The Impacts of a Constructionist Scratch Programming Pedagogy on Student Achievement with a Focus on Gender

Oladele O. Campbell
Niger State Polytechnic
Zungeru, Nigeria
51898772@mylife.unisa.ac.za

Oluwatoyin Adelakun-Adeyemo
Bingham University
Nasarawa, Nigeria
toyin@sure-impact.com

Fatimah Yetunde Akinrinola
Lagos State University of Education
Lagos, Nigeria
akinrinolafy@lasued.edu.ng

Patience Chewachong Akih
North West University
Potchefstroom, South Africa
patmeninkele@gmail.com

Ethel Tshukudu
University of Botswana
Gaborone, Botswana
tshukudue@ub.ac.bw

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT

Learning to program is a challenge for many novice computing students. This may be partially due to the inadequacy of many conventional pedagogical approaches resulting in dropouts and failure, especially among females and minoritized students. Consequently, more effective methods to increase success rates and effectively broaden participation are needed. We investigated the effectiveness of a constructionist programming pedagogy with college-level CS1 students, and how that effectiveness varies by gender. Our quasi-experimental design includes participants from CS1 classes at several polytechnics in Nigeria who were assigned to either experimental or control groups receiving six weeks of constructionist instruction using Scratch and a more conventional approach. Findings indicated that the constructionist cohorts had significantly higher mean post-test scores than the conventional classes. We observed no significant difference in post-test scores between genders. However, a significant gender difference was found in gain scores within one study where we also observed a significant relationship between gender and academic background. Overall, our approach demonstrated positive effects on the programming achievements of novice programming students, emphasizing the need to consider the role of contextual factors such as academic background and regional disparities across gender differences.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **Computer science education**; **CS1**.

KEYWORDS

Constructionism; Constructivism; CS1; Introductory Programming; Novice Programmers; Gender; Scratch

ACM Reference Format:
Oladele O. Campbell, Oluwatoyin Adelakun-Adeyemo, Fatimah Yetunde Akinrinola, Patience Chewachong Akih, Ethel Tshukudu, and Brett A. Becker. 2023. The Impacts of a Constructionist Scratch Programming Pedagogy on Student Achievement with a Focus on Gender. In *Proceedings of the ACM Conference on Global Computing Education Vol 1 (CompEd 2023), December 5–9, 2023, Hyderabad, India.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3576882.3617911

CompEd 2023, December 5–9, 2023, Hyderabad, India

ACM ISBN 979-8-4007-0048-4/23/12.
https://doi.org/10.1145/3576882.3617911

1 INTRODUCTION

Programming holds a central role in computing education curricula. Extensive research has been dedicated to assessing the effectiveness of programming education, often coupled with the common perception that learning programming poses challenges [4, 25]. Concurrently, a notable proportion of students encounter difficulties and experience failures in their initial programming course [5]. However, such outcomes align with those observed in introductory courses across diverse STEM disciplines [24, 41]. Despite this comparative context, the drive to enhance programming learning outcomes remains a cornerstone objective for computing educators.

Diverse programming pedagogies have been studied, revealing positive impacts in K-12 and higher education contexts [28, 32]. Yet, direct instruction persists as a dominant instructional approach in programming education. Notably, the empirical substantiation of novices effectively acquiring programming skills through a constructionist approach is limited [23]. Addressing this gap, this study investigates the ramifications of a constructionist approach on novice programming education, with an additional focus on its implications for gender dynamics.

1.1 Research Questions

Our research is framed by the following research questions:

- RQ1 What is the effect of a constructionist Scratch programming pedagogy on college CS1 students' achievement in programming, as measured by post-test scores, after controlling for a pre-test?
- RQ2 How do post-test scores for the constructionist (Scratch) approach vary with gender?

The following sections explore the theoretical frameworks of constructivism and Papertian constructionism. We then provide a literature review on Scratch in higher education CS1courses and gender effects in courses using Scratch. Following that, we detail our methodology, including participant demographics, data collection procedures, and analysis methods. Next, we present the study's

outcomes in alignment with the research questions. Subsequently, we discuss these findings, relating them to existing literature and offering recommendations for further research within the computing education community. We also acknowledge the study's limitations before concluding thoughts.

2 RELATED WORK

2.1 Theoretical Framework

The constructivist pedagogy is based on the theory that students come to class with prior knowledge and experience which serve as scaffolds for building new knowledge [13]. Therefore, the constructivist classroom provides opportunities for students to engage with content and to draw meaning and understanding by relating that content to existing knowledge. Very important in the constructivist theory is the role of previous experience (sensory and social) in creating knowledge. However, there has not been much research into whether a constructivist pedagogy helps to close the gender gap in learning programming.

