Rima International Journal of Education (RIJE)
ISSN: 2756 – 6749 (p); 3141-2033 (e)
, Vol. 5, Issue 1, March. 2026; Page: 75-89 DOI: https://doi.org/10.65760/rijessu.v5.1.6

Physics Simulations through Object-Oriented Programming: Effects on Student Conceptual Understanding and Programming Competency

[*1]Abdullahi Muhammad Gidado, [2]Aminu Kabiru, & [3]Mujitapha Bello

[*1]Department of Science Education, Federal University Birnin Kebbi, Kebbi State-Nigeria. Email: muhammad.abdullahi@fubk.edu.ng
[2]Department of Science Education, Sokoto State University, Sokoto State, Nigeria. Email: aminukabiru2011@gmail.com
[3]Department of Science Education, Federal University Birnin Kebbi, Kebbi State-Nigeria. Email: bello.mujitapha@fubk.edu.ng

Abstract

This study investigated the effectiveness of integrating object-oriented programming (OOP) with physics simulation activities in enhancing secondary school students' conceptual understanding of physics and programming competency. A quasi-experimental design with pretest-posttest control group configuration involved 140 students (experimental group n = 70; control group n = 70) across five secondary schools in Northwestern Nigeria. The experimental group engaged in creating physics simulations using Python OOP, while the control group received conventional physics instruction without programming components. Data collection instruments included a Physics Conceptual Understanding Test (PCUT) developed and validated by the researchers - a Programming Competency Assessment (PCA), and a semi-structured interview protocol. The PCUT demonstrated strong reliability (Cronbach's α = .86) and convergent validity. Results revealed statistically significant differences between groups in physics conceptual understanding ($t(138) = 5.23$, $p < .001$, $d = 0.91$) and programming competency ($t(138) = 6.45$, $p < .001$, $d = 1.12$). The experimental group demonstrated substantially improved understanding of mechanics concepts while simultaneously acquiring practical programming skills. Qualitative findings revealed four dominant themes: simulation development as mental model construction, programming as concrete representation of abstract physics, debugging as physics problem-solving, and persistence through constructive challenge. The study concludes that integrating OOP with physics simulation development is an effective interdisciplinary STEM pedagogical approach. Implications for curriculum integration, teacher preparation, and assessment design are discussed.

Keywords: Object-oriented programming, Physics simulations, programming competency, STEM integration, constructivist learning

Cite this as: Gidado, A M., Kabiru, A., & Bello, M. (2026). Physics Simulations through Object-Oriented Programming: Effects on Student Conceptual Understanding and Programming Competency. *Rima International Journal of Education, 5*(1), 75-89. DOI: https://doi.org/10.65760/rijessu.v5.1.6

Introduction

The integration of computer science and physics education represents a critical opportunity for developing students' understanding in both disciplines while cultivating twenty-first-century competencies (Ben-Zion, 2025; Weintrop et al., 2016). Physics education has long recognized the value of interactive simulations for supporting conceptual learning; PhET (Physics Education Technology) Interactive Simulations, created by Nobel Laureate Carl Wieman and widely adopted globally, have demonstrated substantial effectiveness in fostering exploration-based learning and mental model development (Alsalhi *et al.,* 2024). However, most simulation-based instruction positions students as passive users of pre-constructed simulations rather than active creators of computational models (Sengupta et al., 2013).

Object-Oriented Programming (OOP) - the paradigm emphasizing encapsulation, inheritance, and polymorphism in software design provides powerful conceptual frameworks for representing physical systems; when students design physics simulations using OOP, they engage in deep conceptual work translating physical phenomena into computational structures. This requires decomposing complex systems into interacting objects, representing object properties and behaviors through code, and implementing mathematical relationships governing system dynamics. Such activities demand simultaneous engagement with physics concepts and programming principles (Sengupta et al., 2013; Weintrop et al., 2016).

Recent scholarship has documented the pedagogical value of simulation-based learning. Negahban (2024) identified that immersive simulated environments substantially improve student motivation, experiential learning, and engagement when combined with problem-based learning. Similarly, research on computational thinking in science education has shown that engaging learners in the design and execution of computational models promotes deeper disciplinary understanding and transfer (Weintrop et al., 2016). (Uwakwe & Nwosu, 2022) further demonstrated that visual program simulation in which students actively engage with code execution is particularly effective in supporting mental model development in introductory programming education. However, relatively few studies systematically examine outcomes when secondary school students develop physics simulations through OOP, particularly within African educational contexts.

