

An Interactive Bot Command Simulation Game for Learning Programming

Wilson Mamani-Casilla *, Edilson Mamani-Yucra, Roxana Flores-Quispe, and Yuber Velazco-Paredes

Escuela Profesional de Ciencia de la Computación, Universidad Nacional de San Agustín de Arequipa, Arequipa, Peru

Email: wmamani@unsa.edu.pe (W.M.-C.); emamaniyu@unsa.edu.pe (E.M.-Y.); rfloresqu@unsa.edu.pe (R.F.-Q.);

yvelazco@unsa.edu.pe (Y.V.-P.)

*Corresponding author

Manuscript received February 13, 2026; revised March 23, 2026; accepted June 12, 2026; published October 16, 2026

Abstract—In recent years, many students have faced difficulties learning programming due to the complexity of their chosen language syntax or a lack of well-developed programming logic, posing challenges for the implementation of interactive computational tools to support this learning process. Therefore, this study proposes the development of an interactive Bot command simulation game designed to teach fundamental computational thinking concepts, such as modularity and control structures, using visual tiles with real-time feedback. This tool aims to help students solve challenges of varying difficulty by developing algorithmic strategies in a gamified and engaging environment. To evaluate its effectiveness, a quasi-experimental study was conducted with 44 high school students from two classes. A pre-test was administered to measure their initial knowledge level, followed by a post-test after a hands-on workshop in which students used the proposed game to solve increasingly complex challenges in four programming domains: Sequential logic, functions, loops, and conditionals. The proposed game had a positive effect on the development of computational thinking skills, as demonstrated by the statistical analysis, which shows significant improvements in the experimental group across all components, with average scores increasing from 7.64 to 12.41. These findings confirm that interactive, game-based simulations can significantly improve programming instruction and foster computational thinking more effectively than conventional approaches.

Keywords—computer game, programming learning, game-based learning, computational thinking

I. INTRODUCTION

Teaching students the fundamentals of programming can be a difficult task. Several factors can compound this difficulty, including age, culture, socioeconomic background, capacity, development environment, and motivation. In addition, computer programming as a course requires cognitive and metacognitive abilities. This means that students must understand the syntax and semantics of a chosen programming language and use their creativity combined with logical thinking to solve problems [1]. Therefore, maintaining these students' interest in the syntax and semantics of programming languages is considered a tedious and challenging task [2].

Furthermore, some programming languages require significant effort to understand their syntax, which can be a restriction; this can be overcome by focusing on sequences, selection, and iteration in everyday life, thus reinforcing the concepts of programming and their relationship with problem solving [3].

On the other hand, in recent years, game-based learning environments have been shown to effectively enhance student engagement and learning outcomes [4]; likewise,

video games have also demonstrated great potential to help teachers and engage students in a new and challenging way [5]. However, significant challenges remain, as successful educational games must balance education, gameplay, and storytelling and most games alone provide an intuitive but fragmented understanding [6].

In addition, some studies show that, when using the same syllabus with computer games as teaching and learning tools, students can pick up skills and knowledge much more efficiently than with the traditional teaching approach [5]. Nevertheless, the adoption of these tools involves significant challenges, and the planning and development of educational games can be costly. Most existing productivity tools struggle to find a balance between offering powerful features and maintaining an accessible learning curve for novices [7]. Moreover, most have focused on superficial gamification (e.g., quizzes, badges, or point systems) leaving a gap for fundamental programming practices such as real-time coding or iterative problem-solving [4]. Consequently, educators face technical barriers that limit their ability to foster computational thinking effectively. Furthermore, most gameplays alone have an intuitive but fragmented understanding, requiring instructor explanation for conceptual structuring [8]. For that reason, this article proposes an interactive tool to develop computational thinking using a configurable and customized programming board to resolve challenges applying strategies to help students assimilate fundamental concepts in programming logic, control structures, and algorithmic thinking; which encompass two (modularity and control structures) of the five fundamental aspects of computational thinking [9]. Where the instructor's explanation for conceptual structuring is a complementary task; since students can check in real time whether their proposal meets the challenge or not.

To achieve this objective, the following tasks were established: (1) Design and implement an environment for building increasingly complex, customized challenges, testing users' programming skills.; (2) Design and implement an interactive board with modularity options and control structures to build solutions to the proposed challenges by applying computational thinking. Its execution will allow real-time verification of the proposed solution., (3) Evaluate the effectiveness of the proposed interactive tool in teaching programming without depending on the language's syntax.

The quasi-experimental study involved 44 high-school students enrolled in a vocational training course specializing in Informatics. The results were collected through pre-post tests, which reported both global and domain-specific learning gains.

This study also revealed a compensatory effect in which lower-performing students achieved greater absolute gains in their learning; because the game was structured in a three-dimensional environment, which led to improvements in their spatial intelligence, allowing students to direct all their cognitive resources towards logical abstraction and algorithmic planning.

II. LITERATURE REVIEW

Significant research has been conducted in recent years to enhance the learning of programming through digital tools and game-based learning. This section reviews the evolution of game design frameworks, the effectiveness of specific educational games, and recent systematic analyses of their impact.

A. Evolution of Design Frameworks and Strategies

Research on the design of educational games has evolved from fundamental principles to collaborative and integrative approaches. Early work by Sharda [6] established key areas for Digital Educational Games (DEGs), emphasizing the balance between educational content, game design, and narrative design. Sharda introduced concepts such as Motion-Oriented Design (MOD) and adapted narrative principles to cater to different learning styles. Years later, amidst the challenges of the COVID-19 pandemic, Agbo *et al.* [10] shifted the focus to co-design strategies. They worked with computer science students in Nigeria using Online Co-Design (OCD) to create minigames, demonstrating that collaborative prototyping helps students connect contextual stories with computational thinking concepts. More recently, Wu *et al.* [11] analyzed the integration of computational thinking in education, finding an evolution in tools based on complexity: visual programming tools predominate in early stages, while text-based tools and robotics emerge at higher levels for complex tasks.

For this reason, our approach emphasizes the creation of a visual programming tool through a board of tiles that can be used seamlessly by high school students.

B. Effectiveness of Specific Educational Games and Simulators

Regarding the application of specific tools, early studies

like that of Zorn *et al.* [12] investigated the potential of “Minecraft” to promote interest in programming using block-based languages to control virtual robots. Following this line, Theodoropoulos *et al.* [13] confirmed that video games serve as effective and motivating environments for learning basic concepts. Subsequent research focused on specific platforms, Hooshyar *et al.* [14] evaluated its effects on secondary school students, finding positive changes in attitudes. Later, Zhang *et al.* [15], extended this investigation to primary school students in informal home environments, also observing significant improvements in computational thinking. Methodological approaches have also diversified. Soboleva *et al.* [16] proposed using games with mathematical content to develop cognitive activity and problem-solving processes. Finally, in a recent analysis of modern simulators, Kovtaniuk *et al.* [17] examined tools such as Blockly Games, Rabbids Coding, and 7 billion humans. Their results indicated that these diverse simulators significantly improve student engagement and algorithmic thinking skills.

