, respectively. These figures underscore a persistent challenge that demands targeted interventions across all levels of higher education.

In computer science education, dropout rates are closely linked to students' proficiency in basic mathematics and programming. A lack of foundational knowledge in these areas often leads to poor academic performance and early withdrawal from programs. This correlation highlights the need for better preparatory training in schools to equip students with the skills necessary for success in higher education (Insuasti, 2016).

Today, numerous products are available to teach children STEM and computer science using programming robots—a practice that began over four decades ago. The concept dates back even further, to the 1960s, when mathematician and educator Seymour Papert introduced the first STEM educational robots: Logo Turtles. These tools were designed to help students tackle mathematical and programming challenges by transforming abstract concepts into hands-on experiences.

Due to the abstract nature of programming, there has been a growing focus on revitalizing computational thinking through visual tools like Turtle graphics. For instance, Universiti Kebangsaan Malaysia incorporated Turtle Graphics into its Java programming course, using animation and graphics to increase student engagement and motivation (Bakar *et al.*, 2020).

With the recent inclusion of computer science in primary school curricula, evaluating diverse student solutions has become increasingly complex. A new tool has been developed to facilitate the creation and evaluation of custom programming assignments using Turtle graphics. This

system, implemented in the XLogoOnline environment, compares student solutions pixel-by-pixel with sample outputs, streamlining the grading process (Marbach *et al.*, 2022).

Laura-Ochoa and Bedregal-Alpaca (2022) explored the integration of computational thinking in a first-year Python course, using tools such as PSeInt, CodingBat, and the Turtle graphics library. Their findings showed that students exposed to these tools demonstrated greater learning gains and improved motivation—suggesting that visual and interactive environments significantly enhance student outcomes in programming education.

Turtle graphics and similar “microworld” environments have long been integral to programming education strategies. Kay and Tyler (1992) developed one such microworld featuring a robot named Darel, which enabled students to explore algorithmic thinking through simple, visual tasks. The environment encouraged the use of predicates to define program states and goals, helping students better conceptualize abstract programming constructs.

Block-based programming environments have also gained popularity as entry points for novice coders. Programs like the Hour of Code have introduced millions of students worldwide to coding through interactive challenges. Pelánek and Effenberger (2020) analyzed these environments, identifying key characteristics of successful microworlds and offering design guidelines to optimize educational outcomes.

Kiper and Bishop-Clark (1995) advocated for the integration of computer science within humanities curricula, emphasizing accessibility for students from all disciplines. Their course design—centered on abstraction, design, and theoretical foundations—aimed to demystify computing and shift students’ perceptions from anxiety to appreciation. Evaluations showed that such interdisciplinary approaches can effectively engage diverse learners and foster deeper understanding.

Cooper, Dann, and Pausch (2003) introduced a beginner-level programming course using 3D interactive animation. This approach prepared students for traditional CS1 courses by teaching object-oriented concepts in an engaging, visual

format. Their research highlighted improvements in problem-solving skills and student engagement, supporting the use of motion graphics to bridge the gap between novice and advanced programming instruction.

Method and materials

This study employs mixed-methods research design, integrating both qualitative and quantitative approaches to provide a comprehensive understanding of the research problem (Creswell, 2014). By combining these methods, we explore the complex phenomenon of teaching programming fundamentals from multiple perspectives. The qualitative component offers deep insights into participants’ experiences, while the quantitative aspect enables the analysis of patterns and the generalization of findings across a broader population (Tashakkori and Teddlie, 2004).

We adopted a concurrent exploration design, as described by Creswell and Plano Clark (2011), in which qualitative and quantitative data are collected simultaneously and integrated during the interpretation phase. This design was selected because it allows for immediate data triangulation, providing a richer context to assess the effectiveness of teaching strategies and computational thinking in programming education.

The quantitative phase of the study involved surveying students to assess their understanding of basic programming concepts following the implementation of instructional guides. Descriptive statistics were used to analyze the data, with a focus on frequency distributions and percentage calculations to identify response patterns. Additionally, inferential statistical tests, such as the chi-square test for independence, were conducted to examine the relationship between teaching methods and student performance.

For the qualitative component, we conducted semi-structured interviews and classroom observations to capture students’ perspectives on the instructional methods used. A thematic analysis was then performed to identify recurring themes related to learning experiences and the perceived impact of computational thinking on problem-solving skills. This approach enabled a better

understanding of individual learning processes and the challenges students face in mastering programming fundamentals.

The study was conducted during the first semester of 2024 and involved 66 students enrolled in the introductory computer programming course of the Systems Engineering program at the University of Nariño, in Pasto, Colombia. Convenience sampling was used to select the sample, given the practical constraints of working with a specific group of students during the course's implementation. This approach was deemed appropriate to gather initial insights into instructional strategies within a real-world educational setting.

