A Dual-Agent Adaptive Arcade Mode for Rapid Code Evaluation in Programming Education

William Martins[1], **Kleydson Beckman Barbosa**[1,2], **Allan K. B. S. da Cruz**[1], **Sávio Câmara**[3], **Carlos de Salles S. Neto**[1,2,3]

[1] TeleMídia@MA Lab – Federal University of Maranhão (UFMA)
São Luís – Maranhão – Brasil

[2]Postgraduate Program in Computer Science – Federal University of Maranhão (UFMA)
São Luís – Maranhão – Brasil

[3]Pontifical Catholic University (PUC)
Rio de Janeiro – Rio de Janeiro – Brasil

{william.valther, kleydson.beckman}@discente.ufma.br,

{saviopatrickc, allankassio}@gmail.com,

carlos.salles@ufma.br

Abstract. Introduction: *Programming education often emphasizes code construction, but offers fewer opportunities for students to practice rapid code evaluation, a skill related to debugging, code review, maintenance, and identification of bugs and bad practices.* ***Objective:*** *This paper investigates preliminary evidence on the use of an adaptive arcade-style mini-game for rapid code evaluation in introductory programming contexts.* ***Methodology:*** *The mini-game asks players to compare two code snippets and select the correct alternative under time pressure while a fuel resource decreases. Questions are selected by a dual-agent architecture inspired by minimax, balancing diversity and difficulty progression through gameplay telemetry.* ***Results:*** *The evaluation involved 13 active participants, 36 valid sessions, and 1,375 answers. Global accuracy was 87.3%, with an average response time of 3.35 seconds. From the first to the last valid session, mean accuracy increased from 73.9% to 80.1%, mean distance from 30,270 to 44,353, and mean maximum combo from 14.8 to 23.6. Results also showed differences among mechanics and dominance of the progression agent, suggesting the need for utility normalization, parameter tuning, and future comparison with baseline selection strategies.*
Keywords *Game-based learning, Programming education, Adaptive learning, Code evaluation, Question selection.*

1. Introduction

Learning computer programming involves more than memorizing syntax or writing functional solutions to well-defined problems. Students also need to develop the ability to read, interpret, and evaluate code written by others, identifying errors, inconsistencies, bad practices, and recurring defect patterns [Lister et al. 2004]. This competence is central to professional activities such as debugging, code review, software maintenance, and refactoring, in which programmers frequently need to quickly judge whether a

code snippet is correct, appropriate, or potentially problematic [Fowler et al. 1999, Lozano et al. 2024].

Despite this relevance, many programming learning environments still emphasize code construction tasks, such as completing exercises, submitting solutions to automated judges, or implementing algorithms from problem statements. Although these practices are important for developing implementation skills, they offer fewer opportunities for students to practice the rapid evaluation of existing code. This limitation is reinforced by the systematic review conducted by [Keuning et al. 2018], which showed that automated feedback in programming education tools predominantly focuses on error identification and functional correctness, while less frequently supporting students in understanding how to improve code quality or solve problems.

This study has two complementary objectives: to design and implement an adaptive arcade-style mini-game for rapid code evaluation, and to analyze gameplay telemetry to examine performance, response time, progression patterns, and selector behavior. In each round, students compare two code snippets under time pressure, while survival, feedback, distance, combo, and difficulty progression support repeated practice with syntax, assignment, comparison, loop, index, logic, type, indentation, scope, and best-practice issues.

The main distinctive feature is the adaptive question selection mechanism. Instead of using a fixed sequence, random choice, or a single difficulty rule, the system adopts a dual-agent architecture inspired by minimax: one agent prioritizes diversity and novelty, while the other prioritizes difficulty progression and pedagogical gain. An arbitration function combines these utilities with recent history, frustration risk, difficulty adequacy, and expected learning value.

Based on these objectives, this work investigates the following research question: what preliminary evidence does an adaptive arcade-style mini-game, supported by adversarial question selection and gameplay telemetry, provide regarding the practice of rapid code evaluation in introductory programming contexts? The main contributions are: (i) an arcade-style learning activity for rapid code evaluation; (ii) a computational architecture for adaptive content selection based on two internal agents; (iii) a telemetry-based dynamic calibration mechanism for question difficulty; and (iv) an empirical analysis of performance, response time, mechanic type, difficulty level, template coverage, and selector behavior.

