International Journal of Electrical Engineering & Emerging Technology, Vol. 06, No. 01, JUN 2023, pp 20-23

# An Exploration of Visual and Textual Based Programming Languages: A Comparative Analysis

Awais Ahmed[1], Abdul Hameed Soomro[2], Sumair Shah[1], Ayaz Attar[1], Kaneez Fatima[1], Urmelia Kumari [1]
Dept. of Electrical Engineering, QUEST Nawabshah, Campus Larkano Pakistan
Corresponding author e-mail: (abdul.hameed@quest.edu.pk)

[**Received on:** 17/05/2023 **Accepted on**: 19/06/2023 **Published on:** 21/06/2023]

***Abstract*— This study aims to compare the comprehension of textual programming languages and visual programming languages in the context of their increasing significance in today's digital world. As technology advances, programming languages have become essential for software development, and two prominent methods for creating applications are visual programming languages and textual programming languages. In this research, Python and C++ were selected as representative textual programming languages, while Scratch and LEGO Mind storms EV3 were chosen as representative visual programming languages. To assess the effectiveness of textual programming languages, the study developed separate teaching materials for each language type. A questionnaire based on the ARCS motivation model was administered to over 40 students to evaluate their motivation levels. The results revealed that children's understanding improved as they progressed in classes using visual programming languages. Conversely, the study found that the use of textual programming languages, such as Processing, resulted in increased variability in the satisfaction factor as classes progressed, providing limited benefits to students with high self-decay tendencies. The outcomes of this analysis have practical implications for software developers, educators, and students interested in exploring the realm of programming languages and can make informed decisions when designing programming tools and environments. Furthermore, students can make more informed choices regarding the programming languages they wish to learn based on their individual preferences and goals.**

***Index Terms*— Textual programming languages, Visual programming languages, Python, C++, LEGO Mind storms EV3**

## I. INTRODUCTION

Learning is a consistent interaction that includes the change of data and experience into capacities and information. Programming, also known as the foundation of computer science (CS), is the primary mode of instruction in computer science [1]. People learn a variety of programming languages to communicate with computers, laptops, and other electronic devices. Rarely, programming is taught visually, through slides, texts, or both [2]. Researchers are trying to run through the most effective way to teach programming so that students can learn more quickly [3]. Text-based programming (TBP) and visual- based programming (VBP) are two distinct approaches to programming instruction. Programs in text-based programming are written in a typed or text format [4], whereas in visual bases programming, code components can be implemented by simply dragging and dropping instead of manually writing codes as shown in Fig. 1(a) [5]. Students have been the subject of a variety of experiments designed by researchers to determine the most effective method for facilitating student learning while maintaining the quality of instruction [6]. They concluded that students are more interested in learning programming visually than text- based [7]. Additionally, visual-based programming increases students' motivation to learn to program, whereas text-based programming increases students' variance satisfaction [8]. Another reason students are interested in visual programming is that there are no run time errors and no need to learn programming language data types, which can be helpful but also dangerous for not grasping fundamental programming concepts [9]. Visual-based instruction enhances comprehension and increases the number of learning outcomes . In contrast to text-based programming languages, in visual-based programming, errors like compilation errors, syntax errors, and runtime errors are handled by the programmer [10]. Ouahbi et al. claim that, according to the endings of a 2015 study, only 10.3% of students learn through textual programming, whereas 65 percent of students learn and achieve core concepts through visualization [11]. Additionally, students are willing to learn programming more attentively and interestingly through visual means. In contrast to the traditional textual programming language, learning visually includes the process of using graphic elements in program design to perform certain functions [12]. The learning environment is suitable for students who are not necessarily majoring in computer science and may ease the learning of programming languages [13]. The way programming concepts are understood is greatly in fenced by the learning environment. One major advantage of visualizing learning is that programming concepts are accessible to students of all majors [14]. Visual-based instruction enhances comprehension and increases the number of learning outcomes [15]. The educator's experience will likely be different depending on the type of programming, such as textual or visual programming. The results of a quasi-experiment indicate that students with moderate and low self-decay, in particular, bane toted more from visual teaching when it comes to fundamental programming concepts [16]. In adequate comprehension of fundamental programming concepts and the mapping of programming to actual simulated situations may contribute to difficulties in programming education. Visual programming, for instance, reduces the likelihood of beginners making syntax errors in programming [3] and helps students visualize the actual functionality without focusing too much on the syntax, thereby increasing their interest in and motivation to learn to program. Research in education shows that students learn more when they can interact with a suitable representation (Ainsworth, 2006) [17] and some researchers have also suggested changing the programming environment to improve learning skills. According to Ouahbi et al.2005, the complexity of teaching programming has been identified as one of the challenges in the literature [15]. According to the endings of a 2015 study, only 10.3% of students in the comparison group who were learning textual programming expressed an interest in visual programming, whereas 65 percent of students who were visualizing expressed an interest in learning more programming [8]. On the other hand, a VBP makes it easier for students to visually identify which blocks perform different tasks. However, some blocks may be too similar, which can lead to confusion [7]. In this research study 40.3% of students in the comparison group who were learning textual programming expressed an interest in visual programming, whereas 67.7 percent of students who were visualizing expressed an interest in learning more programming [8].

