

Code Trials: An Approach to Teaching Programming Logic Through Platform Game Mechanics

José Kaynã Pereira de Sousa Bitencourt¹, Jose Olimpio de Freitas Dias Junior¹,
Julio Cezar Costa Furtado¹

¹Departamento de Ciências Exatas e Tecnológicas (DCET)
Universidade Federal do Amapá (UNIFAP)
Rod. Juscelino Kubitschek, KM-02 – 68.903-419 – Macapá – AP – Brasil

{josekayna13, olimpiokbz}@gmail.com, furtado@unifap.br

Abstract. *Introduction:* Teaching programming logic in introductory Computer Science courses faces persistent challenges related to concept abstraction, student disengagement, and high failure rates, with game-based learning identified as an effective strategy. **Objective:** This paper presents Code Trials, a 2D platform game developed in Godot 4 to support the teaching of programming logic, targeting sequencing, conditionals, and repetition loops. **Stages:** The game was developed through design-based research in three phases: pedagogical design, implementation in Godot 4 with a modular architecture separating pedagogical parameters from engine logic; and internal validation through iterative testing cycles with undergraduate students. **Expected Results:** Code Trials is expected to reduce cognitive load related to syntax, increase engagement and time on task, and improve learning outcomes compared to traditional approaches.

Keywords: educational games; programming education; programming logic; platform game; block-based programming.

1. INTRODUCTION

The introduction of programming logic in Computer Science is one of the oldest pedagogical challenges for introductory courses. Newcomers find themselves unable to grasp the abstraction behind fundamental notions such as sequential processing, decision making, and repetition loops, especially when faced with educational strategies that focus more on syntactic correctness than on conceptual development. Dropout and failure rates are persistently high in various academic environments [Hosseini et al. 2019; Berssanette and Francisco 2021; Calderon et al. 2023; Gomes and Mendes 2007]. These two problems can be solved simultaneously through game-based learning (GBL). In fact, this educational technique contextualizes the concept acquisition process in meaningful problem-solving activities [Prensky 2001; Gee 2003].

In this paper we present Code Trials, a game-based learning environment designed to teach programming logic in introductory courses. The 2D platform game comprises eight tutorial levels structured into three conceptual arcs corresponding to sequencing, conditionals, and repetition loops, which serve as scaffolded preparation for the Infinite Mode, a procedurally generated practice environment that provides unlimited challenge at three difficulty tiers. It rests on one basic premise: there is no input from the player to the character. Instead, the player writes a program by dragging-and-dropping visual blocks, and then waits for the character to execute it fully before doing anything else.

There is no way for the student to interact with the game world except through programming; and there is no way to receive information about the correctness of the program aside from watching how the character behaves. The game does not teach concepts through exposition; it sets the conditions necessary to make their acquisition happen in the context of solving the problem.

2. RELATED WORK

Three most similar studies to Code Trials include Bug Quest [Liberalino et al. 2025], the natur(Eco) prototype [Silva et al. 2025], and Digital Logic [Costa et al. 2025]. Bug Quest is an analog game intended to teach computer science students the basics of programming. On the other hand, natur(Eco) is an educational 2D open-world game built in Unity with the aim of teaching children aged six to ten the principles of computational thinking (CT). Each of the four CT skills is represented in an individual minigame. Digital Logic is a digital game built by means of game modification of an existing open-source game engine. The focus of the game is propositional logic, connectives, and truth tables and can be used by Computer Science majors. A comparative analysis is provided in Table 1 below.

Table 1. Comparison between related works and Code Trials.

Dimension	Bug Quest	natur(Eco)	Digital Logic	Code Trials
Medium	Analog (card game)	Digital (Unity)	Digital (modded JRPG)	Digital (Godot 4)
Genre	Board game	Open-world / minigames	JRPG (narrative RPG)	2D platform game
Programming interface	Physical cards (directional, loops, functions)	Drag-and-drop command blocks	None (quiz/dialogue-based)	Drag-and-drop labeled blocks
Concepts covered	Sequencing, loops, functions	Algorithms, decomposition, abstraction, pattern recognition	Propositional logic, connectives, truth tables	Sequencing, conditionals, loops
Pedagogical progression	No structured sequence; community-driven	Modular (one CT skill per minigame); open-world order	Narrative-driven; content in story progression	8 tutorial levels (sequencing, conditionals, loops) + Infinite Mode (procedural, 3 difficulty tiers)
Feedback mechanism	Verbal execution; peer evaluation during play	Visual and audio per minigame action	Correct/incorrect per question; narrative advancement	Embodied: character behavior; step-by-step execution controls; star objectives; persistent progress across sessions
Target audience	Higher education (CS undergraduates)	Elementary school (ages 6-10)	Higher education (CS undergraduates)	Higher education (introductory programming)

3. METHODOLOGY

Code Trials was developed following a design-based research (DBR) methodology, structured into three phases: pedagogical design, implementation, and internal validation.