2. Theoretical Background and Related Work

The design of educational games requires more than adding motivational elements to conventional learning tasks. Serious games and gamified learning environments tend to be more effective when game mechanics are grounded in learning theory and aligned with explicit instructional goals [Abd El-Sattar 2023, Chou et al. 2023]. Flow theory and digital game-based learning help explain how challenge, skill, feedback, attention, and engagement interact during gameplay [Stidham 2022, Pan et al. 2025]. However, flow or gamification alone does not guarantee learning: structured feedback, reflection, task design, and cognitive alignment remain necessary for educational value [Singletary 2025, Takács et al. 2023, Scaico et al. 2024]. Therefore, mechanics such as challenge, clear goals, feedback, progression, and sustained concentration are pedagogically relevant

when intentionally connected to the learning task [Malone 1981, Sweetser e Wyeth 2005, Kiili 2005]. This is central to the present work, since the proposed mini-game links its main action—comparing two code snippets—to the intended cognitive skill: rapid code evaluation.

Adaptive learning and AI-supported educational technologies reinforce the importance of regulating challenge according to learner behavior. Recent work has explored real-time difficulty adjustment, learner modeling, personalized pathways, intelligent feedback, and adaptive game features [Idika e Saihi 2025, Gkintoni et al. 2025, Loor et al. 2024, Zhong e Zhou 2024]. These systems should be evaluated not only by technical performance, but also by their support for interactivity, engagement, accessibility, and pedagogical goals [Sánchez et al. 2026]. The present work follows this direction, but applies adaptation to a specific programming education problem: selecting the next code-evaluation challenge during a time-constrained arcade session.

Taken together, the literature supports three design assumptions for this study: educational games should connect mechanics to the intended cognitive process; engagement should be balanced with feedback, challenge regulation, and opportunities for correction; and adaptive systems should consider multiple pedagogical objectives rather than optimize a single score. Existing approaches commonly adapt quiz difficulty, tutoring strategies, platform features, or course configurations, but rarely target rapid code evaluation as a specific cognitive skill. This paper addresses that gap by proposing an arcade-style programming mini-game in which learners compare code snippets under time pressure, while a dual-agent selector balances novelty, repetition, difficulty, frustration risk, and expected pedagogical value.

3. Methodology

This study adopts an applied, design-oriented methodology aimed at designing, implementing, and analyzing a game-based training artifact for programming education. The methodological framing is informed by Design-Based Research (DBR), since the work involves developing an educational technology artifact, implementing it in an authentic learning context, and analyzing interaction data produced during use [Koh et al. 2024]. It is also aligned with Technology-Enhanced Learning (TEL), in which technology, pedagogy, learner activity, and assessment are treated as interdependent components of a designed learning environment [Kyza 2023, Martel e Garcias 2024].

The research question guided the methodological structure by connecting the design of the arcade-style mini-game, the adaptive question selection mechanism, and the empirical analysis of gameplay telemetry. The methodology was organized into three design stages: defining the pedagogical proposal, modeling the adversarial selector, and implementing dynamic calibration and telemetry mechanisms. The empirical evaluation is described separately in Section 4.

3.1. Implementation Context

The arcade mini-game was implemented as a feature of a gamified programming learning platform organized around courses, concepts, and activities. Each course is divided into curricular concepts, and student progression is sequential. Arcade sessions are linked

to the student's active concept; therefore, the selector restricts candidate templates to that concept when a session starts. This design makes the mini-game a complementary curricular activity and is important for interpreting the results in Section 5, since the distribution of answers reflects participants' curricular progression rather than free topic choice.

3.2. Pedagogical Proposal of the Arcade Mini-Game

The arcade mini-game was designed to support the practice of rapid code evaluation. Unlike exercises in which students must write a complete solution, the proposed activity emphasizes reading, comparison, and judgment of existing code snippets. In each round, two alternatives are displayed, one correct and one incorrect or less appropriate, and the player must select the correct option under time pressure. Figure 1 illustrates the main visual elements of the arcade mini-game.