Section 2 describes the Methodology
Section 3 describes the Results and discussions
Section 4 describes the Conclusions
Section 5 describes the Future recommendations

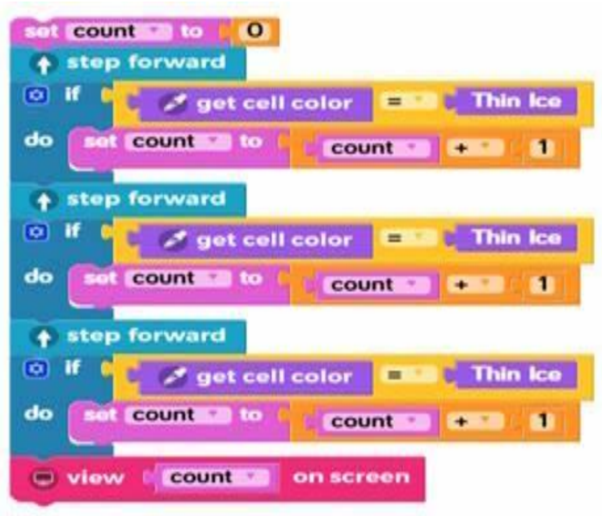
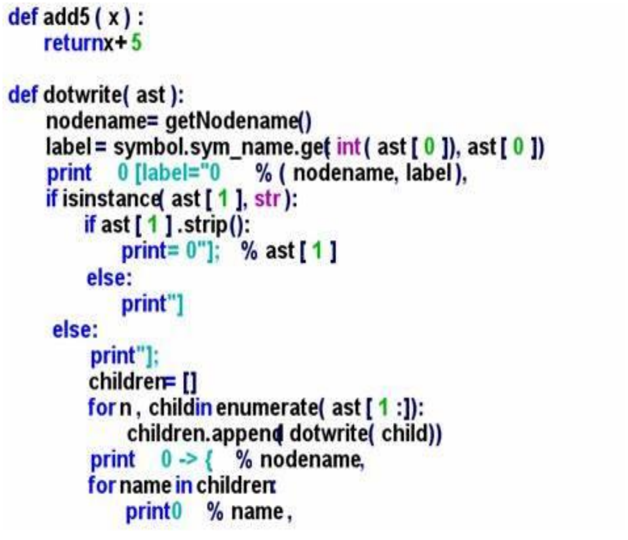

set count to 0
step forward
if get cell color = Thin Ice
do set count to count + 1
step forward
if get cell color = Thin Ice
do set count to count + 1
step forward
if get cell color = Thin Ice
do set count to count + 1
view count on screen