According to these latest studies, our proposal has incorporated the principles that determine the essence of computational thinking: decomposition; abstraction; pattern recognition; creation of an algorithm applying principles of structured programming [16] through options to choose on the instruction panel, which allowed the development of the cognitive activity of the students.

C. Trend and Long-term Impact

To understand the broader trends and long-term effects, several systematic reviews have been conducted. Videnovik *et al.* [18] reviewed 113 articles, concluding that while games increase motivation, there is no single standardized methodology for creating educational games for computer science. Providing quantitative evidence, Ma *et al.* [19] conducted a meta-analysis of 24 studies, finding that collaborative activities and prior experience increase the effectiveness of game-based learning and Che *et al.* [20] have emphasized the expanding adoption of game-based learning across educational contexts and the growing need for adaptive, learner-centered, and multimodal designs to address diverse learner needs and cognitive levels.

Table 1. Comparison of educational games

Educational game	Main programming concepts	Interaction type	Feedback	Tracking
Scratch	Sequence, loops, conditions	Visual block programming	Immediate and visual. Feedback is behavioral.	Display variables on screen. A complete block of instructions is activated during the execution.
Blockly games	Logic, conditions, functions	Puzzle-based	Immediate and visual. Showing whether the puzzle is solved	Progress tracked by levels.
CodeCombat	Syntax, debugging, algorithms	Role-playing coding	Immediate during execution. Character move, attack, etc. Syntax or logic errors display messages or stop the action at the relevant point.	Pause, review code, re-run.
Code.org challenges	Fundamental constructs	Guided tasks	Immediate; error highlights	Detailed progress monitoring, time spent
Teacher-designed gamified tasks	Problem solving	Structured activities	Automatic: Unit tests	Execution logs and version history,
Interactive Bot Command Simulation (Proposed method)	Sequence, loops, conditions, functions	Isometric command simulation in a 3D environment.	Immediate and visual, during the execution. The Bot moves, executing the instructions. Logic errors stop the action at the relevant point.	Each instruction slot is activated during its execution, allowing monitoring of the execution of each instruction.

Most recently, several studies have focused on specific outcomes. Callins *et al.* [21] performed a systematic mapping of 49 games, confirming they increase assimilation of programming content. Lathifah *et al.* [22] demonstrated that participation in game-based learning environments enhances student motivation, confidence, and engagement in programming activities, contributing to improved learning outcomes and a more positive disposition toward Science, Technology, Engineering, and Mathematics (STEM)-related learning. Finally, Zourmpakis *et al.* [23] investigated the long-term impacts in a quasi-experimental study, confirming sustained improvement in computational thinking competencies.

As can be seen, a considerable number of games have been implemented to increase student learning motivation. Likewise, there is an increasing need for adaptive, student-centered designs, which is why our game adopted a 3D isometric design instead of a simple board to solve the proposed challenges. Finally, the main advantage of our proposed method is shown in Table 1.

III. MATERIALS AND METHODS

A. Research Objectives

The main objective of this study is to design, implement, and evaluate an interactive tool based on a Bot command simulation game for learning programming, with the aim of developing computational thinking skills in students. Specifically, the study seeks to determine whether the proposed tool improves the understanding of fundamental programming concepts such as sequences, loops, functions, and conditionals through an interactive environment with personalized challenges that encourage the creation of diverse strategies using programming logic to solve them in real time.

B. Interactive Bot Command Simulation Game Design

The architecture of this proposed interactive Bot command simulation game was designed to address the gap in programming logic acquisition through a six-stage pipeline.

1) System architecture overview

As illustrated in Fig. 1, the architecture is composed of six core modules:

- **Input System:** The student defines the game level and also builds the scenario by placing tiles, obstacles, and establishing the objective(s) to achieve.
- **Programming System:** Manages the logic built by the student using.
- **Execution Engine:** Interprets and executes the sequence of instructions defined by the student.
- **Movement System:** Calculates positions and detects collisions on the game board.
- **Game Logic:** Validates the win/lose conditions and level progression.
- **Rendering System:** Projects the 2D logic into a 3D isometric visualization of the Bot command simulation game.

The interaction flow begins at the Input System, where students enter commands into the active programming grid, which includes three zones: Main, Function, and Loop. Subsequently, the Execution Engine reads the resulting

instruction matrix and translates it into sequential movement commands for the Bot.

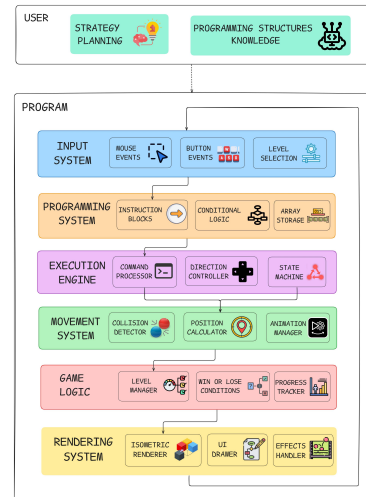


Fig. 1. Architecture of the proposed interactive simulation game.

2) Programming modules and data structures

To foster modular thinking, the system utilizes specific arrays to store instructions, defined in Eq. (1). This structure allows the system to differentiate between the main execution flow (Main), subroutines (Func), and loops (Loop) respectively.

$$\begin{aligned} Main &= [i_1, \dots, i_{12}], \\ Func &= [j_1, \dots, j_8], \\ Loop &= [k_1, \dots, k_4] \end{aligned} \quad (1)$$

The proposed tool supports two key programming aspects of basic Computational Thinking—modularity and control structures:

- **Control structures:** Three control structures (sequence, iteration, and selection) are sufficient for any program [9]. This module contains:
- **Sequential Module:** This module included basic directional movement.
- **Iterative Module:** Logic based on cycles with configurable counters.
- **Conditional Module:** Logic based on Traffic Light states to control program flow under contextual conditions.
- **Modularity:** Reducing complex systems to structures of many simple components [9]; it contains:
- **Functional Module:** Code reuse through subroutines.

3) Design of the isometric environment

a) Pedagogical foundations of isometric navigation

The design of the proposed interactive tool transcends mere visual aesthetics; it is based on established cognitive principles that foster the spatial reasoning essential for computational thinking. The main mechanics of the game consist of navigating with a Bot through a strategy development environment to find the routes that allow reaching the objective points defined on the game board, which requires users to plan, evaluate and propose algorithmic solutions through a sequence of instructions, processes analogous to algorithmic thinking and the structuring of programming logic [24].