In educational research, using an entire course as both the population and the sample is a common practice—especially when the group represents a well-defined cohort within a specific academic context. When the number of participants is manageable and all individuals are accessible, this approach is justified, as it enables a comprehensive analysis without introducing bias through sampling techniques (Creswell, 2014).

By considering the entire class as the population, the study captures the full range of student experiences, learning behaviors, and outcomes, offering a holistic understanding of the instructional approach's effectiveness. In cases where the research seeks to evaluate a particular pedagogical strategy in a controlled academic environment, involving all enrolled students ensures that the findings are directly applicable to similar educational contexts (Fraenkel *et al.*, 2019).

Given these conditions, using the entire course as the sample is methodologically sound and enhances the study's validity and reliability by reducing sampling errors and supporting more robust statistical analysis (Gall *et al.*, 2007).

Regarding ethical considerations, all participants were informed about the purpose of the study, and their consent was obtained prior to data collection. Confidentiality and anonymity were maintained throughout the research process, in accordance with the ethical guidelines outlined by the American Educational Research Association (AERA, 2011).

Computational thinking (CT) was a key element of our instructional approach, recognized for enhancing critical thinking and problem-solving skills in higher education (Wing, 2006). We incorporated CT into our instructional guides, emphasizing essential concepts such as constants, variables, arithmetic and logical expressions, comparisons, and loops—using Turtle graphics to facilitate practical application. The aim was to offer students a structured approach to solving problems in programming, aligned with CT principles (Barr and Stephenson, 2011).

The case study methodology was employed to apply the instructional guides within a real-world classroom context. Students' progress was closely monitored throughout this phase, and their feedback was collected regarding their learning experiences. The results were analyzed to evaluate the effectiveness of the instructional guides and the pedagogical approaches implemented. This iterative process contributed to refining our teaching methods and deepening our understanding of how computational thinking can be successfully integrated into programming education.

Within the qualitative phase, we adopted an exploratory research design to investigate the underlying patterns, behaviors, and challenges in teaching programming fundamentals. Exploratory research is especially suitable in areas with limited prior knowledge, as it allows researchers to uncover new insights and generate hypotheses for future inquiry (Stebbins, 2001). This approach aimed to understand students' learning experiences, attitudes toward computational thinking, and contextual factors influencing their engagement with programming concepts. Characterized by its flexibility and open-ended nature, exploratory research is ideal for qualitative inquiry in educational settings (Babbie, 2016).

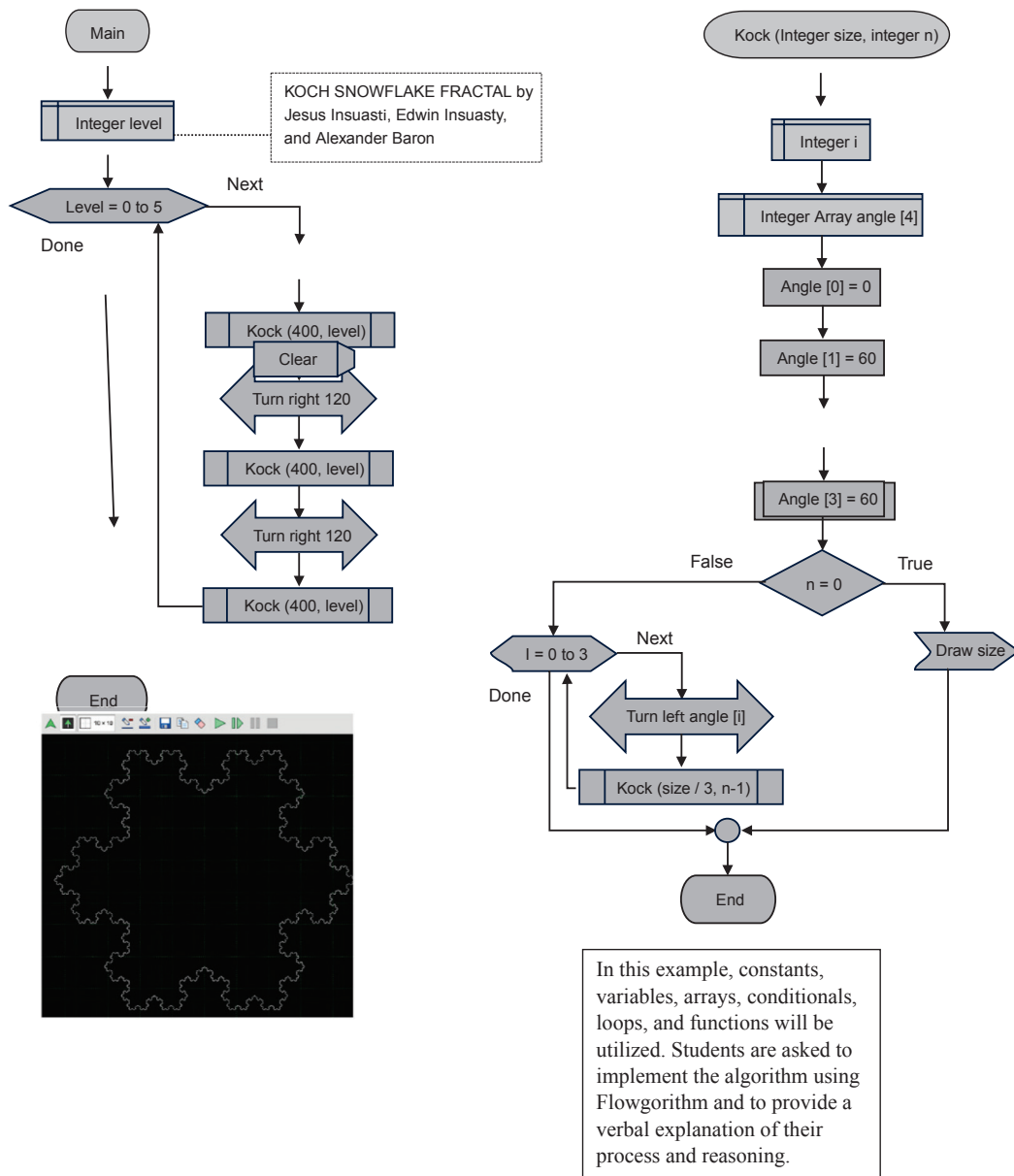
Through interviews and classroom observations, we sought a comprehensive view of the dynamics of teaching and learning in programming courses. This approach enabled the adaptation of the research process based on emerging findings, enhancing the richness and relevance of the data collected (Swedberg, 2020).