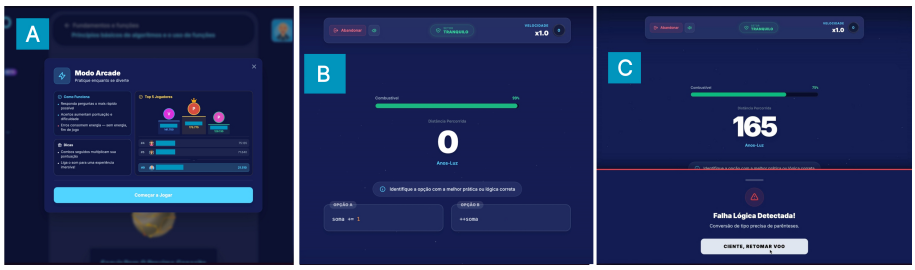

Figure 1. Visual elements of the arcade mini-game: (a) Session start; (b) Active gameplay with fuel bar and code comparison; (c) Immediate feedback after an incorrect choice.

The game loop is survival-based. The player starts each session with an energy resource called *fuel*, which decreases continuously over time. Correct answers recover part of the fuel, increase the distance traveled, and may extend a streak of correct answers called a *combo*, which temporarily increases the player's speed, as illustrated in Figure 2. Incorrect answers reduce fuel, reset the combo, and display a pedagogical explanation related to the error. Thus, the main mechanics—binary choice, time pressure, survival, distance, combo, and feedback—are intended to support visual discrimination, rapid pattern recognition, and conceptual correction.

These design choices were grounded in prior work on game-based learning and player engagement. Binary choice keeps each round focused on the intended cognitive task: comparing two code alternatives. Time pressure and survival introduce challenge and urgency, which are associated with intrinsically motivating environments [Malone 1981] and flow-oriented educational game design [Kiili 2005]. Distance and combo systems provide visible progression and reinforce consecutive successful actions, supporting clear goals, feedback, and sustained concentration [Sweetser e Wyeth 2005]. Immediate explanations after errors preserve the pedagogical function of the activity, since formative feedback is more useful when timely, specific, and connected to the learner's response [Shute 2008].

Questions are organized as parametrizable templates. Each template contains a correct code snippet, an incorrect code snippet, an explanation, a difficulty level, an estimated learning value, and a mechanic type. The mechanics considered in this study include syntax, assignment, comparison, loop, index, logic, type, indentation, scope,

and best-practice issues. Template instantiation allows superficial variation in variables, values, and contextual elements without changing the pedagogical purpose of the item.

Figure 2. Combo activation during gameplay, showing a temporary speed increase from 1.0 to 1.6 and the message *"Warming up the engines"*.

3.3. Adversarial Question Selection Architecture

The selection of the next question is modeled as an adaptive decision problem. At each instant t, the system has a set of candidate questions $\mathcal{Q}_t$ and an estimated player state x_t. Each candidate question $q \in \mathcal{Q}_t$ has attributes such as difficulty $d(q)$, associated concept $c(q)$, mechanic type $m(q)$, and estimated learning value $\ell(q)$. The player state is represented as:

$$x_t = (k_t, f_t, e_t, r_t, \delta_t, h_t), \tag{1}$$

where k_t represents estimated knowledge, f_t frustration, e_t engagement, r_t repetitiveness, δ_t recent average difficulty, and h_t the recent question history.

The proposed selector uses two internal agents with objectives in tension. Agent A prioritizes diversity, novelty, and low repetition, while Agent B prioritizes difficulty progression, adequacy, and pedagogical gain. The model is described as minimax-inspired because each agent evaluates candidate questions as if the other agent could degrade its objective, although the final decision is not a classical zero-sum minimax game. Instead, it is operationalized by adversarial utilities followed by arbitration.

