Effects of a Block-based Scaffolded Tool on Students' Introduction to Hierarchical Data Structures

Pedro G. Feijóo-García, Amanpreet Kapoor, Christina Gardner-McCune, and Eric Ragan

***Abstract*—*Contribution:* In this paper, the authors present findings and insights on the efficacy of using an educational block-based programming (BBP) environment—Blocks4DS, to teach the Binary Search Tree (BST).**

***Background:* For a decade, BBP environments have been a hot topic in the Computer Science Education (CSEd) community to promote interactive active learning of programming. However, little attention has been paid to BBP environments' efficacy on non-introductory courses like Data Structures & Algorithms (DS&A). DS&A courses are challenging to students due to levels of abstraction that could be reduced by syntax-free features existing in BBP interfaces.**

***Research Questions:* 1) Can undergraduate computing-major students learn about the BST using Blocks4DS? 2) Do undergraduate computing-major students understand better BSTs when learning with a BBP environment? 3) How do undergraduate computing-major students perceive Blocks4DS for non-introductory CS topics?**

***Methodology:* A mixed-method study was designed, using a control and intervention group with 83 first and second-year Computer Science (CS) students, to evaluate the efficacy of Blocks4DS compared to traditional instructional methods (e.g., whiteboard and pseudo code). The authors evaluated its efficacy based on students' conceptual understanding and perceptions.**

***Findings:* It was found that, regardless of prior experience with text-based programming languages and instructional approaches, students introduced to the BST with Blocks4DS gained significant conceptual understanding, and performed as well as peers instructed with pseudo-code. Also, 83.3% of students found the tool engaging and 72.3% found it useful to learn DS&A. This suggests that Blocks4DS can be used to teach DS&A.**

***Index Terms*—Computer science education, data structures and algorithms, visual languages**

I. Introduction

THE Computer Science Education (CSEd) research community has explored for decades how students learn to program, building literature on pedagogy, tools, and strategies to develop computing literacy [16]. This exploration has mainly focused on novice students in contexts such as K-12 and undergraduate introductory Computer Science (CS) courses like CS0 and CS1 [20], with little focus on non-introductory undergraduate courses such as Data Structures & Algorithms (DS&A) [28, 37]. DS&A courses are commonly taught in the second year of Computer Science (CS) curricula, and generally require students to have previously gained programming and abstraction skills in one text-based programming language (e.g., Java, C++) [20, 28]. The introduction of new levels of abstraction, time and memory for efficiency considerations, and non-linear representations make these courses challenging for students to learn and instructors to teach.

Algorithms Visualization (AV) has been a way the CSEd community has taken to assist students [3, 33, 34]. Multiple visualization tools, static and dynamic, have been designed and used to foster and scaffold abstraction understanding on different educational levels. Static tools display algorithms' operations as a whole or step-by-step, avoiding any code input from the user [13, 33]. Although these tools help in illustrating the procedures students are supposed to follow, they lack active-learning elements to allow students to visualize and code at the same time. On the other hand, dynamic tools such as Python Tutor [15], or Willow [25], feature visualizations with code-editing capabilities in a text-based language, accompanied by an interpreter in most cases [33]. These tools are excellent for addressing students' needs in DS&A courses as they provide visualization and coding features [17]. However, text-based languages bring syntax challenges dependent on the programming language used in the tool, implying unnecessary cognitive load due to syntax errors [19].

One way the community has tried to address syntax barriers for novice programmers and to broaden the participation of students in computing is through block-based programming (BBP) environments [4, 5, 23, 24]. For a decade, these environments have been a hot topic in the CSEd community to promote interactive active learning of programming [23]. Environments featuring BBP assist students to learn