In the pedagogical design phase, learning objectives were derived from standard introductory programming curricula and organized into the three conceptual arcs, with each arc introducing a single new construct while preserving the ones previously taught. The implementation phase used the Godot 4 engine and a modular architecture that separates pedagogical parameters, such as available blocks, objectives, and constraints, from the engine logic, and relies on a sequential interpreter that ensures a deterministic correspondence between the constructed program and the character's behavior. The internal validation phase consisted of iterative testing cycles conducted by undergraduate computer science students under faculty supervision, focused on interface usability, clarity of interaction, and alignment between intended learning outcomes and observed player behavior. Feedback obtained in this phase guided successive refinements in block design, drag-and-drop interaction, and tutorial structure.

4. CODE TRIALS

4.1. Game Concept

Code Trials is a 2D platform game, created in Godot 4, which allows deployment on several operating systems due to Godot's features. It was intended as an auxiliary application for introductory courses in programming logic at universities. The game is organized into two complementary components: eight tutorial levels that guide the student progressively through the three core programming concepts, and the Infinite Mode, a procedurally generated practice environment that serves as the main application destination once the tutorial sequence is completed.

A distinguishing feature of the game is its level goals system, where each level comes with three independent objectives that grant stars upon completion: getting to the terminal (the main goal), using a certain code block necessary for the current level, and finishing a level using the minimum amount of blocks.

4.2. Mechanics

Mechanics of Code Trials game depend on two important elements: environment and block-based programming interface. Environment is a 2D plane containing static platforms, dynamic platforms, spikes and target platform. Player's character does not respond to any user input during gameplay, it reacts strictly to program written using blocks. Programming interface consists of two panels: commands available (panel with blocks that can be used in certain level), and blocks composition area (where players compose the blocks using drag-and-drop). The available blocks cover sequencing (movement, turning, jumping, stopping, and waiting), conditionals, and repetition.

After finishing eight tutorial levels, student gets access to the Infinite Mode, which he can choose from level selection screen depending on desired difficulty (easy, medium and hard). This game generates maps of different difficulty procedurally with random placement of terrain, obstacles and platforms according to the chosen one. For example, easy maps have elevation and require control over direction of motion, while medium maps contain pits, spikes, and hard maps all the previously described features plus moving platforms. Generated map always has at least one correct solution because there is a built-in check for its existence during map generation process. Infinite Mode creates conditions for practicing learned skills outside tutorial levels: since student already got acquainted with all three conceptual arcs, it provides infinite variety of

exercises where all of them must be used simultaneously. The programming interface also supports textual input of commands in a separate tab, parsed into the same internal representation as the visual editor, which facilitates a gradual transition from block-based to text-based programming within the same framework [Weintrop and Wilensky 2017]. Figure 2 illustrates the possibility of programming both through block-based commands and pseudocode.



Figure 1. Code Trials main screen: platform environment (center), block programming panel (left), and level objectives (right).



Figure 2. Code Trials interface showing dual programming modalities, allowing interaction through block-based commands and text-based programming.

4.3. Feedback

Code Trials combines two complementary layers of feedback. The first layer is behavioral feedback grounded in embodiment: the character's movement is a literal translation of the assembled program into action, so any logical flaw is communicated through the character's observable behavior rather than through external messages. In Pérez et al.'s (2018) terms, this is intrinsic feedback, apparent in real time from the consequences of the program [Calderon et al. 2023].

The second layer is based on the player's session objectives, which include finishing the level, using a specified command, and assembling a solution within a maximum number of blocks. The three criteria are displayed throughout the session and dynamically update once each is achieved or failed, producing a multi-layered achievement structure that encourages abstract thinking and code optimization. This feedback system is augmented by a built-in tutorial that provides explicit instructions at the first level and fades into the background in subsequent ones, limiting explanations to each newly introduced block [Malliarakis et al. 2017; Vygotsky 1978].