Results and discussion

After implementing study guides designed to introduce Turtle graphics using Flowgorithm—a programming fundamentals tool developed by Davin Cook at Sacramento State University (available at <http://www.flowgorithm.org/>)—students began completing programming exercises in a pair programming format. This collaborative learning strategy fostered an interactive environment that

enhanced students' understanding of programming concepts. *Figure 2* in the study presents excerpts from the instructional guides, while *Figure 3* provides photographic evidence of students engaging with the material during the case study. This hands-on application enabled the integration of theory and practice, allowing both students and the instructor to deepen their understanding of programming in real time.

Figure 2. An excerpt from the guidelines for using Turtle Charts in Flowgorithm



Source: Own elaboration.

Pair programming, a collaborative approach where two students work together on a computer, offers significant pedagogical benefits. Firstly, it encourages active engagement and participation, as students must communicate and collaborate effectively to solve problems and write code together. This not only enhances their technical skills but also promotes teamwork and communication, both of which are crucial in real-world scenarios (Wei *et al.*, 2021).

Moreover, pair programming fosters knowledge sharing and peer learning as students exchange ideas, approaches, and solutions, enriching each other's understanding of the material. Workshop orientation guides within computer labs complement this method by providing structured instructions and support, ensuring that students

remain focused and make the most of their collaborative efforts.

These guides act as roadmaps, facilitating the learning process and reinforcing key concepts, which in turn enhances the effectiveness of pair programming as an educational tool. When combined with mentoring guides, pair programming helps develop essential technical and interpersonal skills, preparing students for success in both academic and professional environments (Becker *et al.*, 2023).

Students were asked to participate in a survey after engaging in Turtle Charts. The survey included a section where participants acknowledged and signed an informed consent form, which was an essential part of the research process. The survey followed a structured format, as illustrated from Figure 4 to Figure 6.