To optimize this cognitive process, a fixed isometric (dimetric 2:1) projection was selected. This perspective provides a stable and structured spatial reference frame, significantly reducing the extraneous cognitive load associated with manipulating a 3D camera or reconciling dynamic perspective shifts [25]. This stability allows students to direct their full cognitive resources toward logical abstraction and algorithmic planning. Crucially, this stable isometric representation actively trains mental rotation and spatial visualization abilities. Empirical evidence indicates that sustained interaction with structured three-dimensional environments leads to measurable improvements in spatial intelligence [26], a core cognitive domain linked to performance in navigation and complex problem-solving [27, 28].

Furthermore, the choice of an isometric maze is supported by research on spatial knowledge acquisition. As noted by Park *et al.* [25], rotated or oblique three-dimensional representations (such as an isometric view) enable users to develop a more robust understanding of environmental layout (survey knowledge), overcoming the limitations of flat 2D maps. This is essential for building the integrated mental models required for structured problem-solving.

b) Spatial navigation as a catalyst for algorithmic thinking

The maze-like structure of the levels is intentionally designed to develop complementary types of spatial knowledge critical for structured thinking. The user must acquire both route knowledge (the step-by-step sequence of actions to reach the goal) and survey knowledge (an integrated mental map of the environment's layout and the relationships between obstacles and paths). Virtual navigation research confirms that such restrictive, guided environments are effective methodologies for developing these spatial skills [29].

This design directly simulates the cognitive demands of navigating real-world complex infrastructures with limited access or partial visibility, thereby serving as a valid and safe tool for evaluating and training spatial learning. Frequent interaction with navigational video games has been associated with enhanced spatial awareness and greater adaptability to new spatial challenges, as players learn to process spatial relationships within complex environments [24]. These enhanced spatial strategies are cognitively transferable, supporting the higher-order processes of decomposition, pattern recognition, and algorithmic planning inherent to programming [28].

Thus, the act of finding and programming the optimal path within this isometric maze does not merely present a puzzle; it systematically exercises the executive functions, visuospatial processing, and sequential planning that form the foundation of computational thinking.

c) Technical implementation of the isometric environment

The Execution Engine interprets the created instruction arrays using a deterministic, state-driven loop. For each movement command, the engine calculates the target position within an internal logical grid.

To ensure a precise and coherent spatial representation, the game world is internally modeled as a two-dimensional matrix, where the value of each cell categorizes the terrain: -1 for goals, 0 for walkable paths, and any positive integer >0

for obstacles on the game board. This logical abstraction is completely decoupled from the visual representation, allowing all game logic (movement validation, collision detection and win-condition checks) to operate on a simple and deterministic grid.

The transformation from this logical grid to the on-screen isometric view is governed by a standard dimetric projection with a 2:1 pixel ratio. This ratio was selected to maintain correct depth perception and eliminate graphical artifacts, such as pixel shimmering during diagonal movements. The mathematical basis that defines the projected isometric tile width (W_{iso}) for a grid cell of side L is shown in Eq. (2).

$$W_{iso} = \sqrt{2}L, H_{iso} = \frac{W_{iso}}{2} \quad (2)$$

Rather than detailing the extensive linear algebra (rotation, scaling, and translation) required for the bidirectional mapping between logical and screen coordinates, the critical educational feature of this architecture is its seamless background execution. The Rendering System transforms the previous model, projecting the abstract 2D logic grid into a coherent and visually intuitive 3D-like isometric space without exposing the user to the underlying coordinate math.

This transformation allows the Movement System to perform all position calculations and collision detections based on the efficient logical grid, while the rendering layer provides a stable, engaging, and spatially clear navigation scene. All visual assets—the Bot, obstacles, goal tiles, and interactive elements—are designed and rendered to align perfectly with this fixed isometric perspective. This means that any changes in the 3D environment are updated in real time from the 2D board.

The decoupling of the logical grid and isometric view is a critical design feature. It ensures that the cognitive load on the user is directed toward algorithmic planning and spatial reasoning, rather than being burdened by graphical complexities or perspective manipulation. The stable, structured reference frame provided by the fixed isometric projection reduces extraneous cognitive load, allowing students to focus entirely on the core tasks of decomposition, pattern recognition, and sequence planning [23]. This seamless integration of a simple logical backend with a cognitively optimized visual frontend creates a user experience that prioritizes and directly supports the development of computational thinking.

4) Educational design and level progression

To evaluate the educational potential, a representative set of four progressive level categories was designed, visual examples of these are presented in Fig. 2. Each category focuses on specific programming constructs, gradually increasing in complexity:

- Basic Navigation (Sequential Level): Focuses on sequential execution and spatial orientation using only the Main block, limited to 12 instructions.
- Modular Thinking (Modular Level): Introduces instruction limits that encourage the use of the Function block to encapsulate repeated sequences and promote code reuse.
- Iterative Logic (Iterative Level): The Loop block

implements a fixed-count repetition (repeat N times), where $N \in [0, 5]$. Allowing zero iterations enables students to reason about boundary cases and algorithm correctness without executing actions.

- **Conditional Logic (Conditional Level):** Incorporates a Traffic Light slot to simulate if-else structures, fostering adaptive problem-solving.

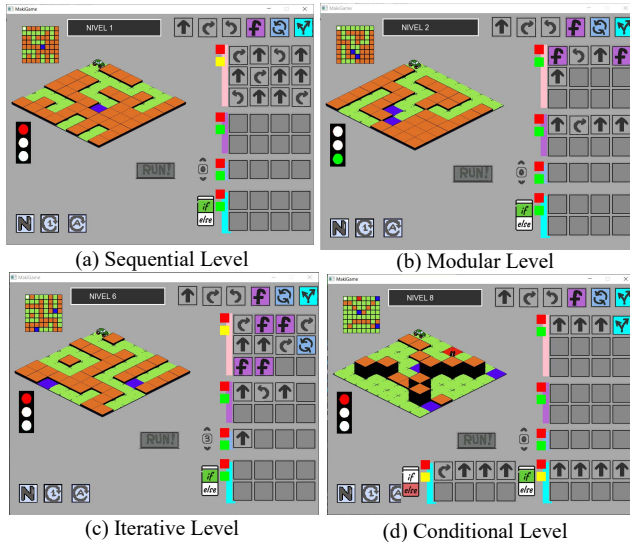


Fig. 2. Progression of game levels: (a) basic navigation focusing on sequence, (b) modular logic focusing on reusability, (c) loop integration for recurring patterns, and (d) conditional logic application.