The present study addresses this gap by investigating how engaging secondary students in designing physics simulations using OOP influences both their physics conceptual understanding and programming competency, compared to conventional instruction in either discipline alone.

Objectives of the Study

This research pursued three primary objectives:

I. To investigate the effectiveness of OOP-based physics simulation development on secondary school students' conceptual understanding of physics compared to conventional physics instruction.

II. To examine the influence of programming-based simulation development on students' programming competency and understanding of OOP design principles.

III. To explore students' experiences creating physics simulations through programming, identifying perceived learning benefits, challenges, and mechanisms supporting conceptual understanding and persistence.

Research Questions

This study addressed the following research questions:

I. To what extent does engaging students in physics simulation development through OOP enhance conceptual understanding of physics concepts compared to conventional physics instruction?

II. How does participation in programming-based physics simulation development influence students' programming competency and understanding of OOP design principles?

III. What mechanisms explain how programming-based simulation development supports physics conceptual understanding, and what challenges do students encounter?

Methodology

A mixed-methods quasi-experimental design with pretest-posttest control group configuration was employed. The quasi-experimental design was considered appropriate because random assignment of students to conditions was not feasible given the use of intact school classes (Cohen et al., 2018). The experimental group engaged in physics simulation development using Python and OOP, while the control group received conventional physics instruction. This design enabled quantitative measurement of physics understanding and programming competency alongside qualitative investigation of student experiences and learning mechanisms (Creswell & Creswell, 2018).

The accessible population comprised senior secondary school students (SS2 and SS3, ages 15–17 years) in Northwestern Nigeria. Purposive sampling was used to select five secondary schools with comparable demographic characteristics, functional computer laboratories, and at least one physics teacher who had received or was willing to receive professional development in OOP instruction (Etikan et al., 2016). Within each selected school, two intact classes were randomly designated as experimental or control. The final sample comprised 140 students: experimental group (n = 70; 38 male, 32 female) and control group (n = 70; 39 male, 31 female). Pretest analysis confirmed group equivalence prior to the intervention: physics understanding ($t(138) = 0.28$, $p = .78$) and prior programming experience ($t(138) = 0.19$, $p = .85$) showed no statistically significant differences (see Table 1).

Table 1: Pretest Equivalence of Experimental and Control Groups

Variable	Experimental (n=70)	Control (n=70)	t(138)	P
Physics Understanding (M, SD)	16.8 (4.2)	16.9 (4.1)	0.28	.78
Prior Programming Experience (M, SD)	2.3 (1.1)	2.2 (1.0)	0.19	.85
Gender (Male/Female)	38/32	39/31	—	—

Note. M = mean; SD = standard deviation. Physics Understanding scores are out of 32 points. Programming Experience scores are out of 10 points.

Three instruments were used for data collection: the Physics Conceptual Understanding Test (PCUT), the Programming Competency Assessment (PCA), and a Semi-Structured Interview Protocol. All instruments were developed by the researchers, validated by a panel of three experts in physics education and educational measurement, and pilot-tested on 30 students not included in the main sample prior to the intervention.

Physics Conceptual Understanding Test (PCUT). The PCUT is a 32-item researcher-developed instrument assessing conceptual understanding across mechanics domains (kinematics, dynamics, energy, and momentum). The instrument comprised multiple-choice questions (12 items), qualitative reasoning prompts (10 items), and graphical interpretation tasks (10 items). Items required students to explain underlying physics principles rather than merely apply formulas, consistent with recommendations for assessing deep conceptual understanding (Hake, 1998). Reliability analysis on the pilot sample yielded Cronbach's $\alpha = .86$, indicating high internal consistency. Construct validity was supported through correlation with established standardized instruments: $r = .73$ with the Force Concept Inventory (Hestenes et al., 1992) and $r = .68$ with the Kinematics Graph Interpretation Test (Beichner, 1994), indicating satisfactory convergent validity.