Figure 3. Preparation of guides and exercises in a computer laboratory



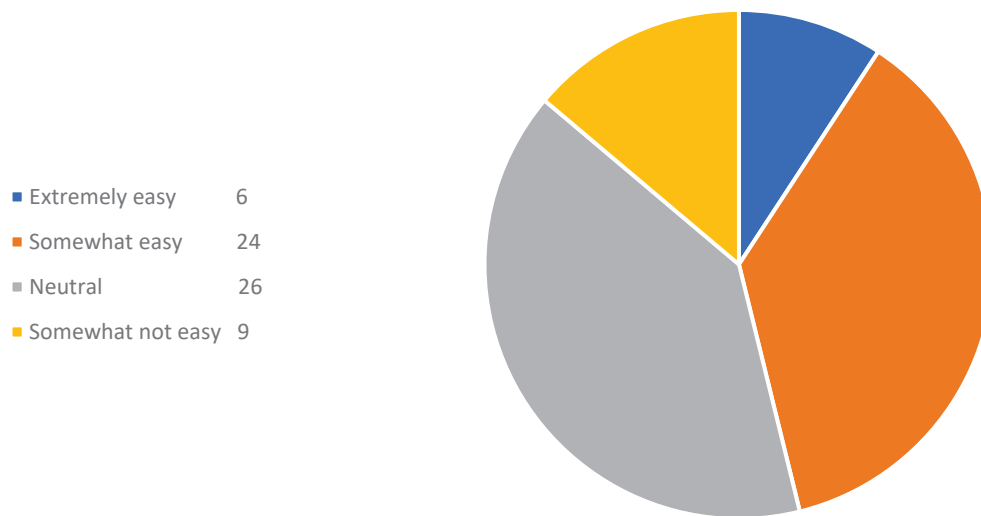
Source: Own elaboration.

Figure 4. Survey results in understanding programming concepts using Turtle Charts

Thirty-seven percent of respondents indicated that they found the question "Somewhat easy," while the majority selected "High" for Question 10.



How easy was understanding the programming concepts using Turtle Graphics?



Source: Own elaboration.

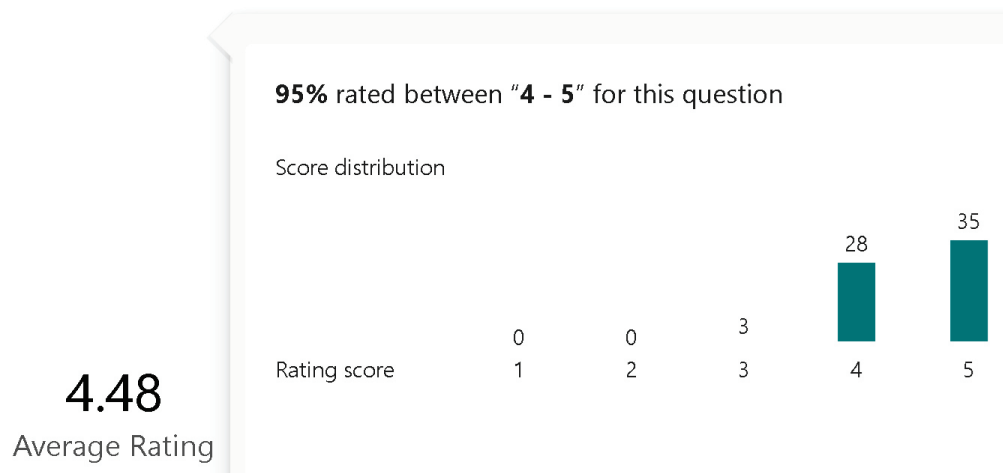
The pie chart in *Figure 4*, illustrates responses to a survey question about the ease of understanding programming concepts using Turtle charts. Most of the 26 participants reported a neutral stance, suggesting they found it unchallenging. A considerable number of 24 respondents found it easy, indicating that Turtle graphics are beneficial, although challenging for students.

Those who found it extremely easy are significantly fewer, totaling 6. This could imply that Turtle charts are the most intuitive method for

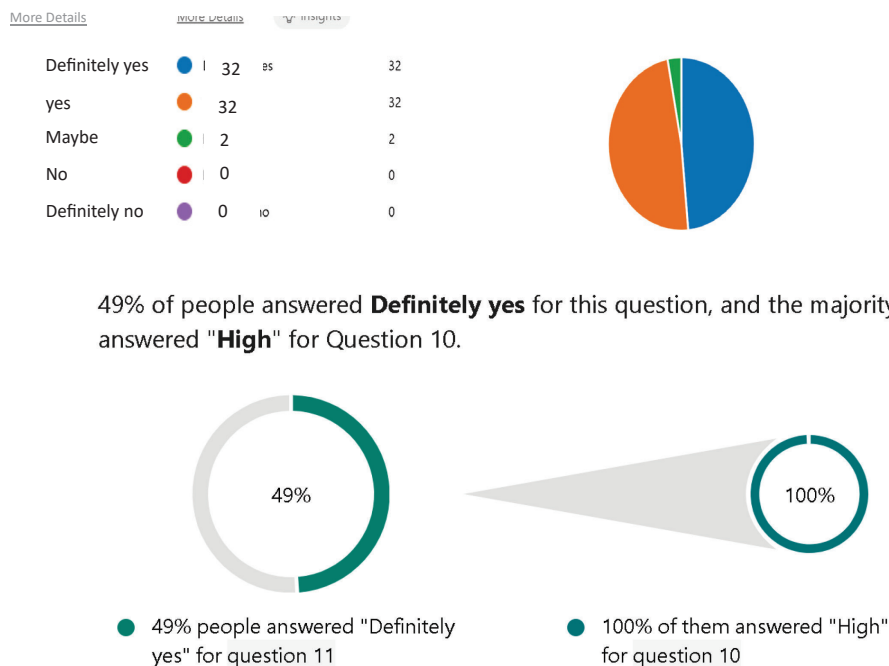
understanding the fundamentals of computer programming. On the more challenging side, nine participants rated it as not easy, and one individual found it exceptionally difficult, highlighting that some students need help with this approach. This diversity in responses could suggest that while Turtle Charts can be an effective educational tool, its effectiveness may vary widely among students and must be supplemented with additional resources or teaching methods to accommodate different learning styles.