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

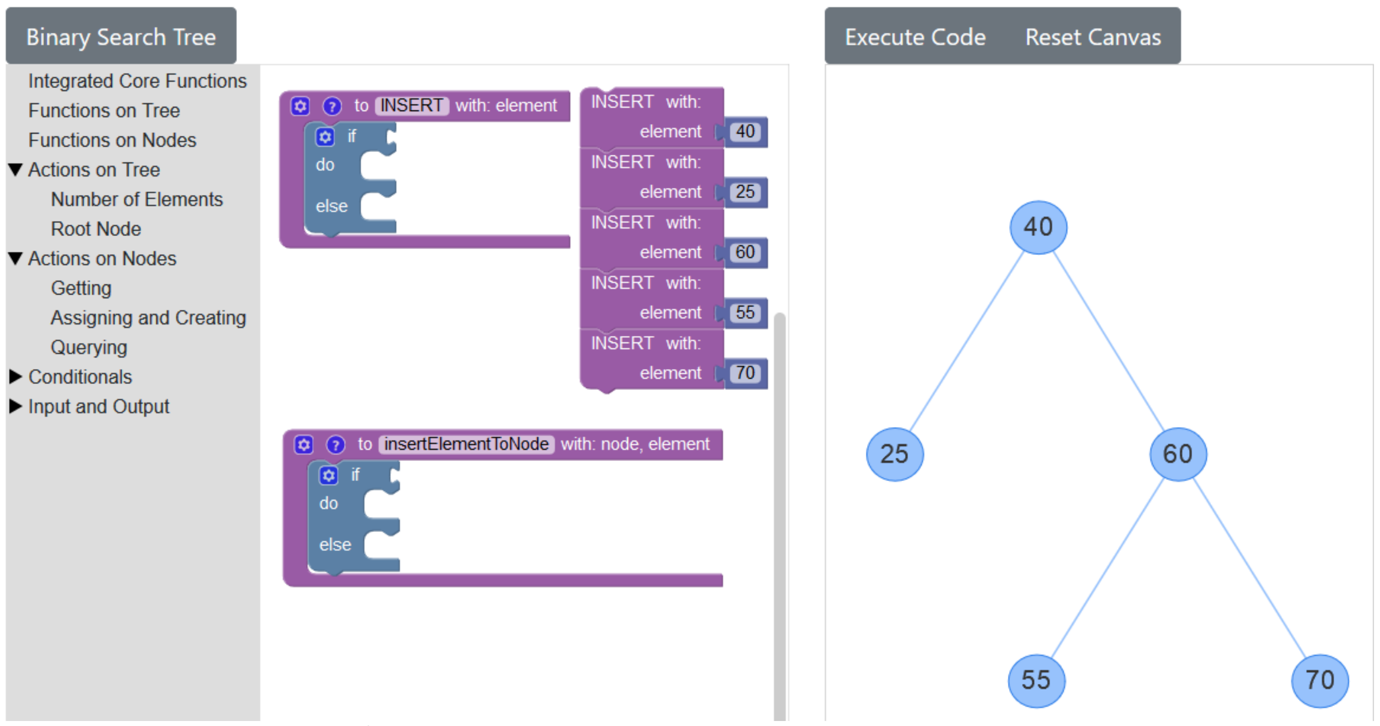

Fig. 1. Blocks4DS—Graphical User Interface (GUI)

programming without syntax errors, decoupling syntax from computational thinking and CS theory [5, 23, 24]. Although popular, the CSEd community generally uses BBP with sand-box environments for children and K-12 [4], introducing students to simple constructs such as variables, conditionals, loops, and functions (i.e., procedural programming). BBP environments can be immensely helpful on topics that involve high levels of abstraction, such as DS&A, as the removal of syntax errors may help students reduce their cognitive load and focus on computational thinking skills [5, 19]. Unfortunately, BBP environments have been underexplored in CS undergraduate contexts: only a few studies have referred to them in CS1 [4, 5, 22], or to teach linear data structures in DS&A courses [1]. More research is necessary to understand their effectiveness on topics that involve higher levels of abstraction, such as hierarchical and non-linear data structures.

To bridge the gap of exposing students to algorithmic visualization with less syntactic barriers and through active learning, the tool Blocks4DS was designed. The tool is a BBP environment designed to assist in the instruction of the Binary Search Tree (BST) [11]. The tool features a block-based canvas that assists on the BST elements and operations (i.e., create, search, and delete), in addition to a visualizer that displays the data structure programmed by the student. The authors combined the best of both worlds, Algorithms Visualization (AV) and BBP (see Figure 1), to evaluate if dynamic visualizations coupled with languages that have less syntax sugar (e.g., semi-colons, brackets, parenthesis, reserved words—public or private, etc.) can be effective in students' understanding of higher-level CS topics—hierarchical data structures. The tools were evaluated addressing the following questions:

- Q1: Can undergraduate computing-major students learn about the BST using Blocks4DS?
- Q2: Do undergraduate computing-major students understand better BSTs when learning with a BBP environment?
- Q3: How do undergraduate computing-major students perceive Blocks4DS for non-introductory CS topics?

For the scope of this paper, l*earning* is considered as the action of gaining new knowledge about the BST in terms of its properties, constrains, and operations. *Understanding*, on the other hand, concerns to students' expertness on the BST—on how robust their mental models on the topic are. The study assessed *learning* based on how students were able to identify a BST from a Binary Tree (BT), and on how they could recall the data structures' characteristics. *Understanding* was assessed based on how well students succeeded inserting elements into the BST and on how they explained the step-by-step process of its search operation (see Section III.E).

This paper reports about an online-asynchronous study in a CS2 course with undergraduate students (n=93) from a North American university, that compared students' conceptual understanding of the BST on two separate modes: Blocks4DS or pseudo-code (i.e., syntax-light textual approach to code). Students were assessed before (pre-assessment) and after (post-assessment) using one of the two modes. Some students (n=10) scored higher or equal to 80% (letter grade of B at the University of Florida, U.S.A.) for their pre-assessment and were not assessed further: participants were expected to be novices towards the BST. Then, students (n=83) were asked about the BST organization and basic operations such as inserting and searching for an element. Students were assessed using a between-subjects design with two groups: Blocks4DS (n=44), and pseudo-code (n=39). After the post-assessment, students were asked to switch modes and to report their perceptions. Finally, students had to compare modes, to score features of Blocks4DS, and to evaluate the tool's design: usefulness and engagement.

The study found that students improved their conceptual understanding with Blocks4DS between assessments, as their post-assessment scores differed significantly from the pre-assessment. Additionally, although students from the intervention group reported to be unfamiliar with BBP environments, they performed as well as students from the control group. This was true regardless of students' prior experience with text-based programming languages (i.e., Java and C++) and instructional approaches (e.g., instructors relaying on whiteboards in classrooms to code or illustrate data structures). The majority of students found the tool engaging (83.3%) and useful to learn about the BST (72.3%), with particular higher scores by Female students and non-native English-speaking students. These findings suggest Blocks4DS can be useful in non-introductory CS courses, helping students to learn programming and abstraction for DS&A.

II. Background

A. *Teaching DS&A in CSEd Literature*

The CS Curricula 2013 identifies learning outcomes of DS&A in three categories: Familiarity, Usage, and Assessment [29]. These outcomes cover recognizing, conceptually understanding, implementing, and critically evaluating scenarios for choosing different data structures and algorithms. Researchers and practitioners focusing on DS&A courses have created concept inventories to measure students' conceptual understanding of data structures [29], identified topics that students face difficulty when learning DS&A [10, 18, 31], used active learning approaches to strengthen foundations [7, 27], and incorporated dynamic visualization tools that require text-based languages to master basics [6, 8, 9, 15, 25]. This research includes work by Zingaro *et. al.*, who identified that students have challenges with both linear and non-linear data structures [37]. Specifically, they found that students face difficulties in analyzing the impact of BST insertions and how insertions impact shape [37]. Other research focusing on the use of visualization to foster students' understanding of DS&A is elaborated in Section II.B. Blocks4DS focuses on incorporating dynamic visualizations similar to [6, 8, 9, 15, 25]. However, the study reported in this paper uses a BBP language to allow students to visualize the DS&A unlike previous work (see Figure 1).