For each candidate question, the system computes novelty N, repetition R, difficulty pressure P, frustration risk F, adequacy A, and expected learning gain L. The utilities of both agents are defined compactly as:

$$\begin{aligned} U_A(q, x_t) &= \lambda_1 N - \lambda_2 R - \lambda_3 P - \lambda_4 F + \lambda_5 e_t, \\ U_B(q, x_t) &= \mu_1 P + \mu_2 L + \mu_3 A - \mu_4 N - \mu_5 F. \end{aligned} \tag{2}$$

In this formulation, U_A favors novelty and penalizes repetition, excessive difficulty pressure, and frustration risk, whereas U_B favors progression, adequacy, and learning gain while penalizing excessive dispersion and frustration risk.

The final arbitration score is computed as:

$$J(q, x_t) = \omega_A U_A(q, x_t) + \omega_B U_B(q, x_t) - \gamma_F F(q, x_t) + \gamma_L L(q, x_t), \tag{3}$$

where ω_A and ω_B control the influence of each agent, while γ_F and γ_L adjust the penalty for frustration and the incentive for learning gain. The next question is selected by:

$$q_t^* = \arg \max_{q \in \mathcal{Q}_t^{feas}} J(q, x_t), \tag{4}$$

where $\mathcal{Q}_t^{feas}$ is the subset of questions that satisfies minimum feasibility constraints, such as maximum frustration risk and maximum repetition. If no candidate satisfies these constraints, the system progressively relaxes them to preserve session continuity.

Operationally, the selector receives the candidate templates restricted to the active concept, the player-state vector x_t, and the available empirical difficulty calibrations. It outputs a selected template q_t^*, instantiated as a randomized A/B code-comparison question with the correct option and explanation, while storing $U_A(q_t^*, x_t)$, $U_B(q_t^*, x_t)$, and $J(q_t^*, x_t)$ as telemetry. Candidate metrics are computed from telemetry and template metadata: adequacy compares effective difficulty with a target derived from estimated knowledge and recent difficulty; difficulty pressure captures how much the candidate exceeds recent difficulty; learning gain combines pedagogical value and adequacy; novelty, repetition, and frustration risk are derived from recent-history similarity, current frustration, and difficulty pressure. The implementation uses fixed weights, including $\omega_A = \omega_B = 0.5$, $\gamma_F = 0.3$, $\gamma_L = 0.2$, $\tau_F = 0.75$, and $\tau_R = 0.8$, and each candidate is scored once, making selection linear in the number of candidates.

3.4. Dynamic Calibration and Telemetry

The system records anonymized telemetry for each answer to support both adaptation during the session and empirical analysis after use. The telemetry schema includes a session identifier, template identifier, selected option, correctness, response time, fuel, combo, accumulated distance, mechanic type, difficulty level, Agent A and Agent B scores, and final arbitration score. These fields cover player responses, gameplay state, question metadata, and selector behavior.

Figure 3. Session summary screen displaying final distance, accuracy, and performance metrics used for telemetry and calibration

During gameplay, the player state is updated after each answer using online rules. Let $o = 1$ when the answer is correct and $o = 0$ otherwise. Estimated knowledge follows a delta update, $k_{t+1} = \text{clip}_0^1(k_t + \eta_k(o - k_t))$, with $\eta_k = 0.1$. Frustration decreases after correct answers and increases after errors, with harder questions producing larger increments. Engagement is updated toward a target of 0.6 after correct answers and 0.4 after incorrect answers, with $\eta_e = 0.2$. Recent difficulty is computed from the last five

questions, while h_t stores a FIFO window of the last ten entries. This allows subsequent selections to consider both global performance and recent behavior.

The system also performs dynamic calibration of template difficulty. Each template has a design difficulty $D_{\text{design}}(q)$, but the system estimates empirical difficulty from observed success rates. The effective difficulty is computed as:

$$D_{\text{eff}}(q, z) = 0.7 \times [4 \times (1 - \text{successRate}(q, z))] + 0.3 \times D_{\text{design}}(q), \tag{5}$$

where q is the template and z is the player's knowledge tier. The knowledge tiers are defined from k_t: beginner (0.00–0.33), intermediate (0.34–0.66), and advanced (0.67–1.00). Empirical calibration is used only when the template reaches at least five attempts in the corresponding tier; otherwise, the design difficulty remains the reference.