Fig.1 (a) Visual programming



```
def add5 ( x ) :
    returnx+ 5

def dotwrite( ast ):
    nodename= getNodename()
    label= symbol.sym_name.get( int( ast [ 0 ]), ast [ 0 ])
    print    0 [label="0     % ( nodename, label ),
    if isinstance( ast [ 1 ], str ):
        if ast [ 1 ] .strip():
            print= 0"];   % ast [ 1 ]
        else:
            print"]
    else:
        print"];
        children= []
        for n , childin enumerate( ast [ 1 :]):
            children.append( dotwrite( child))
        print   0 -> {   % nodename,
        for name in children:
            print0   % name ,
```

Fig.1 (b) Text Programming

## II. Methodology

We provide species about how, when, and with whom the study was conducted in this section. After conducting a survey, we conducted a session with students. The textual and visual-based programming was evaluated by 12 choice-based questions in the survey. From February 5 to February 20, 2023, the goal was to survey as many students as possible about textual and visual programming in Pakistan. The survey took positive minutes to complete anonymously, and participants were asked to share their thoughts and experiences with textual and visual-based programming. The analysis includes the survey's summarized results, this study surveyed elementary, primary, and university students. Participants in the survey were asked- closed- ended questions to get a general understanding of the students' perspectives on text-based programming and visual-based programming.

Fifty students were taught visual programming using Minestrone EV3, while another 40 were taught text programming using the C language to collect qualitative and quantitative data. An additional 40 students were taught VBP using Minestrone EV3, and 40 were taught TBP using the C language. We covered fundamental programming concepts such as variables, loops, functions, and the declaration, devotion, and assignment of conditional statements. During the session, we examined the students' conduct, interest, motivation, and participation in programming. A brief test was administered to students who had previously learned text-based and visual-based programming following the session. This evaluation included questions such as "What does the program do?" and "Locate the error in the program's solution?" as well as identifying missing steps or the algorithm's order of steps.

## III. Results and Discussion

In this section, the analysis of the data gathered from the survey conducted, which included the students' evaluations, reviews, suggestions, experiences, and opinions, is covered. According to the survey's data, 59.9% of students had not learned K-12 standard programming, either visually or textually as shown in Fig. 2(a). However, 15.3% of students had learned programming through VBP, 68.8% had learned programming through TBP, and 15.9% had not learned programming at all as shown in Fig. 2(b), it showed that, interestingly, 68.8% of students favored the use of VBP rather than TBP when analyzing their opinions regarding TBP and VBP. The students taught by VBP were extremely excited to learn to program and had actively participated, according to the qualitative analysis data gathered from assessments of both TBP and VBP students. They were observed attempting to complete the requested tasks independently by exploring the software and utilizing various components. The students quickly griped the fundamental idea thanks to the code's visual representation. In less time and a better manner, they improved their comprehension of fundamental programming concepts. On the other hand, students who had learned programming through TBP were initially very excited, but as time went on, they appeared less eager to move on. As soon as they started using the code and learning C, they kept getting stuck in implementations, as if they were waiting for us to tell them what to do next. They spent more time analyzing and resolving the errors because, when they implemented TBP students' group lost interest. According to the data, 20% of the TBP students were still trying, but 80% had given up. It is important to note that the TBP group consumed 1.5 times as much as the VBP group, so the TBP group students needed more time to absorb the information. According to the assessment, students who had learned through VBL performed better than students who had learned through TBL. With an overall average score of 90%, students who learned VBP first and then TBP performed exceptionally well.

Following the evaluation, we collected student reviews, solicited their feedback, and asked how they felt while completing tasks and how well they understood fundamental programming concepts.