B. *Role of Visualizations in DS&A Education*

The use of interactive visualizations in CSEd for text-based languages dates back to the 1980s [33]. To understand the recent advancements in the field of AV, Shaffer et. al. [33] reviewed over 350 visualizations and found that most visualizations were of low quality and targeted introductory concepts [33]. They argued that most visualizations had low pedagogical utility as the visualization tools "rarely" illustrated the process of operations. For example, they explained that visualizations that covered insert and delete operations on tree data structures showed students the result of an operation rather than the process itself [33]. To organize the large set of studies and tools developed for Visualization purposes in Computing Education, an ACM ITiCSE Working Group developed a taxonomy of six ways in which students engage with visualizations: (1) No viewing, (2) Viewing, (3) Responding, (4) Changing, (5) Constructing, and (6) Presenting [26]. "No viewing" is classified as instructions without any visualizations, and "Viewing" is a passive activity of seeing visualizations. "Responding" classifies activities that include both watching visualizations as well as answering questions while the visualizations are presented with a tool. "Changing" involves altering the inputs or modifying the visualizations. "Constructing" refers to creating visualizations. Surprisingly, the results of the use of tools in the "Constructing" domain have had mixed results so far regarding learning [14, 27]. The last category, "Presenting" requires students to present visualizations to an audience for feedback and discussion. Studies have used this taxonomy for classifying the area under consideration [27]. For example, Grissom *et. al.*, studied algorithmic learning using the JHAVE visualization tool [14] under three conditions: "No viewing", "Viewing", and "Responding" levels and concluded that learning of computing concepts increases with the level of engagement, i.e., active learning or "responding" for visualization is better than static visualizations or "Viewing" [14]. In the context of the reported study, standalone pseudo-code is classified as "No viewing", whiteboard (i.e., refers to the classroom's artifact or object that is used by instructors in lectures to write, code, illustrate examples, etc.) diagrams or presentations as "Viewing", and Visualization using Blocks4DS as "Constructing".

C. *BBP Languages in Undergraduate CSEd*

BBP languages such as Scratch and Snap! that have massive adoption rates, have been primarily designed to target novice students and introduce them to programming [23]. These languages support novice programmers in learning to code without worrying about syntax [31, 35]. However, older students (>14 years of age) have underrated these environments as these languages have become a central component of curricula in introductory high school CS courses [35, 36]. In the context of undergraduate CSEd, BBP environments have been adopted and studied in preliminary CS courses such as CS1 [22, 30] and hybrid environments have also been studied to ensure students' smoother transition between blocks and text [4, 5]. In these undergraduate CS courses, novice students have reported positive experiences with Scratch before moving to text-based programming languages further along the course [22, 30]. The goal behind the design of Blocks4DS was to complement students' experiences when understanding hierarchical data structures through an environment different from drawing data structures on whiteboards (i.e., refers to the classroom's artifact or object that is used by instructors in lectures to write, code, illustrate examples, etc.) or using pseudo-code (i.e., syntax-light textual approach to code) for explanations. The utility of Blocks4DS is considered between pseudo-code and text-based programming languages.

III. Method

With the goal of evaluating the efficacy of Blocks4DS in the instruction of the BST, the authors conducted an online-asynchronous study with undergraduate students enrolled in a CS2 course at a large public university in North America. Data was collected via multiple questionnaires and surveys over two semesters: Spring 2020 and Summer 2020, after students were introduced to recursion and linear data structures—the linked list. The CS2 course is taught in C++ and it is the second programming course offered in the CS curriculum. Students' participation was no longer than 90 min for the study, and all communication with them was conducted via email.

A. Participants

The data reported in this paper corresponds to a sample population consisting of undergraduate computing-related majors (e.g., computer science and computer engineering), who had no prior experience with non-linear data structures (n=83). Participants' reported age was: 18-20 years of age (n=74), 21-25 years of age (n=3), 26-30 years of age (n=4), and >30 years of age (n=1). This range of participant ages in CS1 and CS2 courses at the University of Florida is common as more students are pursing minors in CS or post-baccalaureate degrees to gain valuable computing skills. A 7-point Likert scale was used to ask students to self-report their previous experience with BBP environments (1—I am not familiar at all to them, 7—I am very familiar to them). Participants were generally unfamiliar with BBP environments: 78.3% of them (n=65) reported to have none (1) or little (2-3) familiarity with these environments. Table I presents additional demographics of the study participants who agreed to respond to demographic questions (n=82).

TABLE I
Sample Population - Demographics

Variable	*Demographic Group*	*Counting*	*Percentage*
Gender	*Male*	*n=49*	*59.8%*
	Female	*n=30*	*36.6%*
	Non-Binary	*n=3*	*3.66%*
Race / Ethnicity	*Asian / Pacific Islander*	*n=27*	*32.9%*
	Black / African American	*n=3*	*3.66%*
	Hispanic / Latin American	*n=19*	*23.2%*
	Native American /Alaska Native	*n=0*	*0.00%*
	White, Non-Hispanic / Latin American	*n=28*	*34.1%*
	Middle Eastern / North African	*n=3*	*3.66%*
	Other	*n=2*	*2.44%*
Native Language	*Native English Speakers*	*n=64*	*78.0%*
	Non-Native English Speakers	*n=18*	*22.0%*

B. Study Design

The study was online and asynchronous (participants had not face-to-face interactions with the research team and could participate in the study at their own pace), and had participants from a CS2 course over two semesters: Spring 2020 and Summer 2020. The study was divided in three phases, giving access to students to the next phase as they completed the previous one.

1) First Phase

In this phase, the authors gave students an informed consent document with the details of the study. They were asked to read it and respond to an online survey if they agreed to participate. The authors then asked students to create an anonymous ID to use in the upcoming online questionnaires—to ensure anonymity. Further, students were asked to respond to the pre-assessment questionnaire followed by a phase-completion survey that collected their emails—so that the authors could contact them for the next phase. The pre-assessment questionnaire is described in Section III.E.