Figure 5. Survey results on the use of Turtle Charts and recommendation attitude

10. How engaging was the use of Turtle Graphics in learning programming?



11. Would you recommend using Turtle Graphics to other students learning programming?



Source: Own elaboration. Figure 5 comes from NVivo 15.

The bar graph data in *Figure 5*, illustrates students' engagement levels when using Turtle Graphics to learn programming fundamentals. Ninety-five percent of respondents rated their engagement as high, giving scores of 4 or 5 out of 5, indicating a positive reception. The average rating

was 4.48, suggesting that most students found Turtle Graphics captivating. The score distribution supports this sentiment, with 35 respondents giving it the maximum score of 5, and 28 rating it a 4. This demonstrates that Turtle Graphics successfully maintained a high level of interest among

students. The absence of ratings 1 and 2 further emphasizes the success of Turtle Graphics in engaging students. Only a small number of three participants gave it a 3, suggesting that while the method was generally well-received, a few students may have been less enthusiastic. These findings imply that Turtle Graphics, is a desirable and potentially beneficial approach for teaching programming, fostering a positive learning environment and sparking greater interest in the subject.

Furthermore, the pie chart in Figure 4, summarizes the recommendations for Turtle Graphics as a programming learning tool from the group of respondents. A convincing majority, with 32 respondents each, answered “Definitely Yes” and “Yes,” which makes up the bulk of the chart. This overwhelming support reflects a strong consensus on the effectiveness of Turtle Graphics in teaching programming concepts. Additionally, a small fraction, represented by only two respondents, answered “Maybe,” indicating some reservations about the tool’s universal applicability. Notably, no respondents answered “No” or “Definitely not,” suggesting there were no strong objections to the use of Turtle Graphics among participants. The absence of negative responses could indicate a favorable opinion of Turtle Graphics in the educational context, reinforcing its value as a helpful resource for students beginning to learn programming.

Statistical Analysis and Inference

This section presents an in-depth analysis of the data gathered from students’ feedback regarding the use of Turtle Graphics for teaching programming concepts. The analysis is divided into descriptive statistics, which summarize the students’ responses, and inferential statistics, which explore the relationships between ease of understanding and overall satisfaction with the teaching approach. The statistical analysis also assesses the significance of the results by calculating confidence intervals and performing a chi-square test for independence.

The first step in analyzing the data was to examine how students perceived the ease of understanding programming concepts using Turtle Graphics. Most participants rated the method

positively, with 36.4% (24 students) indicating that the concepts were “Somewhat easy” to understand and 39.4% (26 students) expressing a “Neutral” response. A smaller subset of students found the approach to be “Extremely easy” (6 students), suggesting that while many found the method accessible, there were variations in student experiences. On the other end of the spectrum, nine students (13.6%) rated the method as “Somewhat not easy,” and one student found it “Extremely not easy,” indicating some challenges in comprehending the material.

The average rating for the overall effectiveness of the approach was 4.48 on a scale from 1 to 5, with 95% of students assigning a score between 4 and 5. This high average indicates that most students found using Turtle Graphics beneficial in enhancing their understanding of programming concepts. Furthermore, when asked whether they would recommend the approach or affirm its usefulness, an overwhelming 97% of respondents (64 out of 66) answered “Definitely yes” or “Yes,” reinforcing the overall success of this pedagogical method.

In addition to summarizing the responses, we conducted inferential statistical tests to explore whether students’ perceived ease of understanding was associated with their likelihood of recommending the method. A chi-square test for independence was applied to examine this relationship. The chi-square test is commonly used in educational research to determine if there is a statistically significant association between two categorical variables (Field, 2018). In this case, the test revealed whether students who found the method easier to understand were more likely to recommend it to others.

Furthermore, we calculated a 95% confidence interval for the average rating of 4.48 to assess the precision of the students’ satisfaction scores. The confidence interval provides a range within which the actual population mean is likely to fall, offering a more reliable interpretation of the data. In educational research, confidence intervals are valuable for understanding the degree of certainty associated with student feedback (Creswell, 2014).