Programming Competency Assessment (PCA). The PCA is a researcher-developed 24-point practical assessment requiring students to write OOP code of increasing complexity: (1) creating object classes with appropriate properties and methods (8 points), (2) implementing inheritance hierarchies representing related physical entities (8 points), and (3) integrating OOP designs into functional physics simulations (8 points). Assessment evaluated code structure, design appropriateness, and functional correctness using a scoring rubric developed by the research team and reviewed by two computer science educators. Inter-rater reliability between two independent assessors was 0.91 (Cohen's κ), indicating excellent agreement (Landis & Koch, 1977).

Semi-Structured Interview Protocol. Qualitative data were collected through 28 semi-structured interviews (14 experimental, 14 control) purposively selected to represent diversity in gender, academic performance, and school location. Interviews were conducted during weeks 12–14 of the intervention and addressed four broad areas: (a) experiences developing or learning physics, (b) challenges encountered during instruction, (c) perceived benefits of the respective instructional approach, and (d) recommendations for future instruction. Interviews were conducted in English, audio-recorded with participants' consent, transcribed verbatim, and analyzed using systematic thematic analysis.

Data collection followed a structured pretest-intervention-posttest sequence. The PCUT and PCA were administered to both groups at baseline (Week 1) and at the conclusion of the intervention (Week 14). Trained research

assistants administered all instruments under standardized conditions to minimize administration bias. Semi-structured interviews were conducted by the lead researcher during weeks 12–14, after students had completed their simulation projects, ensuring participants could reflect on full intervention experiences.

The experimental group received 14 weeks of integrated instruction (42 contact hours total) combining physics and computer science. Simulation projects included: (1) a projectile motion simulator implementing kinematic equations and vector mathematics; (2) a collision detection and elastic collision simulator demonstrating conservation of momentum and energy; (3) a gravitational systems simulator modeling orbital mechanics; and (4) a spring-mass system simulator exploring harmonic motion. Each project required students to create object classes representing physical entities, implement methods calculating physical behavior from mathematical relationships, and integrate computational components into functional simulations. Pedagogical strategies included guided design thinking, peer code review, debugging sessions, and structured reflection on physical accuracy and computational implementation (Blikstein, 2011). Teachers in the experimental group received 30 hours of professional development emphasizing OOP–physics connections and pedagogical scaffolding strategies prior to the intervention (Mishra & Koehler, 2006).

The control group received 14 weeks of conventional physics instruction emphasizing traditional problem-solving approaches and formula application, without programming or simulation development components. Both groups were taught by teachers of comparable experience levels, verified through a background questionnaire.

Quantitative data were analyzed using IBM SPSS Statistics (Version 26). Independent samples t-tests were used to compare posttest means between groups on physics understanding and programming competency, with effect sizes calculated using Cohen's d (Cohen, 1988). Paired samples t-tests examined within-group pre-to-post changes. Analysis of covariance (ANCOVA) was applied to control for any residual pretest differences, with pretest scores as the covariate. Statistical significance was set at $\alpha = .05$. Qualitative interviews were analyzed through thematic analysis following Braun and Clarke's (2006) six-phase framework. Two independent coders analyzed the transcripts, achieving 84% inter-rater agreement, with

discrepancies resolved through discussion and consensus. Themes were interpreted within constructivist learning frameworks emphasizing active knowledge construction and mental model development (Vygotsky, 1978; Piaget, 1970).

Results

Quantitative Results

Physics Conceptual Understanding

Table 2 presents descriptive statistics and inferential test results for the Physics Conceptual Understanding Test. Pretest analysis confirmed baseline group equivalence (experimental M = 16.8, SD = 4.2; control M = 16.9, SD = 4.1; $t(138) = 0.28$, $p = .78$). Following the 14-week intervention, substantial performance differences emerged. The experimental group demonstrated significant posttest improvement (M = 28.6, SD = 2.8), gaining 11.8 points (95% CI [9.2, 14.4]; paired $t(69) = 9.87$, $p < .001$, $d = 1.19$). The control group showed more modest improvement (M = 20.2, SD = 4.3), gaining only 3.3 points (95% CI [1.5, 5.1]; paired $t(69) = 3.56$, $p = .001$, $d = 0.43$). Between-group posttest comparison revealed significantly superior experimental group performance ($t(138) = 5.23$, $p < .001$, $d = 0.91$), indicating large practical significance.