2) Second Phase

In this phase, students were randomly distributed into two groups. Both groups watched a video-lecture and worked on a follow-up assignment:

a. Students in the **control group** watched a video-lecture (22 min) introducing the Binary Tree (BT) and the Binary Search Tree (BST) with pseudo-code (n=39), and had to work on an assignment that asked them to complete a sample code, in pseudo-code, of the insertion function of the BST. The video-lecture is described in Section III.D.
b. Students in the **intervention group** watched a video-lecture (24 min) that featured Blocks4DS to introduce the BT and the BST (n=44), and had to work on an assignment that asked them to complete a block-based sample code of the insertion function of the BST. The video-lecture is described in Section III.D.

Both video-lectures and assignments addressed the same topics and problems, regardless of the mode used for them: Blocks4DS or pseudo-code. Subsequently, participants were asked to respond to a post-assessment questionnaire that featured the same number of questions and structure of the pre-assessment—see Section III.E. Once finished, students were asked to respond to a phase-completion survey, as they did in the first phase, which collected their emails for further communication.

3) Third Phase

The authors switched video-lectures and assignments between groups: this was done in order to provide enough experience to students to compare both modes. Then, the authors asked students to respond to a final questionnaire and self-report their perceptions and experience with the tool, its features, and the two distinct modes: Blocks4DS and pseudo-code. The final questionnaire is described in Section III.E. Once finished, students were asked to respond to a phase-completion survey to input their emails, as they did with the previous phases.

C. The Tool: Blocks4DS

Blocks4DS is a BBP environment introduced in 2019 [11], developed to assist students to learn data structures (see Figure 1). It presents a web-based interface that allows students to visualize the BST and work on its operations using a BBP interface (see Figure 1). As described by Feijóo-García *et al.*, [11], the tool categorizes blocks regarding their intended operation, referring to one of two JavaScript back-end classes: *Node* or *Tree*. As it is observed in Figure 1, the tool's scaffolding (i.e., assistance or support) is visually and conceptually organized in seven different categories: *Integrated Core Functions*, *Functions on Nodes*, *Actions on Tree*, *Actions on Nodes*, *Conditionals*, and *Input and Output*.

Blocks4DS was programmed using JavaScript, structured with HTML, and styled with CSS. The tool integrates Blockly—to feature a block-based canvas [12], and vis.js—to provide a visualization of the BST [2]. The tool was designed with a web-based architecture to ensure that instructors or students incur minimal cost for installations. Previously, instructors have reported difficulty in adopting AV tools in classrooms [32]. For this study, the authors updated the tool's interface to present a base template of the search operation, so that students were able to complete the proposed assignment.

D. Teaching Approach: Video-Lectures

To conduct the study, the authors developed two video-lectures that introduced concepts and definitions of the BT and the BST. Both videos also introduced basic operations on the BST: insertion, and search. The latter featured a different mode for each video: The control group watched the first video-lecture (22 min), which explained the search operation using pseudo-code and BST visualizations. The intervention group watched the second video-lecture (24 min), which explained the search operation with Blocks4DS. Both video-lectures were instructed by the first author, in English, and were slideshows.

Before participants' recruitment, both video-lectures were reviewed by six external Computing-related professionals unrelated to the study: three CS Faculty and three Doctoral scholars. Each video had three reviews. The authors used a questionnaire that featured a 7-point Likert scale (1—not clear at all, 7—very clear) for three questions. Descriptive statistics are presented (mean—M, and standard deviation—SD) for each question and video-lecture (control—pseudo-code; intervention—Blocks4DS):

1. How clear is the video tutorial's explanation of what a Binary Search Tree is? —control (M=7.0, SD=0.0); intervention (M=6.7, SD=0.7).
2. How clear is the video tutorial's explanation on how to insert elements in a Binary Search Tree? —control (M=5.7, SD=0.6); intervention (M=6.3, SD=0.6).
3. How clear is the video tutorial's explanation on how to search for an element in a Binary Search Tree? —control (M=6.7, SD=0.6); intervention (M=6.3, SD=0.6).

The instructional material was highly rated by the external peers, who indicated that both video-lectures were clear in the explanation of the BST, its concepts, and its operations, to strengthen reliability.

E. Data Collection

1) Pre-assessment and Post-assessment Questionnaires

Both questionnaires shared the same structure. Each of them had a total of seven (7) questions: four (4) close-ended, and three (3) open-ended [21]; addressing their understanding of the BT and the BST. All answers were scored as wrong (0 points) or correct (6 points). For open-ended questions, answers were scored by the research team using three categories—Satisfactory (6 points), Partially Satisfactory (3 points), and Unsatisfactory (0 points).

Questions used in both questionnaires were divided in three groups. The first group gathered two open-ended questions that asked participants to explain, in their own words, the properties of the BT and the BST—e.g., "Please explain, to the best of your ability, the properties of a Binary Tree." The second group corresponded to questions that asked participants to visually identify the BT, the BST, and to distinguish one from another—e.g., "Which of the following representations are not Binary Search Trees?" with four visual representations to select plus a final option that told "I don't know which of these representations are not Binary Search Trees." The third group had two questions that referred to the BST operations. One close-ended question—e.g., "You are asked to input into a Binary Search Tree the following sequence of numbers: 5, 3, 25, 16, 71, 1, 2, 4. Which BST representation corresponds to this input?" with four visual representations to select plus an option that told "I don't know which is the BST representation for this input.", and one open-ended question—e.g., "Using your own words, please explain the how to search for an element in a Binary Search Tree. If you do not know how to do so, please leave the text *I don't know*."

2) Final Questionnaire—Self-reported Perceptions

The questionnaire featured a 7-point Likert scale for ten questions on students' perceptions and experiences. Additionally, this questionnaire featured demographic questions such as students' age, gender, ethnicity, and native language. Demographic questions were used to study the impact of dynamic block-based visualizations on non-native English Speakers' perceptions.