Dynamic calibration affects the selection process by replacing purely designed difficulty values with empirically adjusted difficulty estimates when enough interaction data are available. As a result, a template originally designed as difficult may become less likely to be treated as challenging if players consistently answer it correctly, while a template initially considered easy may receive a higher effective difficulty if it produces frequent errors. These adjusted values influence difficulty pressure, adequacy, frustration risk, and expected learning gain in subsequent selection steps. Therefore, calibration does not operate as a separate post-hoc analysis only; it can modify how future questions are scored and selected during gameplay.

3.5. Illustrative Gameplay Scenario

In a session linked to *Arrays II*, the platform restricts candidate templates to that concept and presents a code-comparison question involving array indexing. After the answer, telemetry records correctness, response time, fuel, combo, distance, mechanic type, difficulty level, and agent scores. Correct answers recover fuel, increase distance, and may extend the combo; incorrect answers reduce fuel, reset the combo, and display an explanation. The next question is then selected from the updated player state: Agent A may reduce repetition, Agent B may favor pedagogical gain or difficulty progression, and arbitration combines these tendencies with frustration risk, recent difficulty, and learning gain. This illustrates how gameplay mechanics, telemetry, adversarial selection, and calibration interact during play.

3.6. Artifact Availability and Reproducibility

To support transparency, this version provides access instructions for the playable prototype, examples of question templates, and high-level documentation of the adaptive selector: `https://cosmo.telemidia-ma.com.br/arcade`. The full telemetry dataset, analysis scripts, and complete selector configuration are not publicly released due to privacy, institutional, and maintenance constraints. To mitigate this limitation, the manuscript reports the main telemetry fields, selector variables, selection procedure, and evaluation limitations.

4. Evaluation Design

The evaluation examined the research question through telemetry collected during real use of the arcade mini-game. It considered 13 active participants, 36 valid gameplay sessions,

and 1,375 recorded answers, all analyzed in aggregate and anonymized form. The analysis focused on accuracy, response time, progression across sessions, mechanic-specific performance, difficulty distribution, combo, distance, template coverage, effective difficulty estimates, and the score behavior of the diversity and progression agents.

The evaluation was descriptive rather than comparative: it did not compare the selector with random, fixed-sequence, rule-based, or single-agent baselines, and engagement was inferred only from behavioral telemetry rather than validated player-experience questionnaires. Therefore, the results describe feasibility, practice continuity, and selector behavior, but do not establish computational superiority or perceived player experience. Future evaluations should include baselines and instruments such as GEQ, EGameFlow, or PENS [**?**, Fu et al. 2009, Ryan et al. 2006].

5. Results and Discussion

This section discusses the results obtained from the evaluation design described in Section 4. Because each arcade session is linked to the student's active curricular concept, the distribution of data was not uniform: *Fundamentals and Functions* accounted for 16 sessions, 9 participants, and 62% of all answers. The remaining data came from *Arrays II*, *IF Statement*, *Arrays I*, and *Relational Expressions*. Therefore, the findings should be interpreted primarily as evidence from introductory programming contexts.

Since there is no established benchmark for adaptive arcade-style mini-games focused on rapid code evaluation, results were interpreted through internal reference points. High accuracy, sustained answers, increasing distance and combo, and shorter response times for correct answers were considered positive indicators of feasibility and practice continuity. Inversions between designed difficulty and observed performance, imbalanced agent scores, or low template coverage were interpreted as design or calibration issues.

Table 1 summarizes the main metrics from the 36 valid sessions. Participants answered an average of 38.2 questions per session, with high variability across sessions. Mean session accuracy was 74.2%, while global answer-level accuracy was 87.3%. This difference occurs because longer sessions contributed more answers and tended to increase the aggregated accuracy.

Table 1. General summary of analyzed sessions ($n = 36$ sessions, 13 participants).