The results from both descriptive and inferential statistics highlight the success of using Turtle

Graphics to teach programming concepts. The high average rating and positive feedback on recommendation likelihood suggest that this method effectively engages students and enhances their understanding of computational thinking. The findings indicate that visual representation of programming through Turtle Graphics helps students bridge the gap between abstract concepts and practical applications. This aligns with existing research, which emphasizes the importance of interactive and visual tools in beginner-level programming education (Tabachnick and Fidell, 2019).

By performing a chi-square test for independence, we further confirmed that students who rated the method as easier to understand were more likely to recommend it. This relationship suggests that pedagogical strategies that simplify complex concepts can directly impact student satisfaction and their willingness to endorse the teaching method. Additionally, the narrow range of confidence intervals for the average rating of 4.48 reinforces the robustness of the students' positive feedback.

The results of the chi-square test are as follows:

- Chi-square statistic: 0.55
- p-value: 0.46

The p-value of 0.46 is greater than the commonly used significance level of 0.05, indicating that there is no statistically significant difference between the observed distribution of responses (students finding the method “Easy” vs. “Not Easy”) and the expected distribution. In other words, the ease of understanding using Turtle Graphics is consistent with what we would expect if the teaching method had no effect.

This result suggests that, although many students found the method to be straightforward, the data does not provide sufficient evidence to conclude that their perceptions significantly differ from a neutral expectation. Therefore, the effectiveness of Turtle Graphics in improving ease of understanding may vary depending on other

factors not accounted for in this analysis. These findings are based on a numerical interpretation, but the research also includes qualitative insights, as discussed below.

Qualitative Analysis

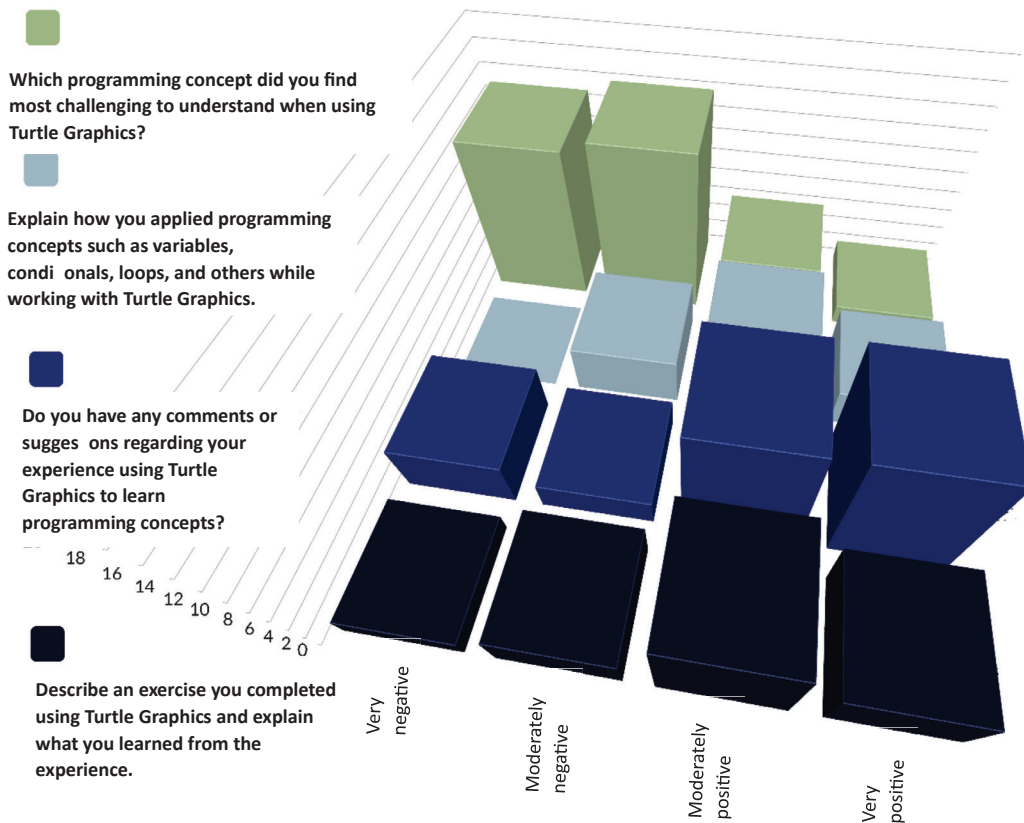
Qualitative analysis is a research method that focuses on understanding the meanings and experiences of individuals or groups through the collection and interpretation of non-numerical data such as words, texts, images, or observations. This approach aims to explore the complexity of human behavior, social interactions, and cultural phenomena by emphasizing the interpretation of participants' perspectives within their specific contexts (Denzin and Lincoln, 2018).