F. Data Analysis

To respond to the questions posed in Section I, the authors broke down their analysis into two categories. The first category corresponds to students' performance between the pre-assessment and the post-assessment. The second one refers to students' perceptions about the tool and their experience. Data was analyzed using quantitative techniques that involved descriptive and inferential statistics [21]. Students' perceptions were analyzed using only descriptive statistics.

1) Students' Performance

Students' performance was analyzed between groups—the control group and the intervention group, using descriptive and inferential statistics [21] on two variables. The first variable

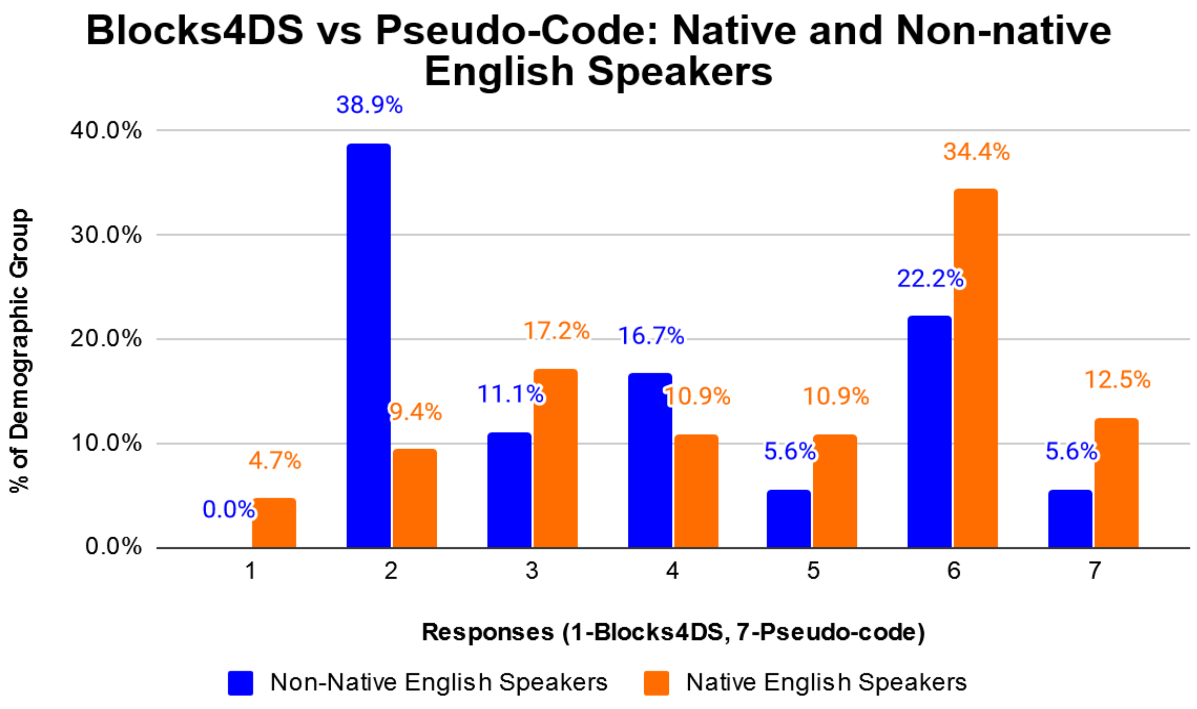

Fig. 2. Blocks4DS vs Pseudo-code: Native and Non-native English Speakers

considered was the post-assessment final score (S2)—rated from 0.0 to 6.0. The second variable was the students' improvement (W) from final scores between the S2 and the pre-assessment S1: this variable could be positive, neutral, or negative (Equation 1). S2 was not normally distributed so the non-parametric independent two-tailed Mann-Whitney U test was employed [21]. Since W was normally distributed, the authors used for it the parametric two-tailed independent Student's t-test [21].

Students' performance was also analyzed within groups comparing post-assessment and pre-assessment scores—S2 vs S1. Since data was not normally distributed in either the control or the intervention groups, the non-parametric dependent two-tailed Wilcoxon Signed Rank test was employed for it [21].

$$W = S2 - S1 \quad (1)$$

2) Students' Perceptions

The authors analyzed students' perceptions from 7-point Likert scales questions on the final questionnaire, by counting the number of responses per scale for each one of their seven options. Their analysis was based on descriptive statistics (**mean**—M, **standard deviation**—SD, and **mode**—Mo) centered on the demographic groups for which they had participants (see Table I). The authors focused their analysis on answers from six (out of ten) questions of the final questionnaire:

1. Based on your interaction with **Blocks4DS**, how **engaging** is the tool?
2. How **useful** do you find **Blocks4DS** to learn of data structures such as the BST?
3. How **useful** do you find a **block-based canvas**, like the one provided in Blocks4DS, to learn of data structures such as the BST?
4. How **useful** do you find a **visual representation** of the data structure, like the one provided in Blocks4DS, to learn of data structures such as the BST?
5. **Comparing** the **two modes** you used in the study, how much do you prefer using one versus the other?
6. **Comparing** the **two main features** (block-based canvas vs data structure visual representation) displayed in Blocks4DS, which one do you find more useful to learn of data structures such as the BST?

Four questions were removed due to their lack of contribution to the discussion of this study: the six questions included already addressed the research questions posed in Section I. The four questions discarded were:

7. Based on your interaction with Blocks4DS, please rate your experience of interacting with the tool.
8. Recalling the assignment, how difficult was working on it using blocks?
9. Recalling the assignment, how difficult was working on it using pseudo-code?
10. Recalling the assignment using pseudo-code, how much do you consider having a visual representation of the BST would have helped you responding to it?

IV. Findings and Results

In this Section, findings on students' performance and perceptions are outlined. These findings are presented following the categories, variables, and analysis described in Section III.F.

A. *Students' Performance*

Table II shows that students' post-assessment scores (S2) were, on average, high for both groups. Both groups scored an average S2 greater than 4.0 on a scale up to 6.0 (see Section III.E). For the control group (n=39), 74.4% of students had an S2 score greater than 4.0., and 39.5% (n=15) had an S2 greater than 5.0. For the intervention group (n=44), 64.1% of students (n=25) had an S2 greater than 4.0, and 38.5% (n=15) had an S2 greater than 5.0. As shown in Table II, S2 was not significantly different between groups (p>0.05).