Metric	Mean	SD
Questions answered per session	38.2	39.5
Session accuracy	74.2%	24.4
Average response time	3.35 s	3.77
Distance reached	31,882	47,814
Maximum combo	16.5	19.4

Using these internal reference points, the overall results are positive for feasibility and practice continuity, but inconclusive regarding learning effectiveness. Participants answered an average of 38.2 questions per session, and global answer-level accuracy reached 87.3%, suggesting that the activity sustained repeated code-evaluation interactions under time pressure. Correct answers took 3.01 seconds on average, while incorrect answers took 5.69 seconds, indicating that errors were associated with longer

deliberation and conceptual uncertainty.

Among participants with at least two valid sessions, mean accuracy increased from 73.9% to 80.1%, mean distance from 30,270 to 44,353 units, and mean maximum combo from 14.8 to 23.6. These changes suggest progression during use, but should not be interpreted as causal learning gains because the study did not include a control group, pre- and post-tests, or external learning measures.

Table 2 presents performance by mechanic type. Syntax, indentation, and assignment errors reached accuracies above 93%, with response times for correct answers below 2.7 seconds. In contrast, loop errors, index errors, and best-practice questions showed the lowest aggregated accuracies.

Table 2. Aggregated performance by mechanic type ($n = 1,375$ answers).

Mechanic	N	Accuracy	T. correct	T. incorrect
Syntax error	350	95.4%	2.66 s	3.66 s
Indentation error	164	95.1%	2.65 s	9.07 s
Assignment error	165	93.3%	2.23 s	3.90 s
Type error	196	90.8%	3.02 s	7.79 s
Comparison error	56	85.7%	3.49 s	4.58 s
Scope error	114	83.3%	3.81 s	6.96 s
Logic error	79	75.9%	4.13 s	7.23 s
Index error	44	72.7%	4.86 s	7.54 s
Best practices	182	72.5%	3.29 s	4.03 s
Loop error	25	48.0%	5.09 s	5.92 s

The aggregated view hides differences by curricular concept. For example, index errors reached 100% accuracy in *Arrays I* but 56% in *Arrays II*, and logic errors reached 100% in *Relational Expressions* but approximately 70–73% in *Fundamentals and Functions* and *IF Statement*. Loop errors were the most critical case, but all 25 answers came from *Arrays II*, accessed by only two participants. Thus, mechanic-level results should be interpreted together with curricular concept and participant distribution.

Regarding difficulty levels, levels 2 and 3 concentrated 68.2% of all answers. The expected monotonic relationship between difficulty and performance was not observed: level 4 had the highest accuracy and lowest average response time, while level 2 had the lowest accuracy and highest response time. This inversion is a negative indicator for the current validity of the manually assigned design difficulty, but a positive indicator for the relevance of dynamic calibration, since empirical data can reveal mismatches between expected and observed difficulty. Given the concentration of answers in introductory concepts and the small sample, these values should be interpreted as indicative rather than conclusive. The adversarial selector covered all ten analyzed mechanics and used all templates available within the concepts accessed by participants, with an average of 2.6 answers per template. This indicates broad coverage and low repetition, but it also reduced the number of attempts available for empirical calibration. The analysis of agent scores revealed an imbalance: Agent A, oriented toward diversity, had a mean score of -0.31 (SD = 0.13), while Agent B, oriented toward progression, had a mean score of 16.20 (SD = 6.95). As a result, question selection was dominated by the progression agent. This is a useful diagnostic finding, but also a negative indicator for the expected balance of the dual-agent architecture. It suggests that the current configuration favors progression and pedagogical adequacy more strongly than diversity. Because the selector

was not compared with computational baselines, the study cannot determine whether it outperforms simpler strategies. Future versions should normalize utility scales, tune parameters, and conduct comparative evaluations.

Dynamic calibration operated conservatively because most templates received fewer than five attempts per knowledge tier. The few templates above the threshold were concentrated in the *ADVANCED* tier and showed 100% success rate, reducing effective difficulty to 1.20 for items originally designed as level 4. This suggests possible overestimation of difficulty for advanced users, but more interactions are needed for reliable calibration.

Overall, the results provide preliminary evidence that the adaptive arcade mini-game can sustain rapid code-evaluation practice and generate useful telemetry for pedagogical analysis. However, they are not sufficient to fully answer whether the mini-game improves rapid code-evaluation ability or whether the selector offers computational superiority over simpler strategies. The evidence is stronger for feasibility, practice continuity, diagnostic potential, and descriptive analysis of selector behavior than for learning effectiveness or algorithmic advantage.