Constructionism is an educational theory developed by Seymour Papert and emphasizes students' hands-on, experiential learning through building, creating, and sharing artefacts with peers [30, 36]. Papert noted that students should be active participants in their own learning, constructing knowledge through their interactions with the world around them [31]. This closely relates to constructivism. [19]. Proponents of constructionism argue that it can be an effective approach to teaching as it allows students to take ownership of their learning, promotes creativity, problem-solving skills, and a deep understanding of concepts [18, 22]. Papert believed that technology, particularly computers, could be a powerful tool for constructionist learning, enabling students to design and create their own programs and simulations [31]. While Papert's vision of constructionist learning has always included programming, many college programming classes utilise direct instruction. There is also limited empirical evidence of novices learning to program following a constructionist approach [23]. The value of this approach could be expansive. For instance, constructivism has been tied to metacognition [15], an area that can be beneficial to learning in general, and in learning programming [12].

However, critics have raised concerns about the practicality and accessibility of a constructionist approach, particularly in under-resourced schools [2, 21]. Another concern is that constructionism may not be suitable for all students; some may prefer to learn in a more traditional, teacher-directed way [35]. Additionally, some students may lack the necessary skills or resources to participate in constructionist learning activities [46]. Some have also argued that the emphasis on building and creating may not be suitable for all subjects or learning objectives [21]. These concerns necessitate studies providing empirical evidence relating to the constructionist approach and confirming whether gender differences exist in its impact on students [2].

2.2 Scratch in CS1

The literature on Scratch, a block-based programming environment, largely presents a collection of studies showcasing its effectiveness as a teaching tool. These studies consistently report positive outcomes, emphasizing Scratch's capacity to enhance student motivation, improve performance, and nurture higher-order thinking skills, particularly in K-12 educational settings [6, 17, 33, 42].

Moreover, Scratch has found its way into higher education, particularly CS1. Becker's [3] survey of CS1 courses in Ireland revealed that Scratch was employed in 8% of the courses alongside more prevalent languages like Java, Python, JavaScript, C, and C#. This represents a non-trivial use of Scratch in higher education and exemplifies its adaptability as an introductory programming tool. There is some empirical evidence of Scratch's positive impact in higher education classes, For instance, Hijón-Neira et al. [16] introduced a Scratch-based learning tool to first-year computer science students in a Spanish university, observing significant improvements across various programming concepts. Similarly, Cárdenas-Cobo et al. [10, 11] observed positive outcomes of Scratch-based pedagogy on students' programming learning experience and performance in an Ecuadorian college setting. Furthermore, Papadakis and Kalogiannakis [29] successfully introduced Computational Thinking and programming to early childhood education at the University of Crete through a Scratch-based introductory programming course for preservice kindergarten teachers, with results surpassing expectations. While these findings underscore the versatility of Scratch in fostering a constructive learning environment, there exists a noteworthy contrast in outcomes from studies conducted in higher education. For instance, Martinez-Valdes et al. [26] introduced Scratch to university students majoring in gaming. However, students' transition from Scratch to Java yielded mixed results, highlighting the need for a deeper examination of the effectiveness of Scratch in college-level contexts. These studies collectively suggest that Scratch holds promise for higher education settings but also highlight the need for further exploration

2.3 Gender Differences in Scratch

Research on gender differences in Scratch programming has shown significant disparities, especially given the underrepresentation of females in the field [13, 45]. Studies by Tsan et al. [44] revealed lower program quality among female 5th graders compared to males. Conversely, Tan et al. [43] found reduced gender gaps using a STEAM approach. It has been observed that girls can prefer simpler projects, while boys gravitate towards complexity, emphasizing loops and more complex structures [14, 40]. Despite this, some studies have found no gender distinctions, particularly in early childhood education [20, 34]. Interestingly, there is a research gap in understanding gender dynamics in college-level Scratch programming classes – possibly due to lower prevalence – indicating a need for inclusive strategies in this context.

3 METHODOLOGY

In order to investigate the effectiveness of a constructionist Scratch programming pedagogy with college-level introductory programming (CS1) students, we conducted a study over two academic years with CS1 classes from public polytechnic colleges in two north-central Nigerian states. We conducted a quasi-experimental study using a non-equivalent control group design. Two groups were formed: the experimental group used a constructionist Scratch

approach, while the control group used a conventional approach using Visual Basic. To minimize bias, we employed the Coarsened Exact Matching (CEM) algorithm to pre-process unmatched data as described below. Descriptive and inferential statistics were used to analyze the matched data. Ethical approval was obtained from the institutions, and participants provided informed consent before data collection.

3.1 Participants

Study 1: This study involved 236 CS1 students from two federal Polytechnic institutions in north-central Nigeria. After employing CEM on unmatched data, the study generated 82 cases (41 in the Experimental group and 41 in the Control group). Details are shown in Table 1. The majority of participants (68.3%) were male. A significant portion (58.5%) fell within the age range of 19-21. Most participants (78%) were at a low academic level. The academic level was calculated from their previous O'Level exam grades in English, Mathematics, and Physics, and their Unified Tertiary Matriculation Examination scores. Notably, none of the participants had (self-reported) experience in programming.