TABLE II
BETWEEN-GROUPS: SUMMARY OF RESULTS

Variable	*Ctrl. M, SD*	*Intv. M, SD*	*p*	*Test*
S2-post-assessment	*4.5, 1.0*	*4.1, 1.4*	*0.28*	*Mann-Whitney U*
W-improvement	*3.5, 1.3*	*3.0, 1.73*	*0.31*	*t-test*

Students' improvement (W) was positive for both groups, as the average for them was 3.5 for the control group and 3.0 for the intervention group (Equation 1). For the control group (n=39), 64.1% of students (n=25) reported a W greater than 3.0, 33.3% (n=13) reported a W greater than 4.0, and 15.4% (n=6) reported a W greater than 5.0. For the intervention group (n=44), 48.7% of students (n=44) reported a W greater than 3.0, 38.5% (n=15) reported a W greater than 4.0, and 20.5% (n=8) reported a W greater than 5.0. As with S2, W was not significantly different between groups (p>0.05).

TABLE III
WITHIN-GROUPS: SUMMARY OF RESULTS

Group	*S1.M, SD*	*S2. M, SD*	*p*	*Test*
Control	*1.0, 1.0*	*4.5, 1.0*	*<0.01*	*Wilcoxon Signed Rank*
Intervention	*1.1, 1.2*	*4.1, 1.4*	*<0.01*	*Wilcoxon Signed Rank*

As observed in Table III, scores from the post-assessment (S2) were greater than those from the pre-assessment (S1) within both groups. Students from the intervention group—Blocks4DS (n=44), performed significantly better on the post-assessment compared to the pre-assessment (p<0.01). Likewise, for those students who were part of the control group—pseudo-code (n=39), S2 was significantly higher than S1 (p<0.01).

B. *Students' Perceptions*

In general, students found Blocks4DS useful to learn DS&A (1—not useful at all, 7—very useful: M = 5.2, SD = 1.5, Mo = 6), and engaging (1—not engaging at all, 7—very engaging: M = 5.3, SD = 1.1, Mo = 5). They reported to find the block-based canvas useful (1—not useful at all, 7—very useful: M = 5.3, SD = 1.5, Mo = 5), as also the visual representation of the BST (1—not useful at all, 7—very useful: M = 6.1, SD = 1.0, Mo = 7). As it is observed, the visual representation of the BST was scored higher, as it was also the students' preferred feature (1—block-based canvas, 7—data structure visual representation: M= 5.1, SD = 1.7, Mo = 6).

From the demographic groups, White students generally preferred pseudo-code over Blocks4DS (1—Blocks4DS, 7—pseudo-code: M = 5.4, SD = 1.4, Mo = 6), as they also were the group of students with the lowest mode (Mo) in regards to the block-based canvas usefulness (1—not useful at all, 7—very useful: M = 4.6, SD = 1.8, Mo = 3). This was different from the Female students' group, who reported the highest score on the block-based canvas' usefulness (1—not useful at all, 7—very useful: M = 5.7, SD = 1.2, Mo = 7).

Although Blocks4DS was well received by the target audience, students reported to prefer using pseudo-code (1—Blocks4DS, 7—pseudo-code: M = 4.5, SD = 1.8, Mo = 6). However, this preference varies when comparing two particular groups: native English-speakers (1—Blocks4DS, 7—pseudo-code: M = 4.7, SD = 1.8, Mo = 6) and non-native English-speakers (1—Blocks4DS, 7—pseudo-code: M = 3.8, SD = 1.8, Mo = 2). While native English-speakers preferred pseudo-code, non-native English-speakers were more inclined towards Blocks4DS (see Figure 2). It is believed that their perceptions reflect the language barrier that might exist for text-based programming languages. However, this is only a hypothesis, and further research is required to evaluate that claim. Overall, Female students (1—not useful at all, 7—very useful: M = 5.7, SD = 1.1, Mo = 5) and non-native English-speakers highly rated the usefulness of the block-based canvas.

V. DISCUSSION

Based on the findings and results presented in this paper, it is concluded that undergraduate STEM-major students were able to learn about the Binary Search Tree (BST) with Blocks4DS. As presented in Section IV, students who were introduced to the BST with the tool performed significantly better for the post-assessment than they did for the pre-assessment (see table III). These findings suggest that block-based environments can assist in undergraduate Computer Science (CS) courses as Data structures & Algorithms (DS&A).

Additionally, despite the improvement observed within the intervention group (table III), there was no significant difference observed between the control and intervention groups (table 2). The authors believe these findings are due to the prior experience students had with text-based programming languages: they were previously exposed to languages like Java and C++ in CS1 and CS2. Nevertheless, it is important to recall that students reported little to none experience with BBP environments prior to the study (see Section III.A). The lack of significant difference between groups also suggests that undergraduate Computing-major students can learn about DS&A with Blocks4DS, as they are used to with text-based instruction.

The study found Blocks4DS was rated as engaging and useful for learning about BST. These findings also suggest that students self-identified as non-native English-speakers respond more positively to features such as the block-based canvas, as their demographic group was the only one that preferred Blocks4DS over pseudo-code. It was also observed that Female students rated the block-based canvas usefulness highly. This was true regardless of students' prior experience with text-based programming languages like Java and C++. Hence, Blocks4DS can be a good tool to support diversity in CS courses.

VI. Future Work

Following this study, and using the design concept of Blocks4DS, a wide horizon for future work appears. Given how positively Blocks4DS was perceived in the instruction of the BST, future studies can explore using a BBP environment to teach graphs, or more advanced data structures (e.g., Fibonacci heaps, Radix trees). Moreover, from a Human-Computer Interaction (HCI) perspective, future studies can involve affordances and features' elucidation for non-novice CS students. BBP technologies and visual languages have not been fully explored in undergraduate contexts, meaning research and design opportunities on how to build tools like Blocks4DS for upper-level courses.