6. Final Considerations

This paper presented an adaptive arcade mini-game for practicing rapid code evaluation in introductory programming education. The proposal combines binary-choice questions, survival mechanics, immediate feedback, telemetry, and adaptive question selection. Its main contribution is the integration of an arcade learning loop with a dual-agent selector inspired by minimax, balancing diversity, novelty, difficulty progression, and pedagogical gain.

The empirical evaluation involved 13 active participants, 36 valid sessions, and 1,375 recorded answers. Results showed global answer-level accuracy of 87.3% and an average response time of 3.35 seconds. Incorrect answers took longer than correct ones, and mean accuracy, distance, and maximum combo increased between first and last valid sessions, suggesting feasibility and progression during use. Results also showed differences among mechanic types and reinforced the need to interpret performance according to curricular concept.

The study produced design-oriented evidence about the adaptive mechanisms: the selector covered all ten analyzed mechanics and used all templates available within the accessed concepts, but the progression-oriented agent dominated arbitration and calibration was limited by the low number of attempts per tier. Therefore, the computational contribution should be understood as the design, implementation, and descriptive analysis of an adaptive selector, not as evidence of algorithmic superiority.

The main limitations are the modest number of participants, concentration of answers in *Fundamentals and Functions*, absence of a control group, lack of external learning measures, absence of player-experience questionnaires, and lack of baseline comparison. Future work should expand the template bank, refine and compare the selector, normalize utilities, tune parameters, and evaluate the mini-game with larger samples, longitudinal designs, control conditions, external learning measures, player-experience questionnaires, and computational baselines.

References

Abd El-Sattar, H. K. (2023). A new learning theory-based framework for combining flow state with game elements to promote engagement and learning in serious games. *Inf. Sci. Lett*, 12:2663–2677. Available in: https://www.naturalspublishing.com/files/published/ul41f12e0070l7.pdf.

Chou, Y.-S., Hou, H.-T., Chang, K.-E., e Su, C.-L. (2023). Designing cognitive-based game mechanisms for mobile educational games to promote cognitive thinking: an analysis of flow state and game-based learning behavioral patterns. *Interactive Learning Environments*, 31(5):3285–3302. Available in: https://www.tandfonline.com/doi/abs/10.1080/10494820.2021.1926287.

Fowler, M., Beck, K., Brant, J., Opdyke, W., e Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, Reading, MA. Available in: https://books.google.com.br/books?id=2H1_DwAAQBAJ&dq=Refactoring:+Improving+the+Design+of+Existing+Code&lr=&hl=pt-BR&source=gbs_navlinks_s.

Fu, F.-L., Su, R.-C., e Yu, S.-C. (2009). Egameflow: A scale to measure learners' enjoyment of e-learning games. *Computers Education*, 52(1):101–112. Available in: https://doi.org/10.1016/j.compedu.2008.07.004.

Gkintoni, E., Antonopoulou, H., Sortwell, A., e Halkiopoulos, C. (2025). Challenging cognitive load theory: The role of educational neuroscience and artificial intelligence in redefining learning efficacy. *Brain sciences*, 15(2):203. Available in: https://doi.org/10.3390/brainsci15020203.

Idika, S. e Saihi, S. (2025). Enhancing student engagement through ai-driven adaptive learning and gamification. *British Journal of Education*, 13(12):57–103. Available in: https://doi.org/10.37745/bje.2013/vol13n1257103.

Keuning, H., Jeuring, J., e Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Trans. Comput. Educ.*, 19(1). Available in: https://dl.acm.org/doi/10.1145/3231711.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education*, 8(1):13–24. Available in: https://doi.org/10.1016/j.iheduc.2004.12.001.

Koh, J., Cowling, M. A., Jha, M., e Sim, K. N. (2024). Design-based research as a value-added methodology for artificial intelligence in education. In *Design and Implementation of Higher Education Learners' Learning Outcomes (HELLO)*, pages 349–367. IGI Global Scientific Publishing. Available in: https://doi.org/10.4018/978-1-6684-9472-1.ch022.