Study 2: This study included two groups of CS1 students (n=182) selected from a federal and state polytechnic college situated in the same region as Study 1. The matched data generated 84 cases (42 in the Experimental group and 42 in the Control group). Details are shown in Table 2. The majority of participants were male (78.6%) with most falling within the age range of 19-21 (40.5%). In terms of academic level, 76.2% had a low level and 23.8% had an average level. Again, none of the participants had prior programming experience. Both study 1 and 2 were taught by the same instructor.

3.2 Procedure

Pre-test The participants in both studies were given the language-independent Introductory Programming Achievement Test (IPAT1, rb.gy/2aft8) prior to interventions.

Constructionist Scratch approach The Experimental groups in both studies learned programming for six weeks using the Constructionist Scratch approach. Students used Scratch to develop projects of interest. The instructor provided guidance but emphasized student exploration and collaboration. Real-life programming problems were given, and students worked in groups to create diverse projects like reciting the Nigerian national anthem, games, and animated stories. The Scratch group did not attend lectures but had demonstrations and lab sessions.

Conventional pedagogical approach In both control groups, programming was taught using the conventional pedagogical approach based on the Nigerian National Board for Technical Education (NBTE) curriculum. The focus was on teaching programming concepts and Visual Basic language basics over six weeks. Like the Scratch group, these students had weekly lab sessions and programming assignments. In the labs, they developed Visual Basic programs individually, without weekly demonstrations. In addition, control groups also attended interactive two-hour lectures and received worked examples in class.

Post-test The participants in both studies were given the language-independent Introductory Programming Achievement Test (IPAT2, rb.gy/2aft8) after the interventions.

3.3 Data Collection and Analysis

Data collected comprised student profiles and pre/post achievement test scores. Questionnaires and achievement tests were administered and marked by a computing educator using a specific rubric available at rb.gy/2aft8. Microsoft Excel and SPSS version 23 were used for data analysis. The Coarsened Exact Matching (CEM) program, an SPSS add-in, generated the matched experimental and control datasets. Groups were matched using the following variables: pre-test scores, age, gender, prior academic level and programming experience. The data were analyzed employing descriptive statistics (mean and standard deviation), as well as inferential statistics (independent samples t-test and ANCOVA) at a significance level of 0.05. For more information on the CEM, power analysis and assumption tests conducted for the ANCOVA and t-tests, please see [9, 37].

4 RESULTS

4.1 Pre-test

Pre-test results for both Study 1 and Study 2 are presented in Table 3. Before matching, the experimental group outperformed the control group 24.8 to 21.2 in Study 1, while the control group outperformed the experimental group 18.7 to 13.4 in Study 2. However, after matching, there were no longer significant differences in pre-test achievement scores between the control and experimental groups in both studies, indicating that the matched samples were similar. These results suggest that any differences in post-test scores between the two groups are likely to be attributable to the treatment and not to pre-existing differences in achievement levels.

4.2 RQ1: Effect of Pedagogy on Achievement

Research question 1 was: *What is the effect of a constructionist Scratch programming pedagogy on college CS1 students' achievement in programming, as measured by post-test scores, after controlling for a pre-test?*. Matched samples data from the two studies were employed to answer this research question, the results of which are presented below.

Study 1 Results: Descriptive statistics showed that the experimental group had a higher mean (M = 49) than the control group (M = 40) in IPAT post-test scores. The ANCOVA model revealed that the treatment (pedagogy) had a significant effect on post-test scores after controlling for pre-test scores ($F(1, 80) = 8.159$, $p = .005$, $\eta 2 = .094$). Partial eta-squared indicated that the experimental group accounted for 9.4% of the variance in the post-test scores. Moreover, the adjusted R2 indicated that the model accounted for 24.8% of the variance. The estimates revealed that the experimental group had significantly higher post-test scores than control ($\beta = 9.513$, SE = 2.342, $p = .001$, 95% CI [4.834, 14.193]).

Study 2 Results: Descriptive statistics showed that the experimental group had a higher mean (M = 28) than the control group (M = 23) in IPAT post-test scores. The ANCOVA model revealed that the experimental group had a significant effect on post-test scores after controlling for pre-test scores ($F(1, 82) = 7.396$, $p = .008$, $\eta 2 = .084$). The partial eta-squared indicated that the experimental group accounted for 8.4% of the variance in the post-test scores. Moreover, the adjusted R2 indicated that the model accounted for 28.5% of the variance. The estimates revealed that the experimental