For this study, the authors conducted their analysis concerning one of many characteristics, at a time, to group participants. The sample population was targeted with three primary independently considered demographic lenses: gender, race-ethnicity, and native language. Future research will consider intersectionality between characteristics to understand how students may perceive educational environments like Blocks4DS: For example, there might be different perceptions of white women than non-white women.

This study found positive perceptions coming from non-native English speakers. Future research will also explore how visual technologies like Blocks4DS can leverage language barriers with text-based programming languages.

The authors believe their research can help students in the understanding of advanced CS concepts, as it can help broadening CS to those who are not from English-centric locations.

VII. Limitations

The study was designed to be asynchronous and online due to the pandemic caused by COVID-19. Although the data collection went through with no issues, the authors believe that the lack of interaction with their participants limited their observations on how participants interacted with the tool and responded to the proposed assessments. Nevertheless, the authors were limited due to the circumstances existing nowadays.

Acknowledgment

The authors extend their gratitude to the students who actively participated in this study; to Mr. Dhrubo Paul for taking part in the early brainstorm on the methodological design; and to Mr. Joshua Fox for helping in the recruitment of participants from his courses. Finally, the authors also thank the CS Faculty and Doctoral scholars who assisted in the revision of the video-lectures used in the study.

References

[1] D. F. Almanza-Cortés, M. F. Del Toro-Salazar, R. A. Urrego-Arias, P. G. Feijóo-García, and F. De la Rosa-Rosero, "Scaffolded Block-based Instructional Tool for Linear Data Structures: A Constructivist Design to Ease Data Structures' Understanding," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, no. 10, p. 161, 2019.

[2] B. V. Almende, "vis.js community edition *," vis.js. [Online]. Accessed on: Dec. 15, 2020. Available: http://visjs.org/.

[3] P. Bille and I.L. Gørtz, "Immersive Algorithms: Better Visualization with Less Information," Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education 2017, pp. 80-81.

[4] J. Blanchard, C. Gardner-McCune and L. Anthony, "Effects of Code Representation on Student Perceptions and Attitudes Toward Programming," 2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2019, pp. 127-131, doi: 10.1109/VLHCC.2019.8818762.

[5] J. Blanchard, C. Gardner-McCune, and L. Anthony, "Dual-Modality Instruction and Learning," Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 2020, pp. 818-824, doi: 10.1145/3328778.3366865

[6] S. Buchanan, B. Ochs, and J. LaViola Jr J., "CSTutor: a pen-based tutor for data structure visualization," *Proceedings of the 43rd ACM technical symposium on Computer Science Education* 2012, pp. 565-570.

[7] T.A. Budd, "An active learning approach to teaching the data structures course," Proceedings of the 37th SIGCSE technical symposium on Computer science education 2006, pp. 143-147.

[8] D. Burlinson, M. Mehedint, C. Grafer, K. Subramanian, J. Payton, P. Goolkasian, M. Youngblood, and R. Kosara, "BRIDGES: A system to enable creation of engaging data structures assignments with real-world data and visualizations," Proceedings of the 47th ACM Technical Symposium on Computing Science Education 2016, pp. 18-23.

[9] J.H. Cross, T.D. Hendrix, J. Jain, and L.A. Barowski, "Dynamic object viewers for data structures," Proceedings of the 38th SIGCSE technical symposium on Computer science education 2007, pp. 4-8.

[10] H. Danielsiek, W. Paul, and J. Vahrenhold, "Detecting and understanding students' misconceptions related to algorithms and data structures," Proceedings of the 43rd ACM technical symposium on Computer Science Education 2012, pp. 21-26.

[11] P. G. Feijóo-García, S. Wang, J. Cai, N. Polavarapu, C. Gardner-McCune and E. D. Ragan, "Design and evaluation of a scaffolded block-based learning environment for hierarchical data structures," *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2019, pp. 145-149, doi: 10.1109/VLHCC.2019.8818759.

[12] N. Fraser, "Ten things we've learned from Blockly," 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond) 2015, pp. 49-50.

[13] D. Galles, "Data Structure Visualizations," *Data Structure Visualization*. [Online]. Available: https://www.cs.usfca.edu/~galles/visualization/.

[14] S. Grissom, M.F. McNally, and T. Naps, "Algorithm visualization in CS education: comparing levels of student engagement," Proceedings of the 2003 ACM symposium on Software visualization 2003, pp. 87-94.

[15] P.J. Guo, "Online python tutor: embeddable web-based program visualization for cs education," Proceeding of the 44th ACM technical symposium on Computer science education 2013, pp. 579-584.

[16] M. Guzdial, "Learner-centered design of computing education: Research on computing for everyone," Synthesis Lectures on Human-Centered Informatics, Vol. 8, 6 2015.

[17] C.D. Hundhausen, S.A. Douglas, and J.T. Stasko, "A meta-study of algorithm visualization effectiveness," Journal of Visual Languages & Computing, Vol. 13, 3 2002, pp. 259-290.

[18] K. Karpierz and S.A. Wolfman, "Misconceptions and concept inventory questions for binary search trees and hash tables," Proceedings of the 45th ACM technical symposium on Computer science education 2014, pp. 109-114.

[19] R. Lister, "COMPUTING EDUCATION RESEARCH Programming, syntax and cognitive load," ACM Inroads, Vol. 2, 2 2011, pp. 21-22.

[20] A. Luxton-Reilly, I. Albluwi, B.A. Becker, M. Giannakos, A.N. Kumar, L. Ott, J. Paterson, M.J. Scott, J. Sheard, and C. Szabo, "Introductory programming: a systematic literature review," *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* 2018, pp. 55-106.

[21] I.S. MacKenzie, Human-computer interaction: An empirical research perspective, Newnes, 2012.