5. EDUCATIONAL POTENTIAL

Sequences of Code Trials are conceptually consistent with fundamental topics taught in an introductory programming course [Gomes and Mendes 2007; Berssanette and Francisco 2021]. Sequencing is essential in solving problems in beginner levels, conditional statements are indispensable when it comes to solving problems involving dynamic obstacles, and loops represent the best approach to dealing with repeated actions in advanced levels. The pedagogical design of the game relies on four interconnected theories. Kolb's experiential learning cycle [Kolb 1984] unfolds in seconds at every execution: active experimentation (coding), concrete experience (character behavior), reflective observation, and abstract conceptualization. Lave and Wenger's situated learning [Lave and Wenger 1991] is manifested in the inversion of the order of code construction, so that concepts are introduced only when required.

Each concept acquires meaning while solving a particular problem, following Gee's framework of pleasant learning [Gee 2003]. Vygotsky's zone of proximal development [Vygotsky 1978] is embodied in the progressive unlocking of blocks and in a tutorial system that fades into the background as levels advance [Malliarakis et al. 2017]. Finally, following Pérez et al. [Pérez et al. 2018], Code Trials uses only intrinsic feedback, embodied in the character's behavior, which favors the construction of a causal model of the system [Hosseini et al. 2019].

6. CONCLUSIONS AND FUTURE WORK

This paper presented Code Trials, a platform game developed in Godot 4 that teaches programming logic by making the student-created program the only way to interact with the environment, thus avoiding the syntax barrier, situating concepts in problem contexts, and providing immediate embodied feedback. The game comprises eight tutorial levels followed by the Infinite Mode, a procedurally generated practice environment with three difficulty tiers, and offers a dual block/text programming interface within the same framework. Future work includes a controlled experiment with pretests and posttests in introductory programming classes, the introduction of adaptive difficulty in the Infinite Mode [Hosseini et al. 2019], and empirical research on the transition from visual to text-based programming through the dual interface [Weintrop and Wilensky 2017].

REFERENCES

- Berssanette, J. and Francisco, A. (2021). Active learning methodologies in the context of teaching and learning computer programming: a systematic literature review. *Educitec - Revista de Estudos e Pesquisas sobre Ensino Tecnológico*, v. 7, e159821.
- Calderon, I.; Silva, W. and Feitosa, E. (2023). Active learning methodologies for teaching programming in undergraduate courses: a systematic mapping study. *Informatics in Education*, v. 0, n. 0.
- Costa, C. M.; Costa, R. M.; Silva, J. P. and Silveira, I. F. (2025). Digital Logic: Um Jogo RPG Baseado em Modding Para o Ensino e Aprendizagem de Lógica Computacional. In: *Anais do XXIV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2025)*, Salvador/BA, p. 1887–1897.
- Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy*. New York: Palgrave Macmillan.
- Gomes, A. and Mendes, A. J. (2007). Learning to program: difficulties and solutions. In: *Proceedings of the International Conference on Engineering Education (ICEE 2007)*, Coimbra, Portugal.
- Hosseini, H.; Hartt, M. and Mostafapour, M. (2019). Learning is child's play: game-based learning in computer science education. *ACM Transactions on Computing Education*, v. 19, n. 3, p. 1–18.
- Kolb, D. A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall.
- Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.
- Liberalino, C. H. P.; Nunes, D. F. L.; Silva, E. C. P.; Laureano, R. S.; Sousa, G. C. M. and Pinto, M. E. S. (2025). Bug Quest: Um Jogo Para o Ensino da Programação. In: *Anais do XXIV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2025)*, Salvador/BA, p. 1411–1420.
- Malliarakis, C.; Satratzemi, M. and Xinogalos, S. (2017). CMX: the effects of an educational MMORPG on learning and teaching computer programming. *IEEE Transactions on Learning Technologies*, v. 10, n. 3, p. 219–233.
- Pérez, E. del M.; Duque, A. P. G. and García, L. C. F. (2018). Game-based learning: increasing the logical-mathematical, naturalistic, and linguistic learning levels of primary school students. *Journal of New Approaches in Educational Research*, v. 7, n. 1, p. 31–39.
- Prensky, M. (2001). *Digital Game-Based Learning*. New York: McGraw-Hill.
- Silva, G.; Silva, I. N. M.; Carvalho, W. R. B.; Pessoa, J.; Moreira, J. H. L.; Ferreira, F. Y.; Brandão, A. L. and Vittori, K. (2025). Um Protótipo em Mundo Aberto para o Ensino de Pensamento Computacional no Ensino Fundamental. In: *Anais do XXIV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames 2025)*, Salvador/BA, p. 159–168.
- Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Cambridge: Harvard University Press.
- Weintrop, D. and Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education*, v. 18, n. 1, p. 1–25.