The process typically involves techniques such as thematic analysis, coding, and data categorization to identify recurring patterns, central themes, or emerging insights (Braun and Clarke, 2006). Unlike quantitative methods, which prioritize numerical data and statistical analysis, qualitative analysis values depth over breadth, seeking to provide a rich, nuanced understanding of the research topic (Creswell and Poth, 2018).

This methodology is especially valuable in disciplines such as the social sciences, education, and healthcare, where the exploration of subjective experiences and complex social dynamics is essential (Patton, 2015).

The 3D bar chart in Figure 5, visualizes sentiment analysis of open-ended responses to a survey about using Turtle Charts to learn programming fundamentals. Four questions are analyzed along a spectrum of feelings ranging from “Very Negative” to “Very Positive.”

Starting from the left, the first question—regarding the most challenging programming concept to understand using Turtle Charts—shows a balanced sentiment, with a slight bias toward the negative. This suggests that while some respondents faced difficulties, the distribution of responses spans the entire sentiment spectrum, indicating a diverse range of learning experiences.

Figure 6. Experience-based sentiment analysis on learning programming fundamentals using turtle charts

Source: Own elaboration.

The second question, asked respondents to explain how they used computer programming concepts such as variables, conditionals, and loops in Turtle Graphics programming. Here, the sentiment was predominantly positive, with the “Moderately Positive” and “Very Positive” categories being the most populated. This pattern, indicates that most students found the application of these concepts in Turtle Graphics beneficial and understandable.

The third question invited students to provide comments or suggestions related to the experience. Responses here were overwhelmingly positive, with “Very Positive” emerging as the most common sentiment, followed closely by “Moderately Positive.” The absence of negative sentiments suggests a high level of satisfaction with the Turtle Graphics learning experience.

The fourth and final question asked students to describe an exercise completed using Turtle

Charts and what they learned from it. This question also yielded strongly positive responses, with “Very Positive” being the dominant category. This suggests that practical applications using Turtle Charts significantly enhanced students’ understanding and engagement.

In summary, the sentiment analysis reveals that Turtle Charts are generally well-received as a tool for teaching programming fundamentals. While some challenges were noted, the overall sentiment was clearly positive, indicating that students appreciated the method’s interactive nature and its ability to reinforce programming concepts effectively.

Conclusions and future work

The synthesis of the survey results reveals that Turtle Graphics, as an educational tool for learning programming, is favorably received by students. Its

ability to visually represent core programming concepts—such as variables, loops, and conditionals—within an engaging and interactive environment has demonstrated pedagogical value. While some respondents expressed a neutral stance, indicating variability in the perceived intuitiveness of Turtle Graphics, the overall data trends strongly positively.

A notable 95% of students rated their engagement with Turtle Graphics as four or higher on a five-point scale. This high level of engagement suggests that the tool not only facilitates the comprehension of programming fundamentals but also does so in a way that captures students' interest—an essential component of effective learning.

Moreover, students' willingness to recommend Turtle Graphics further reflects its perceived educational value. An overwhelming 64 out of 66 respondents provided affirmative recommendations, underscoring a strong endorsement of the platform. The positive sentiment found in open-ended responses, particularly regarding the usefulness of the tool and the exercises completed, further reinforces the positive perception.

Despite the variation in individual experiences, the practical implementation of Turtle Graphics in educational contexts appears to contribute meaningfully to both conceptual understanding and learner engagement. In conclusion, Turtle Graphics emerges as a powerful ally in the effort to demystify programming for beginners, offering a visual and interactive approach that promotes sustained interest and deeper comprehension.

Acknowledgments

The authors extend their gratitude to the 66 students who participated in the case study. These students were enrolled in the first semester of the Systems Engineering program at the University of Nariño. In accordance with the Helsinki Declaration, all participants signed an informed consent form affirming that their participation did not involve any risk. Furthermore, this study was conducted as part of the 2022 Faculty Research Call at the University of Nariño, located in southern Colombia.