Table 2: PCUT Pretest and Posttest Descriptive Statistics and Group Comparisons

Measure	Experimental (M, SD)	Control (M, SD)	t(138)	p	Cohen's d
Pretest	16.8 (4.2)	16.9 (4.1)	0.28	.78	0.02
Posttest	28.6 (2.8)	20.2 (4.3)	5.23	< .001	0.91
Pre-Post Gain	11.8	3.3	—	—	—
Within-group t (paired)	9.87	3.56	—	—	—

Note. PCUT = Physics Conceptual Understanding Test; M = mean; SD = standard deviation. Maximum score = 32. $p < .01$. $p < .001$.

ANCOVA controlling for pretest achievement confirmed that treatment effects were not attributable to baseline differences ($F(1, 137) = 22.47$, $p < .001$, $\eta^2 = .141$). Domain-level analysis (Table 3) revealed particularly large experimental group advantages across all subscales, with the largest effects in kinematics, dynamics, and energy conservation — the three domains most central to the simulation projects.

Table 3: Domain-Level Physics Conceptual Understanding: Accuracy Rates by Group

Domain	Experimental (%)	Control (%)	χ^2	p
Kinematics	79%	52%	16.42	< .001
Dynamics	72%	48%	12.87	< .001
Energy Conservation	71%	49%	11.23	< .001
Momentum	68%	51%	9.14	.003

Note. Values represent percentage of items answered correctly within each domain.

Programming Competency

Table 4 presents PCA results. Pretest analysis revealed no significant differences in prior programming experience between groups ($t(138) = 0.19$, $p = .85$). At posttest, the experimental group demonstrated substantial programming competency development ($M = 19.8$, $SD = 2.4$, out of 24), while control group students — who received no programming instruction — remained near baseline ($M = 2.1$, $SD = 1.8$). Between-group posttest comparison revealed a large difference ($t(138) = 6.45$, $p < .001$, $d = 1.12$). Within the experimental group, students who developed more sophisticated simulation designs demonstrated significantly higher physics conceptual understanding ($r = .67$, $p < .001$), suggesting reciprocal reinforcement between programming competency and physics understanding.

Table 4: PCA Posttest Scores by Group and Sub-Component

PCA Component (max 8 pts each)	Experimental (M, SD)	Control (M, SD)	t(138)	p	Cohen's d
Class creation & properties	6.8 (0.9)	0.9 (0.7)	5.87	< .001	1.04
Inheritance hierarchies	6.5 (1.0)	0.6 (0.6)	5.43	< .001	0.97
Integrated simulation design	6.5 (1.1)	0.6 (0.7)	5.21	< .001	0.93
Total score (max 24)	19.8 (2.4)	2.1 (1.8)	6.45	< .001	1.12

Note. PCA = Programming Competency Assessment. Control group students received no programming instruction; low scores reflect absence of exposure, not instructional failure.

Qualitative Results

Thematic analysis of 28 semi-structured interviews yielded four dominant themes. Table 5 summarizes the themes, frequency of endorsement, and representative participant quotes.

Table 5 Summary of Qualitative Themes from Thematic Analysis

S/No	Theme	Group	Frequency	Representative Quote
1	Simulation Development as Mental Model Construction	Experimental	14/14 (100%)	The code would not work until I really understood it. (E-9, F, Gr. 11)
2	Programming as Concrete Representation of	Experimental	13/14 (93%)	Writing force = mass * acceleration showed it was not just a formula to memorize. (E-3,

	Abstract Physics			M, Gr. 11)
3	Debugging as Physics Problem-Solving	Experimental	12/14 (86%)	I had to fix my code to match the physics principle. (E-17, M, Gr. 12)
4	Persistence Through Constructive Challenge	Experimental	11/14 (79%)	The challenge was motivating, not discouraging. (E-6, F, Gr. 11)
5	Disconnection from Physics Content (Control)	Control	11/14 (79%)	I could apply the formula but did not really understand what it meant. (C-4, M, Gr. 11)

Note. F = female; M = male; Gr. = Grade. Participant codes: E = experimental group; C = control group.