5) Validation of design efficiency

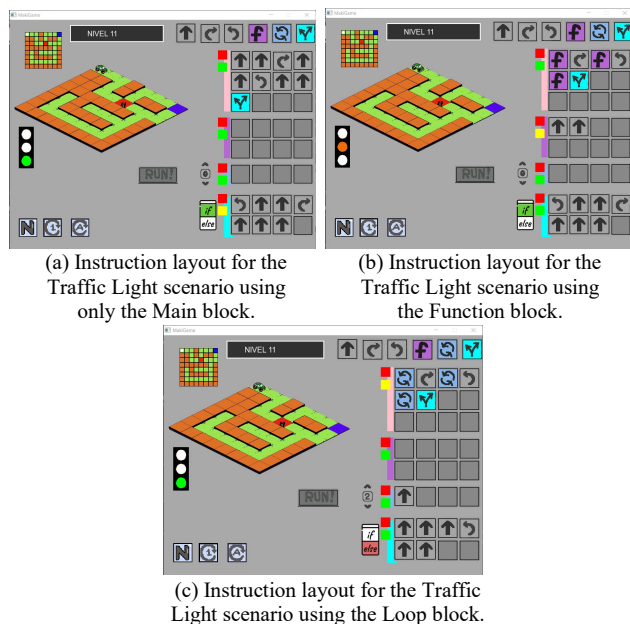


Fig. 3. Visual comparison of programming strategies: (a) redundant sequential code, (b) modularized function code, and (c) optimized loop-based code.

To validate the efficiency of the proposed game levels (Main, Functions, Loops, Conditionals), a comparative analysis of instruction usage was conducted in a Traffic Light scenario example. As visualized in Fig. 3 and detailed in Table 2, the loop-based strategy reduced the total instructions to 20, compared to 22 in the sequential approach, confirming the educational value of advanced programming structures. In this case, the three examples execute the same actions but with different strategies, with the third representing a better

solution since it employs more advanced programming structures.

Table 2. Comparison of instruction usage in the Traffic Light scenario

Test Configuration	Main	Function	Loop	Green State (if)	Red state (else)	Total
Using only Main	9	0	0	7	6	22
Using Function	6	2	0	7	6	21
Using Loop	4	0	3	7	6	20

C. Methodology and Experimental Design

1) Research design

This study adopts a quasi-experimental design with pre-test and post-test measures to evaluate the impact of the proposed tool. The intervention was structured as a workshop following a defined schedule Table 3, allowing for the systematic collection of performance data.

Table 3. Experimental implementation schedule

Stage	Section 1	Section 2	Activities
Pre-test	Day 1 (45 min)	Day 1 (45 min)	Initial Assessment
Intervention	2 sessions	2 sessions	Guided use of Interactive Tool
Post-test	Day 2 (45 min)	Day 2 (45 min)	Final Assessment

To ensure a structured and replicable learning process, the experimental procedure was conducted over two days, spaced exactly one week apart, following a three-phase protocol:

- **Stage 1: Pre-test (Day 1):** The research team introduced the study's objective testing a new tool designed to improve programming logic. Students had 45 minutes to complete the pre-test. Start and completion times for each student were recorded to establish a baseline not only for accuracy but also for cognitive processing speed.
- **Stage 2: Intervention (Day 1 & Day 2):** Session 1 (Day 1, 90 min): This session began with a brief theoretical introduction followed by a practical demonstration of the game's interface. The session focused strictly on Sequential Logic and Functions. Pedagogically, the tool was used to establish the concept of an "instruction" as the fundamental unit of programming, utilizing the game's visual analogy where one code instruction translates directly to one spatial tile movement by the Bot. Functions were introduced as reusable "mini-programs". Students solved pre-designed exercises with escalating difficulty. While some students progressed rapidly, others received guided hints from the facilitators to help them reach the solution, ensuring comprehension over mere completion. Session 2 (Day 2, 90 min): Conducted one week later, this session started with a recap exercise of the previous topics. The core focus then shifted to Loops and Conditional Logic. The tool was continuously used to visually represent how execution flow changes based on conditions (e.g., Traffic Light states) and repetition.
- **Stage 3: Evaluation Phase (Day 2):** Immediately following Session 2, students completed the 45-minute post-test under the same controlled conditions as Phase 1, again with individual completion times recorded.

2) Participants

User testing involved 44 fifth-year high school students from the Emblematic School "Honorio Delgado Espinoza" in Arequipa, Perú. The sample was selected using purposive

sampling, focusing on students enrolled in the regular secondary education curriculum. The distribution is detailed in Table 4.

Table 4. Distribution of students

Section	Number of students	Grade level
Section 1	14 students	5th Year of High school
Section 2	30 students	5th Year of High school
Total	44 students	

3) Data collection instruments

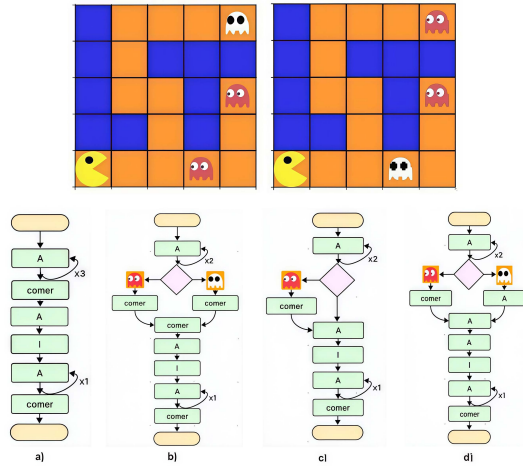


Fig. 4. Test item used to assess conditional logic.

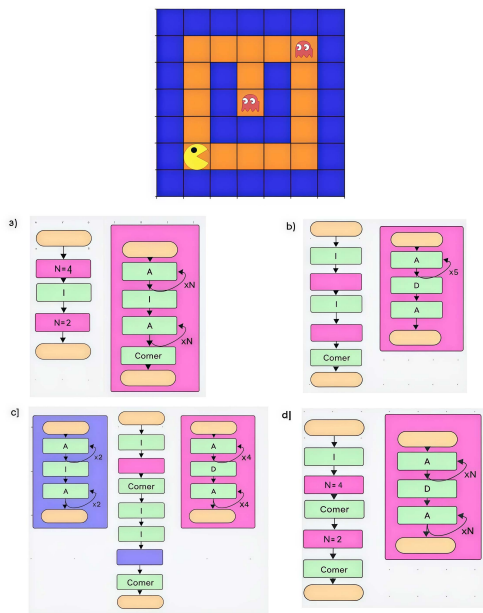


Fig. 5. Test item used to assess the understanding of functions.