Secondary Independent Variables		Primary Independent Variable (Treatment Variable)							
		Constructionist (Scratch) Class				Conventional Class			
		N	%	Mean pre-test	Mean post-test	N	%	Mean pre-test	Mean post-test
Gender	Male	28	68.3	22.2	48.7	28	68.3	22.3	38.9
	Female	13	31.7	20.8	49	13	31.7	21.5	41.2
Age	16-18	4	9.8	27.5	54.5	4	9.8	27.5	51.5
	19-21	24	58.5	21.2	46.8	24	58.5	22	36.6
	22-24	12	29.3	20.5	52.2	12	29.3	20	42.8
	>24	1	2.4	28	36	1	2.4	26	26
Prior Academic Level	Low	32	78	21.8	48	32	78	21.8	39.7
	Average	9	22	21.6	52	9	22	23.1	39.3
	High	0	0	-	-	0	0	-	-
Prior Program Writing	None	41	100	21.8	48.9	41	100	22.1	39.6
	Some	0	0	-	-	0	0	-	-
	Total	41	100.0	21.8	48.9	41	100.0	22.1	39.6

Table 1: Demography and performance of matched samples (study1 participants: 82 students total)

Secondary Independent Variables		Primary Independent Variable (Treatment Variable)							
		Constructionist (Scratch) Class				Conventional Class			
		N	%	Mean pre-test	Mean post-test	N	%	Mean pre-test	Mean post-test
Gender	Male	33	78.6	13.3	29.4	27	64.3	16.6	22.8
	Female	9	21.4	16	24.8	15	35.7	10.2	24.5
Age	16-18	6	14.3	14.3	30	6	14.3	12.5	25.7
	19-21	17	40.5	13.9	29.4	17	40.5	15.4	24
	22-24	17	40.5	13.1	25.5	16	38.1	12.5	21.4
	>24	2	4.8	17.3	35.7	3	7.1	25	29
Prior Academic Level	Low	32	76.2	13	28.3	32	76.2	13.6	23.1
	Average	10	23.8	16.6	29	10	23.8	16.4	24.6
	High	0	0	-	-	0	0	-	-
Prior Program Writing	None	41	97.6	13.4	28	41	97.6	13.4	28
	Some	1	2.4	32	45	1	2.4	32	42
	Total	42	100.0	13.9	28.5	42	100.0	14.3	23.4

Table 2: Demography and performance of matched samples (study2 participants: 84 students total)

		Pre-test		Matched sample	
		Study 1	Study 2	Study 1	Study 2
Class	Conventional	21.2	18.7	22	14.3
	Constructionist (Scratch)	24.8	13.4	21.8	13.9
	Difference	-3.6	5.2	0.3	0.4
	p	**0.035**	**<0.001**	0.886	0.828

Table 3: Groups' mean pre-test performance before and after matching. Significant results are in bold

group had significantly higher post-test scores than the control group (β = 5.247, SE = 1.364, p = .001, 95% CI [2.542, 7.952]).

The results of the two studies revealed that a constructionist Scratch programming pedagogy has a significant effect on college CS1 students' achievements in programming, as measured by the IPAT post-test scores, after controlling for pre-test scores. The experimental group, which received instruction based on this pedagogy, outperformed the control group, which received conventional instruction, in both studies. These findings support a hypothesis that a constructionist Scratch programming pedagogy can result in higher achievement for novice college-level CS1 students compared to traditional direct instruction.

4.3 RQ2: Gender Differences in Scratch

Research question 2 was: *How do post-test scores for the constructionist (Scratch) approach vary with gender?.*

To investigate gender differences in constructionist Scratch programming achievements, we employed data from the two studies (Study 1 and Study 2). Study 1 included 28 male and 13 female participants from a federal polytechnic, while Study 2 had 33 male and 9 female participants from a state polytechnic. An independent samples t-test was used to compare the means of male and female participants' pre-test, post-test, and gain scores (post-test minus pre-test) from each study. Table 4 presents the results of the t-tests.

Study	Variable	Male Mean (SD)	Female Mean (SD)	t-value	p-value	95% CI	Cohen's d
Study 1	Pre-test	22.2 (9.5)	20.8 (10.2)	0.444	0.660	[-5.1411, 8.0312]	0.15
	Post-test	48.7 (16.1)	49.2 (16.6)	-0.095	0.925	[-11.5625, 10.5295]	-0.03
	Gain scores	26.5 (16.5)	28.5 (14.2)	-0.369	0.714	[-12.7084, 8.7853]	-0.12
Study 2	Pre-test	13.3 (8.4)	16.0 (9.0)	-0.844	0.404	[-9.156, 3.762]	-0.32
	Post-test	29.4 (8.3)	24.8 (8.8)	1.470	0.149	[-1.741, 11.034]	0.55
	Gain scores	16.1 (8.0)	8.8 (7.0)	2.493	0.017	[1.389, 13.298]	0.976

Table 4: Gender Differences in Scratch Programming Achievements (t-test results)

Study 1 Results: Pre-test scores showed no significant gender difference (p = 0.660, Cohen's d = 0.15). The mean post-test scores were slightly higher for females, but the difference was not significant (p = 0.925, Cohen's d = -0.03). Similarly, there was no significant difference in gain scores between male and female participants (p = 0.714, Cohen's d = -0.12). These results suggest that gender did not significantly impact programming achievements in Study 1.