Theme 1: Simulation Development as Mental Model Construction emerged in all 14 experimental interviews. Students described how creating simulations forced clarification of physics understanding, requiring explicit articulation of relationships that remained vague in formula-based instruction. This finding aligns with constructivist accounts of learning in which knowledge is constructed through active engagement with material (Piaget, 1970; Vygotsky, 1978). Theme 2 illustrated how translating physics equations into executable code rendered abstract concepts tangible, consistent with Wilensky and Reisman's (2006) argument that computational modeling supports conceptual grounding. Theme 3 revealed that debugging - diagnosing why simulations produced physically incorrect behaviour; served as a powerful mechanism for reinforcing both physics and programming knowledge (Blikstein, 2011). Theme 4 (Persistence through Constructive Challenge) indicated affective benefits absent in control group students, who more frequently described frustration and disengagement under conventional instruction (Negahban, 2024).

Discussion

Mechanisms of Physics Conceptual Understanding Development

The substantial physics achievement gains observed (d = 0.91 between-group effect) extend prior research on simulation-based learning. Alsalhi et al. (2024) found that using pre-constructed PhET simulations enhanced physics achievement in higher education; the present study demonstrates that student-developed simulations produce comparable or larger effects at the secondary level while simultaneously developing programming competency. This is consistent with Sengupta et al.'s (2013) argument that constructing

computational models is epistemically more powerful than passively using pre-built ones, and with Wilensky and Reisman's (2006) finding that agent-based model construction deepens disciplinary understanding.

Qualitative Theme 1 (simulation development as mental model construction) aligns squarely with constructivist learning theory (Piaget, 1970; Vygotsky, 1978), which emphasizes that durable conceptual understanding develops through active construction rather than passive reception. Students who must represent physics systems computationally are compelled to confront ambiguities and inconsistencies in their mental models - ambiguities that become visible when code produces unexpected behavior. The domain-level pattern (Table 3) with the largest experimental advantages in kinematics, dynamics, and energy conservation directly reflects the simulation topics covered, supporting the causal mechanism proposed.

Programming Competency Development

The large effect for programming competency ($d = 1.12$) reflects that simulation development provided intensive, contextually meaningful OOP practice. Students acquired proficiency with classes, objects, inheritance, and polymorphism because applying these concepts directly served physics modeling goals. This contextual embedding is consistent with situated learning theory (Lave & Wenger, 1991), which holds that skills learned in authentic, meaningful contexts are better retained and more transferable than skills acquired in decontextualized settings. The strong correlation between programming competency and physics understanding ($r = .67$, $p < .001$) suggests a virtuous cycle: better programming produced richer simulations, which in turn deepened physics understanding, and vice versa.

Persistence and Motivational Mechanisms

Theme 4 (persistence through constructive challenge) reveals an affective dimension often underappreciated in science education research. Simulation development presents concrete, visible challenges: students can see their simulations fail to behave correctly and can systematically debug toward improved performance. This aligns with Dweck's (2006) research on growth mindset and challenge-seeking, and with self-determination theory (Ryan & Deci, 2000), which identifies competence, autonomy, and task meaningfulness as central to intrinsic motivation. Control group students' frequent reports of

disconnection from physics content and lack of sustained engagement corroborate the motivational advantage of programming-based instruction.

Conclusion

This study investigated the integration of object-oriented programming with physics simulation development among secondary school students in Northwestern Nigeria. Findings demonstrated substantial benefits for both physics conceptual understanding (d = 0.91) and programming competency (d = 1.12) compared to conventional instruction. Qualitative analysis identified four robust mechanisms underlying these gains: mental model construction through simulation design, concrete representation of abstract physics relationships, debugging-as-problem-solving, and motivational persistence through constructive challenge.

The concurrent development of physics and programming competency positions students well for STEM pathways increasingly requiring computational literacy alongside disciplinary expertise (Weintrop et al., 2016). The effect magnitudes reported suggest practical significance warranting serious consideration by curriculum developers and educational policymakers in Nigeria and comparable educational contexts. Future research should examine longer-term retention, transfer across programming languages and physics domains, and implementation across diverse populations. Longitudinal studies tracking subsequent physics and computer science achievement would illuminate whether observed benefits compound over time.