Data were collected using two primary instruments: a Pre-test and a Post-test. Both tests were based on the validated computational thinking exam framework proposed by Román-González [30], but were specifically adapted to align with the visual and spatial nature of our proposed tool. The instrument consisted of 14 multiple-choice items, with a maximum possible score of 14 points (one point per correct item). To comprehensively evaluate the acquired skills, the items were distributed across the four core domains targeted during the intervention: 4 items for sequential logic, 4 items for functions, 3 items for loops, and 3 items for conditional logic. For instance, to assess conditional logic. See Fig. 4, one of the items presented the following scenario: “The objective is to navigate towards the red ghosts and consume them while

strictly avoiding a white ghost. Because the white ghost can spontaneously change its position at any given time, the player must evade it and continue moving forward.” Similarly, to evaluate the understanding of functions, another item challenged students with the task: “The objective is to navigate the scenario and successfully consume all the ghosts present”. This specific scenario required students to identify repeating action patterns and encapsulate them within defined functions to optimize their solution. See Fig. 5. To ensure content validity of our specific adaptation, the modified instrument was reviewed by university professors. This review guaranteed that the terminology, visual aids, and difficulty level were aligned with the cognitive capacity of fifth-year high school students.

D. Data Analysis

Descriptive statistics (mean, standard deviation, minimum, maximum, and standard error of the mean) were computed to summarize student performance. Data normality was assessed using the Shapiro–Wilk Test [31].

The paired samples T-test was used as the primary inferential analysis, given its robustness to deviations from normality for sample sizes greater than 30. To confirm the robustness of the results, the Wilcoxon signed-rank test was also conducted as a non-parametric alternative.

These analyses were conducted to examine the pre–post improvements in students’ computational thinking skills.

Effect sizes were calculated using Cohen’s d for paired designs, computed as the mean of the paired differences divided by the standard deviation of the differences [32]. Statistical significance was set at $\alpha = 0.05$. All analyses were conducted using the Scipy library in Python.

IV. RESULT

A. Statistical Analysis and Global Results

The impact of the interactive tool was evaluated using a pre-test/post-test quasi-experimental design involving the complete sample of 44 students from 2 sessions, as shown in Table 3. Fig. 6 shows that, overall, the students experienced a substantial improvement in their performance after the post-test.

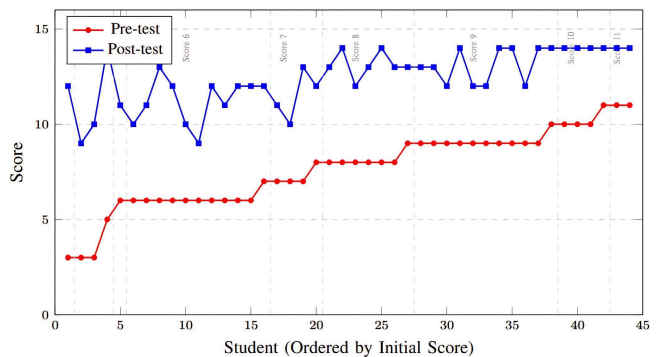


Fig. 6. Individual score progression ordered by initial pre-test score. Each student pair (pre/post) maintains correspondence after sorting.

Initially, the normality of the overall data distribution was assessed using the Shapiro–Wilk test [31], which indicated a non-normal distribution ($p < 0.05$), attributed primarily to the ceiling effect observed in the post-test. The normality of the

individual difference scores (post-test minus pre-test) was also verified, confirming a deviation from the normal distribution (Shapiro-Wilk $p = 0.000373$).

Despite these deviations, the paired samples Student's t-test was applied, supported by its robustness to non-normality [33], with sample sizes greater than 30, and by the Central Limit Theorem [34]. Additionally, the Wilcoxon signed-rank test was performed to ensure robustness. Both parametric and non-parametric tests yielded statistically significant results. The paired-samples t-test showed a significant difference between pre- and post-intervention scores, $t(43) = -21.44$, $p < 0.001$, a result confirmed by the Wilcoxon signed-rank test, $W = 0$ (all ranks positive). The consistency of these findings supports the conclusion that the intervention produced a significant improvement in learning outcomes.

Table 5 shows that the overall mean score increased significantly from 7.64 to 12.41, representing an absolute gain of 4.77 points on a 14-point scale with a standard deviation that decreased from 2.05 to 1.48, indicating a homogenization of learning in students' computational thinking skills for the entire classroom, and not just for a specific subset of students. See Fig. 7.

Table 5. Statistical Summary ($n = 44$)

Metric	Pre-test	Post-test
Mean	7.64	12.41
Standard Deviation (SD)	2.05	1.48
Minimum	3	9
Maximum	11	14
Median	8	13
Q1	6	12
Q3	9.25	14
IQR	3.25	2
Standard Error Mean	0.31	0.22

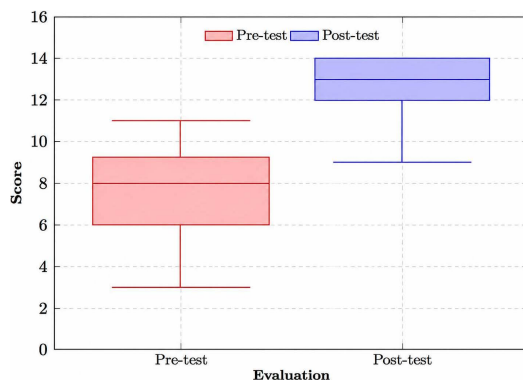


Fig. 7. Distribution of pre-test and post-test scores for the entire sample ($n = 44$).

In addition, there was an increase in the median score from 8 to 13 points, representing a 62.5% improvement. Furthermore, the Interquartile Range (IQR) between pre-test and post-tests decreased from 3.25 points to 2 points, showing a 38.5% reduction in data dispersion, also demonstrating strong homogenization effect, that is, students at all initial performance levels converged toward greater achievement. The lower limit also increased from 3 to 9 points, confirming that even the lower-performing students reached a competent level. These findings provide compelling evidence that the proposed interactive Bot command simulation game not only improves overall performance but also reduces the learning gap, fostering more equitable skill development.

In addition, the effect size was determined using Cohen's categorization [34], reaching $d = 3.23$, which was exceptionally high according to conventional benchmarks [32]. These results corroborate that the interactive simulation game had a substantial educational impact, as all participants showed marked improvement.

The complete paired samples t-test results, including the mean difference, test statistic, degrees of freedom, and effect size, are summarized in Table 6.

Table 6. Paired t-test results ($n = 44$)

Comparison	Mean Diff.	t	df	Sig. (2-tailed)	Cohen's d
Pre—Post	-4.77	-21.44	43	< 0.001	3.23

B. Skill Acquisition by Programming Domain

Beyond the global scores, it is essential to analyze how the tool impacted specific dimensions of computational thinking. As illustrated in Fig. 8 and detailed in Table 7, growth was observed across all four evaluated domains: Sequential Logic, Functions, Loops, and Conditionals. However, the nature of this growth varied meaningfully:

- Sequential Logic: Showed a moderate improvement (28%), reaching near-mastery levels. This suggests that the visual interface makes linear logic intuitive almost immediately.
- Functions: This concept showed remarkable growth (67%). This finding proves that the tool effectively scaffolds modular thinking, allowing students to successfully encapsulate code—a concept often difficult to grasp in syntax-based languages.
- Loops and Conditionals: Initially, these were the most challenging areas. However, they registered the highest relative gains (113% and 114%). Although these remain the most complex concepts post-intervention, the doubling of scores indicates that the isometric visual feedback was crucial. By visualizing the “looping” action of the Bot, students were able to internalize these abstract control structures more effectively than through abstract code alone.