Study 2 Results: The mean pre-test score was higher for females than males, but the difference was not significant (p = 0.404, Cohen's d = -0.32). However, a significant difference was found in gain scores (p = 0.017, Cohen's d = 0.976), indicating that males showed more improvement in programming achievement compared to females. Post-test scores, conversely, exhibited no significant gender difference (p = 0.149, Cohen's d = 0.55).

Looking for a possible explanation for the mixed results (in the gain scores) above, we conducted a chi-squared test of the relationship between gender and academic background of students in the Scratch classes from both studies. The chi-squared test examined the association between gender and academic background in each study. In Study 1, the results indicated no significant association between gender and academic background (p = 0.906), while in Study 2, a significant association was observed (p = 0.012). These findings suggest that the distribution of academic backgrounds differs based on gender in Study 2 but not in Study 1.

The mixed findings suggest gender-programming achievement varies by study-specific factors. Study 1 (federal polytechnic) found no gender differences, while Study 2 (state polytechnic) showed differences. Study 2 had notable gender-academic background variation. One hypothesis is that the constructionist Scratch approach may impact low-achieving males more than females with higher academic backgrounds. Understanding gender/academic-instruction dynamics is vital for programming education. Future research should explore these factors and their impact on constructionist Scratch pedagogies. Further, gender differences may vary, highlighting the need for inclusive approaches in programming education.

5 DISCUSSION

This study examined the effect of a constructionist Scratch programming pedagogy on college CS1 students' achievements in programming, and how that varies by gender.

5.1 Effects on Students' Achievements

Our results indicate that the constructionist Scratch programming pedagogy improved students' programming achievement, and that gender may have played a role in the students' engagement with Scratch programming and their achievements. However, more research on this front is necessary. The constructivist approach is based on the theory that students' prior knowledge and experiences can serve as scaffolds to build new knowledge. This study showed that constructivist pedagogy improved students' programming achievement, supporting previous studies that have found similar approaches to be effective in programming education [10, 27, 39, 47]. These results also support the hypothesis that students' previous experience, both sensory and social, may play an important role in creating knowledge. Therefore, providing opportunities for students to engage with content and to relate it to their existing knowledge may be critical for effective learning. Constructionism extends Piagetian constructivism emphasizing the need for experiential learning, students' free agency and collaborative learning. The result of this study confirms the possibility of improving CS1 learning outcomes by employing a constructionist pedagogy using Scratch, contributing empirical evidence of its positive impact [1, 30]. The results of this study add to the existing literature that shows that Scratch is an effective tool for teaching programming concepts, even at the college level [7, 8, 11, 48].

5.2 Gender and Achievements in Scratch

This section explores the intricate relationship between gender dynamics and programming achievements in the context of constructionist Scratch programming. Using results from Section 4.3, we examine the influence of gender on achievement and illuminate nuanced interactions. Our investigation reveals distinct patterns across two independent studies. In Study 1, conducted at a federal polytechnic in north-central Nigeria, gender differences did not significantly impact programming achievements, aligning with the absence of a substantial association between gender and academic background. Conversely, Study 2, conducted at a state polytechnic within the same region, exhibited significant gender-related disparities in gain scores, indicating that variations in programming achievements may be attributed to gender. These contrasting outcomes underscore the role of regionally specific and institutionally contextual factors in shaping gender dynamics. Such mixed findings emphasize the complex nature of gender's influence on programming achievements, suggesting that the relationship is likely context-dependent.

When comparing our findings with existing literature, we observe both aligning and contrasting patterns. Our results correspond with studies that emphasize the significance of contextual factors in shaping gender-related disparities in programming achievements [38, 48]. In contrast, this study diverges with the findings of Tan et al. [43], who found that Scratch could potentially mitigate

gender disparities in programming achievements. Our nuanced findings contribute to the ongoing discourse on the contextual nature of gender dynamics in programming education.

In conclusion, our exploration of gender dynamics within constructionist Scratch programming achievements underscores the need for an inclusive and context sensitive approach. Educators and policymakers should consider the diverse array of factors that contribute to gender-related disparities in programming education. By identifying and recognizing the multifaceted nature of gender effects, the community may be able to develop targeted strategies that promote equitable programming education and create an environment where all students can excel, regardless of their gender.

5.3 Gaps and Further Research

There is a gap in the literature on the effectiveness of constructionist approaches in programming instruction for different student populations, particularly in developing countries. Future research could explore the long-term impact of our approach on student programming abilities and interests. Investigations could also be carried out to better understand the attitudes and experiences of the participants, particularly female students, and how they may have influenced their programming achievement.

Several open research questions arose from the analysis of this data, which we share here for the community:

(1) What students prefer to learn programming in a constructivist pedagogy?
(2) What students prefer to learn programming in a more traditional direct-instruction pedagogy?
(3) What skills and resources are required for students to effectively learn in a constructivist pedagogy?
(4) What are the attitudes and experiences of female students who learn programming within a constructionist programming pedagogy?
(5) What are the effects of a constructionist programming pedagogy across genders in larger and more longitudinal / multinational contexts?