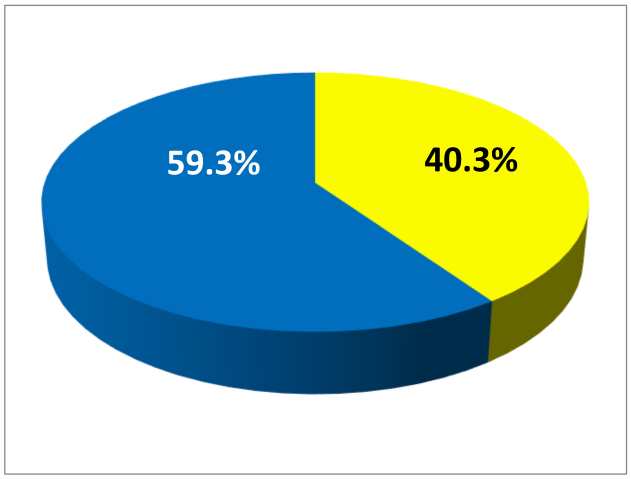


Fig. 2(a) 30 responses in K-12 for program languages Learning

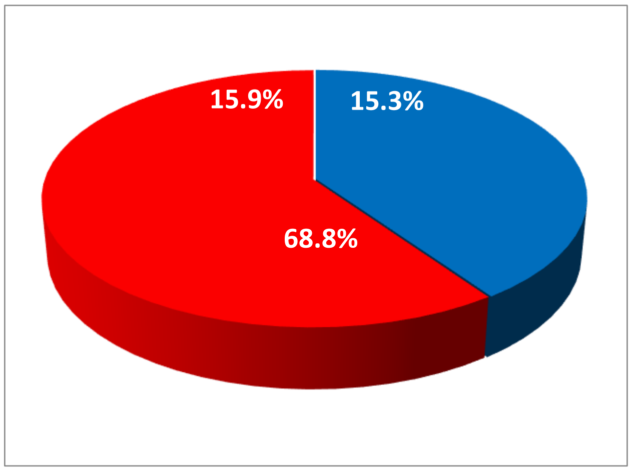


Fig. 2(b) Program languages learning

### A. *Discussion*

The study conducted a survey to determine students' preference for learning programming through TBP or VBP, as well as an effective, time-saving, and less labor-intensive method of instruction. Based on the survey results, it was found that the majority of students preferred to learn through VBP rather than TBP. To obtain more precise and accurate qualitative results, the 150 students were divided into three groups for a face-to-face physical session after the survey, with each section having fifty students. The students learned fundamental programming concepts through various teaching methods in each group. One group learned programming using the TBP strategy, another group learned using the VBP strategy, and the third group learned using a combination of both strategies, starting with VBP and then transitioning to TBP.

During the session, the behavior of each group of students was observed. Initially, all 150 students actively participated, but as they began to learn programming through TBP, they found it more difficult to comprehend the concepts. On the other hand, students who learned through VBP actively participated and attempted to explore the tools on their own. Because they were visualizing the code through components, they quickly grasped programming concepts. The students who used TBP to learn understood the material slowly.

After the first 100 students completed the session, assessments were conducted. It was found that the TBP group students did not perform as well as the VBP group students. However, the third group that was taught using a hybrid strategy performed exceptionally well, with an average score of 90%.

## IV. Conclusion

This paper presented a methodical, classroom-based, and comparative study of the effects of programming modality on students. The study demonstrated how modality affected students' attitudes, perceptions, and conceptual learning. As a result, it supported the idea that programming should be taught first using VBP rather than TBP, even though VBP required more time and produced more meaningful results. The study aimed to help students understand the relationship between TBP and VBP and gain a clear picture of the programming set. It also aimed to determine which tools should be used in classrooms and how future environments for introductory programming should be designed. Findings from studies like this one were crucial to ensuring that the current generation of students was best served because of the growing ecosystem of educational programming environments and curricula and the growing presence of computer science in K-12 education. Although there were still many unanswered questions and a lot of work to be done, this study helped to solve one part of the bigger problem of how to best teach today's students important computing concepts. The researchers hoped that this kind of research would continue to advance our comprehension of the connection between the learning environment and the learner as the number of computer science learning opportunities increased. They also hoped that the results of this research would help shape the next generation of tools, curricula, and classroom practices. By taking on this challenge, students could be prepared for the computational future they would face in the classroom and beyond.