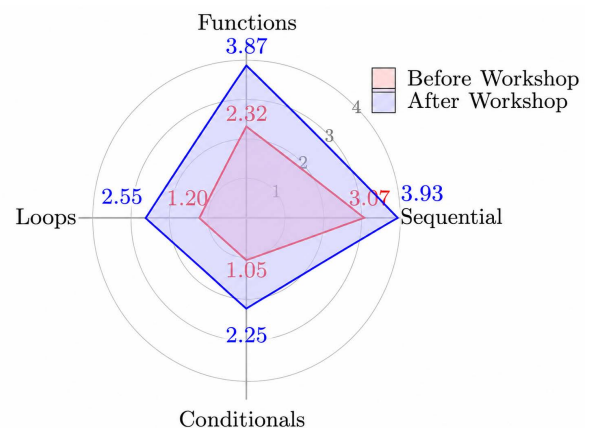


Fig. 8. Average performance by programming skill domain before and after intervention (scale = 4, $n = 44$).

Table 7. Average performance by programming skill domain ($n = 44$)

Skill Domain	Before Workshop	After Workshop
Sequential	3.07	3.93
Functions	2.32	3.87
Loops	1.20	2.55
Conditionals	1.05	2.25

C. Homogenization and Compensatory Effect

The intervention exhibited a compensatory effect across performance levels. The scatter plot in Fig. 9 shows an inverse relationship between students' initial scores and their individual learning gains, indicating that lower-performing students benefited proportionally more from the intervention.

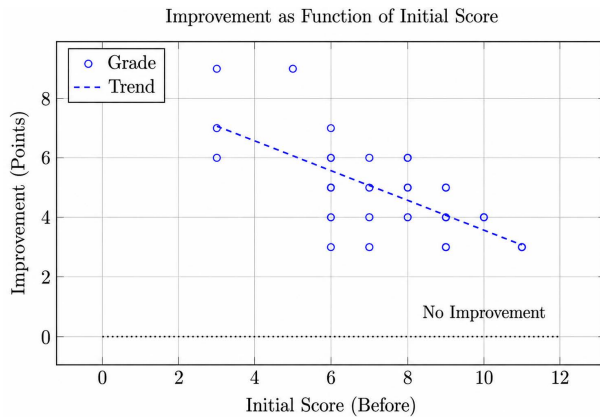


Fig. 9. Relationship between initial scores and individual improvement.

Students with low initial performance (pre-test scores around 3) achieved gains between 6 and 9 points, whereas students with higher initial performance (scores of 10–11) showed smaller but consistent improvements. This negative relationship suggests a convergence of learning outcomes, supporting the reduction in score variability observed in the post-test.

These results indicate that the proposed tool does not disproportionately benefit already proficient students, but instead supports learners with lower initial knowledge, contributing to a more homogeneous distribution of computational thinking skills across the cohort.

V. DISCUSSION

The exceptionally large effect size (Cohen's $d = 3.23$), compared to conventional reference parameters reported in educational research [35, 36], requires careful and contextualized interpretation. While this magnitude reflects a genuine improvement in student performance, it is important to recognize that certain methodological characteristics of the study may have contributed to amplifying this value. Specifically, the relatively short duration between intervention sessions, the strong concordance between the skills practiced in the game and those assessed in the subsequent test, and the restricted scoring range of the instrument (maximum of 14 points) are factors that could have influenced the effect size in educational research, regardless of the actual learning gains. On the other hand, several pedagogical factors further support the observed result. First, a strong concordance was observed between the intervention and the assessment instrument, as the post-test directly assessed the same problem-solving, sequencing, and algorithmic reasoning skills that students continuously practiced in the interactive environment. This reduced measurement noise and improved sensitivity to learning progress.

Second, the tool functioned as an intensive active learning environment. Students continuously engaged in planning, executing, and debugging tasks within a simulation. Prior

research on game-based and simulation-based learning has shown that such high-engagement, task-centered environments tend to produce greater learning gains than traditional lecture-based instruction, especially when immediate feedback is provided [5, 15, 18]

Third, the reduction in score variability indicates a homogenization of learning outcomes. This suggests that the intervention benefited all students, including those with lower initial performance, contributing to the large effect size.

Additionally, the non-normal distribution of post-test scores is attributed to a ceiling effect: a substantial proportion of students achieved near-maximum performance. Rather than a limitation, this ceiling effect provides additional evidence that the learning objectives, a finding supported by well-controlled research with exceptionally large effect sizes [35, 36]; whose factors indicate that the observed effect size reflects a genuine educational impact rather than a statistical artifact.

Incidentally, it is also important to reflect on the conditions under which the current results were obtained and to identify both the strengths and potential limitations of this proposed tool. Among its strengths, isometric simulation provides immediate visual feedback, allowing for the direct observation of abstract programming constructs, such as loops, conditionals, and functions, through the execution of the Bot's movements. This substantially reduces the cognitive load typically associated with syntax-based programming environments. The tool is also highly configurable, allowing instructors to design customized scenarios with varying challenges of complexity, making it suitable for differentiated instruction for diverse student profiles. Furthermore, the compensatory effect documented in this study suggests that the tool is particularly beneficial for all students, including those with lower initial proficiency, thus promoting more equitable learning outcomes. Regarding its limitations, the tool's effectiveness is most pronounced when there is a close correspondence between the skills practiced in the game and the competencies assessed in the evaluation instrument. Educators using it as a supplementary resource within a broader curriculum—where assessments cover skills beyond those addressed in the game—should expect minor effects. Furthermore, the short-term nature of this intervention means that long-term retention of the acquired skills has not yet been determined. Finally, instructors with limited experience in game-based learning may need initial support in designing levels that appropriately facilitate student progress and maintain their interest without compromising educational rigor.

On the other hand, in Skill Acquisition by Programming Domain, the analysis shows that the interactive tool had varying effects on computational thinking skills. Sequential Logic improved moderately (28%), suggesting that visual interfaces help learners quickly grasp basic procedural reasoning by reducing cognitive load. Functions showed a larger improvement (67%), indicating the tool supports abstraction and modular thinking, which are often challenging for beginners. The greatest gains were seen in Loops and Conditionals (113% and 114%), likely because the tool's visual and interactive feedback makes complex control

structures easier to understand. Overall, the results suggest that interactive, simulation-based environments are especially effective for teaching complex computational concepts and bridging the gap between concrete and abstract reasoning in early programming education.