6 LIMITATIONS

This study focused only on the effect of a constructionist Scratch programming pedagogy on college CS1 students' achievement in programming. Other factors that may influence programming achievements, such as socio-economic status, prior programming experience, and motivation, were not directly considered in this study. This study employed a non-equivalent control group design, and the assignment of participants to the experimental and control groups was not random which may introduce bias as factors other than the treatment may have influenced the groups' experiences. This study employed the Coarsened Exact Matching (CEM) algorithm to pre-process the non-quivalent data and match the experimental and control groups. However, matching cannot eliminate bias, and there may still be unmeasured variables that influenced the results. Our study also has limitations due to sample sizes. Our sample sizes are sufficient for ANCOVA analyses for assessing treatment effects and controlling for covariates, enhancing the reliability of results within our specific context. However, our sample sizes for t-tests, especially when examining gender differences, are relatively small and unbalanced. This may limit the precision of findings and the generalizability of results beyond our study. Therefore caution should be exercised when extending these results to broader populations or different educational contexts. There was also a limited intervention period of six weeks in this study, which may not be sufficient to fully assess the impact of these pedagogies on the programming achievement of the participants. Finally, this study used only pre- and post-tests as outcome measures, which may not fully capture the impact of the interventions on the programming achievement of the participants. Other measures, such as long-term retention and transfer of programming skills, were not assessed.

7 CONCLUSIONS

This study demonstrated the positive impact of a constructionist Scratch programming pedagogy on the programming achievement of CS1 students. Our findings support the effectiveness of constructivist approaches in programming education and emphasize the importance of leveraging students' prior knowledge and experiences as scaffolds for learning. We also highlighted the role of gender in students' engagement with Scratch programming, supporting prior research. While Scratch was shown to be an effective tool for teaching programming concepts, further investigation is needed to understand the underlying factors contributing to gender differences.

Our research contributes to the existing literature on effective programming pedagogy and the use of Scratch in CS1 programming education, providing valuable insights for educators seeking to enhance programming education and bridge the gender gap in programming achievement. By doing so, educators can create a more inclusive and effective programming education environment through a hands-on, creative, and collaborative approach to teaching programming. Therefore, students, regardless of their gender, can develop essential skills for the future while also promoting gender equity and diversity in the field of computer science. Embracing this pedagogical approach has the potential to transform education and empower students to thrive in this digital world.

ACKNOWLEDGMENTS

This work was partially funded by the Tertiary Education Trust Fund (TETFUND) and a University of South Africa MnD Bursary Grant.

REFERENCES

[1] Alaa Khalaf Al-Makhzoomy, Ke Zhang, and Timothy Spannaus. 2020. Game Development-Based Learning: A New Paradigm for Teaching Computer and Object-Oriented Programming. In *Examining Multiple Intelligences and Digital Technologies for Enhanced Learning Opportunities*. IGI Global, 244–259.
[2] Ahmed Alanazi. 2016. A Critical Review Of Constructivist Theory And The Emergence Of Constructionism. *American Research Journal of Humanities and Social Sciences* 2 (2016), 1–8.
[3] Brett A. Becker. 2019. A Survey of Introductory Programming Courses in Ireland. In *ITiCSE '19*. Association for Computing Machinery, New York, NY, USA, 58–64. https://doi.org/10.1145/3304221.3319752
[4] Brett A. Becker. 2021. What Does Saying That 'Programming is Hard' Really Say, and About Whom? *Commun. ACM* 64, 8 (jul 2021), 27–29. https://doi.org/10.1145/3469115
[5] Jens Bennedsen and Michael E Caspersen. 2019. Failure Rates in Introductory Programming-12 Years Later. *ACM Inroads* 10, 2 (2019), 30–35.

[6] Abdessamad Binaoui, Mohammed Moubtassime, and Latifa Belfakir. 2022. The Effectiveness and Impact of Teaching Coding through Scratch on Moroccan Pupils' Competencies. *International Journal of Modern Education and Computer Science* 14, 5 (oct 2022), 44–55. https://doi.org/10.5815/IJMECS.2022.05.05

[7] Oladele Oladunjoye Campbell. 2022. *Impact Of Scratch On The Achievements Of First-year Computer Science Students In Programming In Some Nigerian Polytechnics*. Ph. D. Dissertation. University of South Africa, Pretoria, South Africa.

[8] Oladele O Campbell and Harrison I Atagana. 2022. Impact Of A Scratch Programming Intervention On Student Engagement In A Nigerian Polytechnic First-year Class: Verdict From The Observers. *Heliyon* 8, 3 (2022), e09191.

[9] Oladele O. Campbell and Harrison I. Atagana. 2022. In Search Of Experimental Evidence On Scratch Programming And Students' Achievements In The First-year College Computing Class? Consider These Datasets. *Data in Brief* 45 (2022), 108635. https://doi.org/10.1016/j.dib.2022.108635

[10] Jesennia Cárdenas-Cobo, Amilkar Puris, et al. 2019. Recommender Systems And Scratch: An Integrated Approach For Enhancing Computer Programming Learning. *IEEE Trans. on Learning Technologies* 13, 2 (2019), 387–403.