[22] D.J. Malan and H.H. Leitner, "Scratch for budding computer scientists," *ACM Sigcse Bulletin*, Vol. 39, 1 2007, pp. 223-227.
[23] 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, 4 2010, pp. 1-15.
[24] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, "Learning computer science concepts with scratch," Computer Science Education, Vol. 23, 3 2013, pp. 239-264.
[25] P. Moraes and L. Teixeira, "Willow: A Tool for Interactive Programming Visualization to Help in the Data Structures and Algorithms Teaching-Learning Process," Proceedings of the XXXIII Brazilian Symposium on Software Engineering 2019, pp. 553-558.
[26] T.L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and others, "Exploring the role of visualization and engagement in computer science education,", 2002, pp. 131-152.
[27] T.L. Naps, J.R. Eagan, and L.L. Norton, "JHAVÉ—an environment to actively engage students in Web-based algorithm visualizations," Proceedings of the thirty-first SIGCSE technical symposium on Computer science education 2000, pp. 109-113.
[28] L. Porter, D. Zingaro, C. Lee, C. Taylor, K.C. Webb, and M. Clancy, "Developing course-level learning goals for basic data structures in CS2," Proceedings of the 49th ACM technical symposium on Computer Science Education 2018, pp. 858-863.
[29] L. Porter, D. Zingaro, S.N. Liao, C. Taylor, K.C. Webb, C. Lee, and M. Clancy, "BDSI: A validated concept inventory for basic data structures," Proceedings of the 2019 ACM Conference on International Computing Education Research 2019, pp. 111-119.
[30] T.W. Price, Y. Dong, and D. Lipovac, "iSnap: towards intelligent tutoring in novice programming environments," Proceedings of the 2017 ACM SIGCSE Technical Symposium on computer science education 2017, pp. 483-488.
[31] A. Repenning, "Moving beyond syntax: Lessons from 20 years of blocks programing in AgentSheets," Journal of Visual Languages and Sentient Systems, Vol. 3, 1 2017, pp. 68-89.
[32] C.A. Shaffer, M. Akbar, A.J.D. Alon, M. Stewart, and S.H. Edwards, "Getting algorithm visualizations into the classroom," Proceedings of the 42nd ACM technical symposium on Computer science education 2011, pp. 129-134.
[33] C.A. Shaffer, M. Cooper, and S.H. Edwards, "Algorithm visualization: a report on the state of the field," Proceedings of the 38th SIGCSE technical symposium on Computer science education 2007, pp. 150-154.
[34] C.A. Shaffer, M.L. Cooper, A.J.D. Alon, M. Akbar, M. Stewart, S. Ponce, and S.H. Edwards, "Algorithm visualization: The state of the field," ACM Transactions on Computing Education (TOCE), Vol. 10, 3 2010, pp. 1-22.
[35] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: students' perceptions of blocks-based programming," Proceedings of the 14th international conference on interaction design and children 2015, pp. 199-208.
[36] D. Weintrop and U. Wilensky, "Bringing blocks-based programming into high school computer science classrooms," Annual Meeting of the American Educational Research Association (AERA), Washington DC, USA 2016.
[37] D. Zingaro, C. Taylor, L. Porter, M. Clancy, C. Lee, S. Nam Liao, and K.C. Webb, "Identifying student difficulties with basic data structures," Proceedings of the 2018 ACM Conference on International Computing Education Research 2018, pp. 169-177.

Pedro G. Feijóo-García received the B.Sc. degrees in systems & computing engineering (2013) and mechanical engineering (2014), and the M.Sc. degree in systems & computing engineering (2015) from Universidad de los Andes, Colombia. He also received a Grad. Cert. in university teaching from Universidad El Bosque, Colombia, in 2018.
He is currently a Core-Faculty Assistant Professor at the Program of Systems Engineering, College of Engineering, at Universidad El Bosque, Colombia. He is also a Fulbright Doctoral Scholar pursuing his doctoral studies in human-centered computing at the University of Florida, U.S.A., under the supervision of Prof. Benjamin Lok, Ph.D. in the Virtual Experiences Research Group (VERG). His current research interests include intelligent virtual agents, culturally relevant computing, and computer science education.

Amanpreet Kapoor received the B.Tech. degree in computer science and engineering (2015) from Jaypee University of Engineering and Technology, India, and his M.Sc. in computer science (2016) from University of Florida, U.S.A.
He is currently an Instructional Assistant Professor of Computer Science at the Department of Engineering Education and an Affiliate Instructional Assistant Professor in the Computer and Information Science and Engineering Department, Herbert Wertheim College of Engineering, University of Florida, U.S.A. He is also a Research Affiliate at the Engaging Learning Lab where he works with Dr. Christina Gardner-McCune on projects that aim to foster students' formation of computing identities and improve the employability of computing graduates. His current research interests include computing education, informal learning environments, and identity formation.

Christina Gardner-McCune received the B.Sc. degree in computer engineering (2002) from Syracuse University, U.S.A., the M.Sc. degree in computer science (2005), and the Ph.D. in computer science (2011) from the Georgia Institute of Technology, U.S.A.
She is currently an Associate Professor in the Computer and Information Science and Engineering Department, Herbert Wertheim College of Engineering, University of Florida, U.S.A. She is the director of the Engaging Learning Lab and leads projects on computer science education research, artificial education research, educational technology design, curriculum development, and computer science professional identity development.

Eric Ragan received the B.Sc. degrees in mathematics and computer science (2007) from Gannon University, U.S.A., the M.Sc. degree in computer science and applications (2010), the Grad. Cert. in human-computer interaction (2011) and the Ph.D. in computer science (2013) from Virginia Tech, U.S.A.
He is currently an Assistant Professor in the Computer and Information Science and Engineering Department, Herbert Wertheim College of Engineering, University of Florida, U.S.A. He is the director of the Interactive Data and Immersive Environments (INDIE) Lab and leads projects focused on the design and evaluation of applications and techniques that support effective interaction and understanding of data, information, and virtual environments. His current research interests include information visualization, virtual reality, 3D interaction, visual analytics, and explainable artificial intelligence (AI).