VI. CONCLUSION

This study demonstrates that the proposed interactive Bot command game based on an isometric visualization with levels of complexity provides promising evidence for improving computational thinking in secondary school students. Unlike traditional syntax-based approaches, this tool provides a structured graphical environment where abstract concepts—sequences, loops, functions, and conditionals—are visualized through executable blocks that translate into visual results.

In addition, the quantitative analysis revealed a positive impact, with the overall mean score increasing from 7.64 to 12.41, with an effect size of 3.23. This value far exceeds standard educational benchmarks, indicating a substantial impact on student learning.

Two key factors explain this success:

- (1) The implementation of the 2:1 isometric projection successfully reduced the cognitive load associated with spatial orientation. This allowed students to direct their mental resources toward algorithmic planning rather than struggling with perspective shifts, validating the pedagogical choice of a stable spatial reference frame.
- (2) The study also reveals a “leveling up” phenomenon. Students with the lower initial proficiency achieved the highest relative gains (200–300%), converging toward the group average. This demonstrates that the tool can effectively democratize access to programming logic, bridging the gap for students who typically struggle in traditional educational settings.

In conclusion, this research demonstrates that combining gamification with mathematically grounded visual design (isometric projection) significantly enhances computational thinking.

To address the ceiling effect observed in the post-test and to continue measuring the progress of students who achieve high competency levels early, future iterations of the game will incorporate advanced complexity tiers. These new challenges will introduce nested control structures, such as integrating loops within functions and applying complex conditional logic to navigate around dynamic, moving obstacles on the isometric board. Furthermore, optimization challenges will be implemented, requiring students to solve mazes under strict constraints, such as a limited number of instruction slots or execution time. This progressive escalation in difficulty will ensure that advanced learners remain cognitively engaged and that their evolving computational thinking skills can be accurately assessed beyond the current baseline.

Finally, future work will focus on integrating collaborative multiplayer modules to assess peer-learning effects. Furthermore, to validate the scalability and generalizability of these results, we plan to expand the study to a larger, multi-school demographic. This will involve deploying the interactive tool across broader demographics, including diverse socioeconomic backgrounds, different age groups

within secondary education, and various regional educational settings. Crucially, to address the methodological limitations of the current single-group design and strengthen causal claims, these future studies will employ quasi-experimental designs with active control groups, comparing our gamified tool directly against traditional syntax-based instructional methods.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Wilson Mamani-Casilla and Edilson Mamani-Yucra carried out the software development, system architecture design, the implementation of the experimental workshop at the school, and the formal data analysis. Yuber Velazco-Paredes and Roxana Flores-Quispe were responsible for proposing and validating the educational methodology and acted as writing and research advisors, providing critical feedback and editorial guidance, as well as reviewing, editing, and finalizing the article. All authors contributed to the conceptualization of the research, the writing of the article, and reviewed and approved the final manuscript.

FUNDING

This research was funded by the Universidad Nacional de San Agustín de Arequipa, Perú, under the Applied Research Project “Development of an Interactive Robotics-Based Tool for Learning Programming” under contract number PI-011-2023-UNSA. This article was written as part of this project.

ACKNOWLEDGMENT

The authors thank the Universidad Nacional de San Agustín de Arequipa, Perú, for funding this research through the Applied Research Project “Development of an Interactive Robotics-Based Tool for Learning Programming” with contract number PI-011-2023-UNSA and the Honorio Delgado Espinoza Emblematic School of the city of Arequipa for allowing us to carry out experimental tests with 5th grade students, sections B and C, class of 2025, and especially Professor Roberto Valdivia for his support and coordination.

REFERENCES

- [1] I. Eteng, S. Akpotuzor, S. O. Akinola, and I. Agbonlahor, “A review on effective approach to teaching computer programming to undergraduates in developing countries,” *Scientific African*, vol. 16, e01240, 2022. doi: 10.1016/j.sciaf.2022.e01240
- [2] B. Wilkinson, N. Williams, and P. Armstrong, “Improving student understanding, application and synthesis of computer programming concepts with Minecraft,” in *Proc. Eur. Conf. Technol. Classr. (ECTC)*, Brighton, U.K., July 11–14, 2013.
- [3] B. Law, “Teaching programming using computer games: A program language agnostic approach,” in *Proc. 11th Eur. Conf. Games Based Learning (ECGBL)*, Graz, Austria, 2017, pp. 368–376.
- [4] N. I. Prasetya and N. Saurina, “Game-based learning in programming courses: A platform development for enhanced learning engagement,” *Int. J. Inf. Educ. Technol.*, vol. 15, no. 12, pp. 2593–2602, 2025. doi: 10.18178/ijiet.2025.15.12.2455
- [5] W. Y. Seng and M. H. M. Yatim, “Computer game as learning and teaching tool for object oriented programming in higher education institution,” *Procedia - Social and Behavioral Sciences*, vol. 123,