[11] Jesennia Cárdenas-Cobo, Amilkar Puris, Pavel Novoa-Hernández, et al. 2021. Using Scratch To Improve Learning Programming In College Students: A Positive Experience From A Non-weird Country. *Electronics* 10, 10 (2021), 1180.

[12] Paul Denny, James Prather, Brett A. Becker, et al. 2019. A Closer Look at Metacognitive Scaffolding: Solving Test Cases Before Programming. In *Koli Calling '19*. ACM, NY, NY, USA, Article 11, 10 pages. https://doi.org/10.1145/3364510.3366170

[13] Katrina Falkner and Judy Sheard. 2019. Pedagogic Approaches. In *The Cambridge Handbook of Computing Education Research*, Fincher and Robins (Eds.). Cambridge University Press, 445–480. https://doi.org/10.1017/9781108654555.0

[14] Isabella Graßl, Katharina Geldreich, and Gordon Fraser. 2021. Data-driven Analysis of Gender Differences and Similarities in Scratch Programs. In *WiPSCE '21'*. ACM, 1–10. https://doi.org/10.1145/3481312.3481345

[15] Richard F Gunstone. 1992. Constructivism and Metacognition: Theoretical Issues and Classroom Studies. *Research in Physics Learning: Theoretical Issues and Empirical Studies* (1992), 129–140.

[16] R. Hijón-Neira, C. Connolly, D. Palacios-Alonso, et al. 2021. A Guided Scratch Visual Execution Environment to Introduce Programming Concepts to CS1 Students. *Information* 12 (2021), 378. https://doi.org/10.3390/info12090378

[17] Norizwan Ideris, Siti Mastura Baharudin, and Norhasyimah Hamzah. 2019. The Effectiveness of Scratch in Collaborative Learning on Higher-Order Thinking Skills in Programming Subject Among Year-Six Students.. In *Proceedings of the 4th ASEAN Conference on Psychology, Counselling, and Humanities (ACPCH 2018)*. 421–421. https://doi.org/10.2991/acpch-18.2019.99

[18] Yasmin B Kafai and Quinn Burke. 2015. Constructionist Gaming: Understanding The Benefits Of Making Games For Learning. *Edu Psych* 50, 4 (2015), 313–334.

[19] Yasmin B Kafai and Mitchel Resnick. 1996. *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Routledge.

[20] Kalliopi Kanaki and Michail Kalogiannakis. 2022. Assessing Algorithmic Thinking Skills in Relation to Age in Early Childhood STEM Education. *Education Sciences* 12, 6 (jun 2022). https://doi.org/10.3390/EDUCSCI12060380

[21] Paul A Kirschner, John Sweller, and Richard E Clark. 2006. Why Minimal Guidance During Instruction Does Not Work: An Analysis Of The Failure Of Constructivist, Discovery, Problem-based, Experiential, And Inquiry-based Teaching. *Edu psych* 41, 2 (2006).

[22] Chronis Kynigos. 2015. Constructionism: Theory of Learning or Theory of Design?. In *12th Int'l Congress on Mathematical Education*. Springer, 417–438.

[23] Michael Lodi, Dario Malchiodi, Mattia Monga, et al. 2019. Constructionist Attempts At Supporting The Learning Of Computer Programming: A Survey. *Olympiads in Informatics: An International Journal* 13 (2019), 99–121.

[24] Andrew Luxton-Reilly. 2016. Learning to Program is Easy. In *ITiCSE '16*. ACM, NY, NY, USA, 284–289. https://doi.org/10.1145/2899415.2899432

[25] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, et al. 2018. Introductory Programming: A Systematic Literature Review. In *ITiCSE '18 WG*. ACM, NY, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[26] J.A. Martínez-Valdés, J.Á. Velázquez-Iturbide, and R. Hijón-Neira. 2017. A (Relatively) Unsatisfactory Experience of Use of Scratch in CS1. In *TEEM '17*. Article 8. https://doi.org/10.1145/3144826.3145356

[27] Jan Moons and Carlos De Backer. 2013. The Design And Pilot Evaluation Of An Interactive Learning Environment For Introductory Programming Influenced By Cognitive Load Theory And Constructivism. *Computers & Education* 60, 1 (2013).

[28] Marimuthu Mudaray and Govender Predhayen. 2018. Perceptions Of Scratch Programming Among Secondary School Students In Kwazulu-natal, South Africa. *The African Journal of Information and Communication* 21 (2018), 51–80. https://doi.org/10.23962/10539/26112

[29] S. Papadakis and M. Kalogiannakis. 2019. Evaluating a Course for Teaching Introductory Programming with Scratch to Pre-service Kindergarten Teachers. *International Journal of Technology Enhanced Learning* 11, 3 (2019). https://doi.org/10.1504/IJTEL.2019.100478

[30] Sofia Papavlasopoulou, Michail N Giannakos, and Letizia Jaccheri. 2019. Exploring Children's Learning Experience In Constructionism-based Coding Activities Through Design-based Research. *Computers in Human Behavior* 99 (2019).