## RECOMMENDATION

The paper suggests that visual programming languages can be used to teach programming in primary schools, thereby making programming more accessible to K-12 students and reducing entry barriers. The analysis showed that students in grades K-12 are more familiar with visual programming languages, and the study found that using these languages enhanced students' understanding of fundamental programming concepts, leading to increased motivation and satisfaction.

However, the study also observed that students in elementary and secondary school struggled with the transition from text-based to visual-based instruction when they entered higher education. Therefore, it can be argued that visual programming languages are more suitable for teaching programming to K-12 children than text-based programming languages, especially from the perspective of increasing motivation.

## ACKNOWLEDGMENT

We acknowledge the support of QUEST Nawabshah Campus Larkana, Pakistan to achieve this Milestone.

## FUNDING

This research is without external funding

## REFERENCES

[1] T.-T. Wu, C.-J. Lin, S.-C. Wang, and Y.-M. Huang, "Tracking Visual Programming Language-Based Learning Progress for Computational Thinking Education," *Sustainability*, vol. 15, p. 1983, 2023.

[2] M. Glas, M. Vielberth, T. Reittinger, F. Böhm, and G. Pernul, "Improving cybersecurity skill development through visual programming," *Information & Computer Security*, 2023.

[3] L. Li, X. Xue, and X. Yang, "Discussion on programming language teaching of information major based on Python language," in *Third International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI 2022)*, 2023, pp. 280-285.

[4] K. Napier, T. Bhowmik, and S. Wang, "An empirical study of text-based machine learning models for vulnerability detection," *Empirical Software Engineering*, vol. 28, p. 38, 2023.

[5] D. Niermann, T. Doernbach, C. Petzoldt, M. Isken, and M. Freitag, "Software framework concept with visual programming and digital twin for intuitive process creation with multiple robotic systems," *Robotics and Computer-Integrated Manufacturing*, vol. 82, p. 102536, 2023.

[6] M. R. Jegarluei, P. Aristidou, and S. Azizi, "Wide-Area backup protection against asymmetrical faults in the presence of renewable energy sources," *International Journal of Electrical Power & Energy Systems*, vol. 144, p. 108528, 2023.

[7] A. Espinal, C. Vieira, and V. Guerrero-Bequis, "Student ability and difficulties with transfer from a block-based programming language into other programming languages: a case study in Colombia," *Computer Science Education*, pp. 1-33, 2022.

[8] J. Oppenlaender, "Prompt Engineering for Text-Based Generative Art," *arXiv preprint arXiv:2204.13988*, 2022.

[9] H. De Silva Joyce and S. Feez, *Text-based language and literacy education: Programming and methodology*: Phoenix Education, 2012.

[10] D. Weintrop and U. Wilensky, "Comparing block-based and text-based programming in high school computer science classrooms," *ACM Transactions on Computing Education (TOCE)*, vol. 18, pp. 1-25, 2017.

[11] D. Saito, H. Washizaki, and Y. Fukazawa, "Work in progress: A comparison of programming way: Illustration-based programming and text-based programming," in *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 2015, pp. 220-223.

[12] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The scratch programming language and environment," *ACM Transactions on Computing Education (TOCE)*, vol. 10, pp. 1-15, 2010.

[13] M. Adam, M. Daoud, and P. Frison, "Direct manipulation versus text-based programming: An experiment report," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 353-359.

[14] M. Kölling, N. C. Brown, and A. Altadmri, "Frame-based editing: Easing the transition from blocks to text-based programming," in *Proceedings of the Workshop in Primary and Secondary Computing Education*, 2015, pp. 29-38.

[15] R. Jamal and L. Wenzel, "The applicability of the visual programming language LabVIEW to large real-world applications," in *Proceedings of Symposium on Visual Languages*, 1995, pp. 99-106.

[16] M. Noone and A. Mooney, "First programming language: Visual or textual?," *arXiv preprint arXiv:1710.11557*, 2017.

[17] A. Jagadeesha, P. Rayavaram, J. Marward, S. Narain, and C. S. Lee, "Integrating Data Structures and Algorithms in K-12 Education using Block-based Programming," *arXiv preprint arXiv:2302.11659*, 2023.