- pp. 215–224, 2014. doi: 10.1016/j.sbspro.2014.01.1417
- [6] N. K. Sharda, “Designing, using and evaluating educational games: challenges, some solutions and future research,” in *Proc. the First International Workshop on Story-Telling and Educational Games (STEG’08): The Power of Narration and Imagination in Technology Enhanced Learning*, CEUR-WS, vol. 386, 2008.
- [7] A. Ghosh, “Challenges in educational game development,” *International Journal of Darshan Institute on Engineering Research and Emerging Technologies*, vol. 11, no. 1, pp. 54–60, 2022. doi: 10.32692/IJDI-ERET/11.1.2022.2209
- [8] B. Tassuov, A. Nadyrbekova, K. Ibragimova, Z. Tashhurekova, N. Niyetbayeva, A. Abzhapparova, and Z. Aidaraliyeva, “Educational games as a tool for teaching programming in digital extracurricular computer science education,” *Frontiers in Computer Science*, 2026. doi: 10.3389/fcomp.2026.1766830
- [9] P. J. Denning and M. Tedre, “Computational thinking: A disciplinary perspective,” *Informatics in Education*, vol. 20, pp. 361–390, 2021. doi: 10.15388/infedu.2021.21
- [10] F. J. Agbo, S. S. Oyeler, J. Suhonen, and T. H. Laine, “Co-design of mini games for learning computational thinking in an online environment,” *Education and Information Technologies*, vol. 26, no. 5, pp. 5815–5849, 2021. doi: 10.1007/s10639-021-10515-1
- [11] C.-C. Wu, S.-P. Chien, T.-L. Chou, and Y.-M. Huang, “Integrating computational thinking, game design, and design thinking: A scoping review,” *Humanities and Social Sciences Communications*, vol. 12, no. 1, pp. 1–12, 163, 2025. doi: 10.1057/s41599-025-04502-x
- [12] C. Zorn, C. A. Wingrave, E. Charbonneau, and J. J. LaViola Jr., “Exploring Minecraft as a conduit for increasing interest in programming,” in *Proc. 8th International Conference on Foundations of Digital Games (FDG)*, Chania, Greece, May 2013, pp. 352–359.
- [13] A. Theodoropoulos, A. Antoniou, and G. Lepouras, “How do different cognitive styles affect learning programming? Insights from a gamebased approach in greek schools,” *ACM Transactions on Computing Education (TOCE)*, vol. 17, no. 1, 3, pp. 1–25, Sept. 2016. doi: 10.1145/2940330
- [14] D. Hooshyar, L. Malva, Y. Yang, M. Pedaste, M. Wang, H. Lim, “An adaptive educational computer game: Effects on students’ knowledge and learning attitude in computational thinking,” *Computers in Human Behavior*, vol. 114, no. 6, 2020. doi: 10.1016/j.chb.2020.106575
- [15] S. Zhang, G. K. W. Wong, and P. C. F. Chan, “Playing coding games to learn computational thinking: What motivates students to use this tool at home?” *Education and Information Technologies*, vol. 28, no. 1, pp. 193–216, 2023. doi: 10.1007/s10639-022-11181-7
- [16] E. V. Soboleva, E. G. Sabirova, N. S. Babieva, M. G. Sergeeva, and J. V. Torkunova, “Formation of computational thinking skills using computer games in teaching mathematics,” *Educational Media International*, vol. 17, no. 10, em2012, 2021. doi: 10.29333/ejmste/11177
- [17] M. S. Kovtaniuk, S. V. Shokaliuk, and A. N. Stepanyuk, “Game simulators as educational tools for developing algorithmic thinking skills in computer science education,” in *Proc. CTE Workshop*, vol. 12, Mar. 2025, pp. 19–62. doi: 10.55056/cte.926
- [18] M. Videnovik, T. Vold, L. Kionig, A. Madevska Bogdanova, and V. Trajkovik, “Game-based learning in computer science education: A scoping literature review,” *International Journal of STEM Education*, vol. 10, 54, 2023. doi: 10.1186/s40594-023-00447-2
- [19] J. Ma, Y. Zhang, Z. Zhu, S. Zhao, and Q. Wang, “Game-based learning for students’ computational thinking: A meta-analysis,” *Journal of Educational Computing Research*, vol. 61, no. 7, pp. 1430–1463, 2023. doi: 10.1177/07356331231178948
- [20] M. F. C. A. Rani, N. H. Adnan, A. Z. Mansor, M. M. Yunus, I. E. Mourabit, and T. C. Ike, “Needs analysis for developing an educational hybrid board game for python programming,” *Int. J. Inf. Educ. Technol.*, vol. 15, no. 12, pp. 2603–2612, 2025. doi: 10.18178/ijiet.2025.15.12.2456
- [21] A. M. N. Callins, D. A. C. Carmo, and J. C. C. Furtado, “An analysis of the application of games for teaching basic programming: A systematic mapping of the literature,” *Revista Aracé*, vol. 7, no. 1, pp. 2872–2894, 2025. doi: 10.56238/arev7n1-176
- [22] A. Lathifah, A. Asrowi, and A. Efendi, “Students’ perspectives on game-based learning and computational thinking,” *Int. J. Inf. Educ. Technol.*, vol. 13, no. 3, pp. 597–603, 2023. doi: 10.18178/ijiet.2023.13.3.1843
- [23] A. I. Zourmpakis, “Developing computational thinking in early childhood education: Long-term impacts on CT skills and motivation using the CAL approach, ScratchJr, and gamification,” *Advances in Mobile Learning Educational Research*, vol. 5, no. 2, pp. 1536–1547, 2025. doi: 10.25082/AMLER.2025.02.009
- [24] R. M. AlWahaibi, A. M. Aldhowayan, S. M. Alshahrani, B. S. Almadi, R. A. Alamer, F. M. Albaqami, and E. M. Mortada, “Exploring the Influence of video games on self-reported spatial abilities among university students,” *Brain Sciences*, vol. 14, no. 12, 1269, Dec. 2024. doi: 10.3390/brainsci14121269
- [25] H. Park, M. K. Dixit, and F. Pariafsai, “Towards personally relevant navigation: The differential effects of cognitive style and map orientation on spatial knowledge development,” *Appl. Sci.*, vol. 14, no. 10, 4012, May 2024. doi: 10.3390/app14104012.3
- [26] Y. Li, Y. Yang, Z. Yao, and G. Xu, “Virtual 3D environment for exploring the spatial ability of students,” *Virtual Reality & Intelligent Hardware*, vol. 2, no. 6, pp. 556–568, 2020.
- [27] M. Delavar, I. V. Kolesnikova, B. Rahimzade, M. Ghahhari, A. Mosapuor, and A. Moradi, “Development of mental rotation ability at primary school level,” *World Sci. News*, vol. 101, pp. 77–88, Jun. 2018.
- [28] E. X. Wei, E. R. Anson, S. M. Resnick, and Y. Agrawal, “Psychometric tests and spatial navigation: data from the baltimore longitudinal study of aging,” *Frontiers in Neurology*, vol. 11, 484, Jun. 2020. doi: 10.3389/fneur.2020.00484
- [29] M. C. Stites, L. E. Matzen, and Z. N. Gastelum, “Where are we going and where have we been? Examining the effects of maps on spatial learning in an indoor guided navigation task,” *Cognitive Research: Principles and Implications*, vol. 5, 13, 2020. doi: 10.1186/s41235-020-00213-w
- [30] M. Román-Gonzalez, J. C. Pérez-González, and C. Jiménez-Fernández, “Computational thinking test: Design general psychometry,” in *Proc. 3rd International Congress on Learning, Innovation, and Competitiveness*, 2015. doi: 10.13140/RG.2.1.3056.5521
- [31] E. González-Estrada, J. A. Villaseñor, and R. Acosta-Pech, “Shapiro-Wilk test for multivariate skew-normality,” *Computational Statistics*, vol. 37, pp. 1985–2001, 2022. doi: 10.1007/s00180-021-01179-z
- [32] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed., Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 1988.
- [33] A. Field, *Discovering Statistics Using IBM SPSS Statistics*, 5th ed., Newbury Park: Sage, 2017.
- [34] H. Alvarado and C. Batanero, “Meaning of the central limit theorem in university probability and statistics textbooks,” *Estudios Pedagógicos*, vol. 34, no. 2, pp. 7–28, 2008. doi: 10.4067/S0718-07052008000200001
- [35] J. Hattie, *Visible Learning: A Synthesis of over 800 Meta-Analyses Relating to Achievement*, Routledge, 2009.
- [36] M. A. Kraft, “The effect size benchmark that matters most: Education interventions often fail,” *Educational Researcher*, vol. 52, no. 3, pp. 183–187, 2023. doi: 10.3102/0013189x231155154

Copyright © 2026 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).