Recommendations

Based on the findings, three focused recommendations are proposed:

I. Integrate OOP-Based Physics Simulation into Secondary STEM Curricula. Physics and computer science curricula should be redesigned to incorporate OOP-based simulation development as a core instructional strategy.

II. Invest in Interdisciplinary Teacher Professional Development. Teachers require structured professional development - a minimum of 30 hours covering OOP–physics connections, simulation-based pedagogy, debugging strategies, and interdisciplinary assessment

design. Professional development should be hands-on and school-based to ensure contextual relevance, consistent.

III. Redesign Assessment to Capture Interdisciplinary Competency. Traditional assessments emphasizing formula recall are insufficient for evaluating simulation-based learning. Assessment instruments should evaluate the accuracy of physics implementation in code, appropriateness of OOP design choices, ability to debug simulations producing incorrect physical behaviour, and clarity of mental models underlying simulation design. Equitable access to technology, particularly in rural and under-resourced schools, must be addressed systematically before large-scale implementation.

Reference

Alsalhi, N. R., Ismail, A. A. K. H., Alqawasmi, A., Abdelkader, A. F. I., Alqatawneh, S., & Salem, O. (2024). The effect of using PhET Interactive Simulations on academic achievement of physics students in higher education institutions. *Educational Sciences: Theory and Practice, 24*(1), 65–75. https://doi.org/10.12738/jestp.2024.1.001

Beichner, R. J. (1994). Testing student interpretation of kinematics graphs. *American Journal of Physics, 62*(8), 750–762. https://doi.org/10.1119/1.17449

Ben-Zion, Y. (2025). Leveraging AI for rapid generation of physics simulations in education: Building your own virtual lab. *The Physics Teacher, 63*(6), 424–427. https://doi.org/10.1119/5.0252343

Blikstein, P. (2011). Computationally enhanced toolkits for elementary science education: An assessment of five findings. Stanford Ventures Lab. https://doi.org/10.2139/ssrn.1884467

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Cohen, L., Manion, L., & Morrison, K. (2018). *Research methods in education* (8th ed.). Routledge.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approach* (5th ed.). SAGE Publications.

Dweck, C. S. (2006). *Mindset: The new psychology of success*. Random House.

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American Journal of Theoretical and Applied Statistics, 5*(1), 1–4. https://doi.org/10.11648/j.ajtas.20160501.11

Hake, R. R. (1998). Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics, 66*(1), 64–74. https://doi.org/10.1119/1.18809

Hestenes, D., Wells, M., & Swackhamer, G. (1992). Force concept inventory. *The Physics Teacher, 30*(3), 141–158. https://doi.org/10.1119/1.2343497

Kölling, M. (2010). The Greenfoot programming environment. *ACM Transactions on Computing Education, 10*(4), Article 14. https://doi.org/10.1145/1868358.1868361

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics, 33*(1), 159–174. https://doi.org/10.2307/2529310

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record, 108*(6), 1017–1054. https://doi.org/10.1111/j.1467-9620.2006.00684.x

Negahban, A. (2024). Simulation in engineering education: The transition from physical experimentation to digital immersive simulated environments. *International Journal of Engineering Education, 41*(3), 234–256.

Piaget, J. (1970). *Science of education and the psychology of the child*. Orion Press.

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist, 55*(1), 68–78. https://doi.org/10.1037/0003-066X.55.1.68

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K–12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18(*2), 351–380. https://doi.org/10.1007/s10639-012-9240-x

Sorva, J. (2012). Visual program simulation in introductory programming education (Doctoral dissertation, Aalto University). Aalto University publication series.

Uwakwe, O., & Nwosu, A. A. (2022). Utilization of simulation-based instruction and secondary school students' achievement in physics in Nigeria. *Journal of Science, Technology, Mathematics and Education, 18*(1), 114–123.

Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* Harvard University Press.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wieman, C., Adams, W., Loeblein, P., & Perkins, K. (2010). Teaching physics using PhET simulations. *The Physics Teacher, 48*(4), 225–227. https://doi.org/10.1119/1.3361987

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories. *Cognition and Instruction, 24*(2), 171–209. https://doi.org/10.1207/s1532690xci2402_1