[31] Seymour A Papert. 2020. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.

[32] Hamilton Perez-Narvaez, Rosabel Roig-Vila, and Jaramillo–Naranjo Lilian. 2020. Use of Scratch in Learning Programming in Higher Education. , 28–45 pages. https://doi.org/10.29166/10.29166/catedra.v3i1.2006

[33] João Piedade and Nuno Dorotea. 2022. Effects of Scratch-based Activities on 4th-grade Students' Computational Thinking Skills. *Informatics in Education* (oct 2022). https://doi.org/10.15388/INFEDU.2023.19

[34] Colin B. Price and Ruth Price-Mohr. 2021. Exploring Gender Differences In Primary School Computer Programming Classes: A Study In An English State-funded Urban School. *Education 3-13* 51, 2 (2021), 1–14. https://doi.org/10.1080/03004279.2021.1971274

[35] Rebecca Reynolds and Iliana Hale Caperton. 2011. Contrasts In Student Engagement, Meaning-making, Dislikes, And Challenges In A Discovery-based Program Of Game Design Learning. *Educational Technology Research and Development* 59, 2 (2011), 267–289. https://doi.org/10.1007/s11423-011-9191-8

[36] Mohammad Rob and Farhana Rob. 2018. Dilemma Between Constructivism And Constructionism: Leading To The Development Of A Teaching-learning Framework For Student Engagement And Learning. *Journal of International Education in Business* 11, 2 (2018), 273–290.

[37] Kate Sanders, Judy Sheard, Brett A. Becker, et al. 2019. Inferential Statistics in Computing Education Research: A Methodological Review. In *ICER '19'*. ACM, NY, NY, USA, 177–185. https://doi.org/10.1145/3291279.3339408

[38] Despoina Schina, Vanessa Esteve González, and Mireia Usart Rodríguez. 2019. Gender Differences in Students' Feedback and Performance in Scratch Programming. In *Conference Proceedings EDUNOVATIC 2018: 3rd Virtual International Conference on Education, Innovation and ICT*. Adaya Press, 36.

[39] Sadia Sharmin, Daniel Zingaro, Lisa Zhang, et al. 2019. Impact of Open-Ended Assignments on Student Self-Efficacy in CS1. In *CompEd '19*. Association for Computing Machinery, New York, NY, USA, 215–221. https://doi.org/10.1145/3300115.3309532

[40] Kevin Sigayret, André Tricot, and Nathalie Blanc. 2022. Unplugged Or Plugged-in Programming Learning: A Comparative Experimental Study. *Computers and Education* 184, 2 (mar 2022). https://doi.org/10.1016/J.COMPEDU.2022.104505

[41] Simon, Andrew Luxton-Reilly, and Oluwatoyin Adelakun-Adeyemo. 2021. Confirmation Bias and Other Flaws in Citing Pass Rate Studies. (2021), 575–581. https://doi.org/10.1145/3430665.3456357

[42] Tarık Talan. 2020. Investigation of the Studies on the Use of Scratch Software in Education. *Journal of Education and Future* 18 (2020), 95–111. https://doi.org/10.30786/jef.556701

[43] Wee-Ling Tan, Mohd Ali Samsudin, Mohd Erfy Ismail, et al. 2020. Gender Differences In Students' Achievements In Learning Concepts Of Electricity Via STEAM Integrated Approach Utilizing Scratch. *Problems of Education in the 21st Century* 78, 3 (jun 2020), 423–448. https://doi.org/10.33225/PEC/20.78.423

[44] Jennifer Tsan, Kristy Elizabeth Boyer, and Collin F. Lynch. 2016. How Early Does the CS Gender Gap Emerge? A Study of Collaborative Problem Solving in 5th Grade Computer Science. (2016). https://doi.org/10.1145/2839509.2844605

[45] UNESCO. 2015. *UNESCO Science Report: Towards 2030*.

[46] Le Ha Van and Nguyen Long Quoc. 2022. A Systematic Review Of Constructivist Classrooms: Challenges For Teachers And Recommendations. *Journal of Positive School Psychology* 6, 8 (2022), 7216–7229.

[47] Mark J Van Gorp and Scott Grissom. 2001. An Empirical Evaluation Of Using Constructive Classroom Activities To Teach Introductory Programming. *Computer Science Education* 11, 3 (2001), 247–260.

[48] Zhen Xu, Albert D Ritzhaupt, Fengchun Tian, et al. 2019. Block-based Versus Text-based Programming Environments On Novice Student Learning Outcomes: A Meta-analysis Study. *Computer Science Education* 29, 2-3 (2019), 177–204.