References

- American Educational Research Association (AERA). (2011). Ethical standards in research.
- Babbie, E. (2016). *The practice of social research* (14th ed.). Cengage Learning.
- Bakar, M.A., Mukhtar, M., and Khalid, F. (2020). The Effect of Turtle Graphics Approach on Students' Motivation to Learn Programming: A Case Study in a Malaysian University. *International Journal of Information and Educational Technology*, 10(4), 290-297. doi: <https://doi.org/10.18178/ijiet.2020.10.4.1378>
- Barr, V., and Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, 2(1), 48-54 doi: <https://doi.org/10.1145/1929887.1929905>
- Becker, BA, Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., and Santos, EA (2023). Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 500-506). doi: <https://doi.org/10.1145/3545945.3569759>
- Braun, V., and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77-101. doi: <https://doi.org/10.1191/1478088706qp0630a>
- Cooper, S., Dann, W., and Pausch, R. (2003). Using Animated 3D Graphics to Prepare Novices for CS1. *Computer Science Education*, 13(1), 3-30. doi: <https://doi.org/10.1076/csed.13.1.3.13540>
- Creswell, J. W. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (4th ed.). SAGE Publications.
- Creswell, J. W., and Plano Clark, V. L. (2011). *Designing and conducting mixed methods research* (2nd ed.). SAGE Publications.
- Creswell, J. W., and Poth, C. N. (2018). *Qualitative inquiry and research design: Choosing among five approaches* (4th ed.). SAGE Publications.
- Denzin, N. K., and Lincoln, Y. S. (2018). *The SAGE handbook of qualitative research* (5th ed.). SAGE Publications.

- Field, A. (2018). *Discovering statistics using IBM SPSS statistics* (5th ed.). SAGE Publications.
- Fraenkel, J. R., Wallen, N. E., and Hyun, H. H. (2019). *How to design and evaluate research in education* (10th ed.). McGraw-Hill Education.
- Gall, M. D., Gall, J. P., and Borg, W. R. (2007). *Educational research: An introduction* (8th ed.). Pearson Education.
- Insuasti, J. (2016). Problems of teaching and learning programming fundamentals. *Education and Social Development Magazine*, 10(2), 234-246. DOI: <https://doi.org/10.18359/reds.1966>
- Kay, J., and Tyler, P. (1992). A Microworld for Developing Learning Design Strategies. *Computer Science Education*, 3(2), 111-122. DOI: <https://doi.org/10.1080/0899340920030203>
- Kiper, J.D., and Bishop-Clark, C. (1995). Abstraction, Design, and Theory in Computer Science: A Liberal Arts Course. *Computer Science Education*, 6(1), 93-110. DOI: <https://doi.org/10.1080/0899340950060106>
- Kronk, H. (2021). Seymour Papert, logo turtles, and the origin of educational robots. *E-learning inside*. [Online] <https://news.elearninginside.com/seymour-papert-logo-turtles-and-the-origin-of-educational-robots/>
- Laura-Ochoa, L., and Bedregal-Alpaca, N. (2022). Incorporation of computational thinking practices to enhance learning in a programming course-*International Journal of Advanced Computer Science and Applications*, 13(2). DOI: <https://doi.org/10.14569/IJACSA.2022.0130224>
- Marbach, J., Maximova, A., and Staub, J. (2022). A Tool to Create and Conduct Custom Assessments in Turtle Graphics. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 15-26). Cham: Springer International Publishing. DOI: https://doi.org/10.1007/978-3-031-15851-3_2
- MEN (2022). System for the Prevention and Analysis of Dropouts in Higher Education Institutions - SPADIES. Ministry of National Education [Online] <https://www.mineducacion.gov.co/sistemasinfo/spadies/secciones/Estadisticas-de-desercion/>
- Patton, M. Q. (2015). *Qualitative research and evaluation methods: Integrating theory and practice* (4th ed.). SAGE Publications.
- Pelánek, R., and Effenberger, T. (2020). Design and analysis of microworlds and puzzles for block-based programming. *Computer Science Education*, 32(1), 66-104. DOI: <https://doi.org/10.1080/08993408.2020.1832813>
- Stebbins, R. A. (2001). *Exploratory research in the social sciences*. SAGE Publications. DOI: <https://doi.org/10.4135/9781412984249>
- Swedberg, R. (2020). Exploratory research. In *The SAGE Encyclopedia of Qualitative Research Methods*. SAGE Publications. DOI: <https://doi.org/10.1017/9781108762519.002>
- Tabachnick, B. G., and Fidell, L. S. (2019). *Using multivariate statistics* (7th ed.). Pearson.
- Tashakkori, A., and Teddlie, C. (2004). *Handbook of mixed methods in social and behavioral research*. SAGE Publications.
- US Department of Commerce, Census Bureau. (2021). American Community Survey [Data set]. National Center for Education Statistics. [Online] <https://www.census.gov/programs-surveys/acs>
- Wei, X., Lin, L., Meng, N., Tan, W., and Kong, S.C. (2021). The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Computers and Education*, 160, 104023. DOI: <https://doi.org/10.1016/j.compedu.2020.104023>
- Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35 DOI: <https://doi.org/10.1145/1118178.1118215>