Kyza, E. A. (2023). Technology-enhanced learning: A learning sciences perspective. In *Learning, design, and technology: An international compendium of theory, research, practice, and policy*, pages 221–244. Springer. Available in: https://doi.org/10.1007/978-3-319-17461-7_56.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., e Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. ITiCSE-WGR '04,

page 119–150, New York, NY, USA. Association for Computing Machinery. Available in: https://doi.org/10.1145/1044550.1041673.

Loor, M. A. M., Solorzano, D. M. A., Katherine, A., e Moreira, V. (2024). Integration of artificial intelligence in english teaching. *Journal of Cleaner Producfion*, 289:125834. Available in: https://biblioteca.ciencialatina.org/wp-content/uploads/2024/04/Integration-of-Artificial-Intelligence-in-English-Teaching.pdf.

Lozano, R., Prabhu, R. S., e Cheon, Y. (2024). Code smells for assessing and improving students' coding skills and practices. In *Innovations in Computing Research*, volume 1058 of *Lecture Notes in Networks and Systems*, pages 617–631. Springer, Cham. Available in: https://doi.org/10.1007/978-3-031-65522-7_21.

Malone, T. W. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 5(4):333–369. Available in: https://doi.org/10.1016/S0364-0213(81)80017-1.

Martel, J. S. S. e Garcias, A. P. (2024). A model of assessment co-creation in technology-enhanced learning environments in higher education. *Digital Education Review*, 45:204–213. Available in: https://eric.ed.gov/?id=EJ1434322.

Pan, Y., Shao, X., e Shakibaei, G. (2025). Influence of digital game-based learning on social collaboration, problem-solving skills, and motivation: An integrative approach of expectancy-value theory and flow theory. *Learning and Motivation*, 90:102123. Available in: https://doi.org/10.1016/j.lmot.2025.102123.

Ryan, R. M., Rigby, C. S., e Przybylski, A. (2006). The motivational pull of video games: A self-determination theory approach. *Motivation and Emotion*, 30(4):344–360. Available in: https://doi.org/10.1007/s11031-006-9051-8.

Sánchez, F. A. P., Barrera, J. P. R., Maldonado, H. O. S., e Sanabria, C. A. C. (2026). Ai-based learning platforms: A systematic review of evaluation metrics for accessibility, interactivity and adaptability through the lens of universal design for learning. Available in: https://doi.org/10.21203/rs.3.rs-8801285/v1.

Scaico, P. D., Oliveira, W., Hamari, J., MilHosseini, S., e Brambilla, A. (2024). Understanding undergraduate students' flow state in gamified and non-gamified educational systems: A qualitative case study. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE. Available in: https://doi.org/10.1109/FIE61694.2024.10892842.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, 78(1):153–189. Available in: https://doi.org/10.3102/0034654307313795.

Singletary, D. R. (2025). *Beyond the Flow: A Critical Examination of Engagement and Learning Outcomes in Game-Based Learning Through Bibliometric and Manual Literature Analysis*. Boise State University. Available in: https://www.proquest.com/openview/1b3b093d9f76839c31959d66368e1e8e.

Stidham, S. F. (2022). An integrative review of the conceptualization and assessment of the learner flow experience in the digital game-based learning environment between 2011 and 2021. Available in: https://vtechworks.lib.vt.edu/items/477e7432-b998-48d1-8179-c43acd29c5b5.

Sweetser, P. e Wyeth, P. (2005). Gameflow: A model for evaluating player enjoyment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pages 3–10. Available in: https://doi.org/10.1145/1077246.1077253.

Takács, P., Balkányi, P., e Vekety, G. B. (2023). Achieving flow-state through gamification in education. *ELTE*. Available in: https://doi.org/10.1177/0974173920150321.

Zhong, L. e Zhou, J. (2024). Personalized games for computer science education in higher education: the effect of personalization feature on students' engagement and flow state. *Journal of Computing in Higher Education*, pages 1–28. Available in: https://doi.org/10.1007/s12528-024-09427-z.