Gamification in Programming Education: A Zelda-Inspired RPG Approach for Reinforcing Foundational Programming Skills

Max Fritsche
DHSN Duale Hochschule Sachsen, Staatliche Studienakademie Leipzig, Germany
Security Robotics D&S GmbH, Leipzig, Germany

Alexander Paar
DHSN Duale Hochschule Sachsen, Staatliche Studienakademie Leipzig, Germany

Abstract

The acquisition of programming skills is a notoriously difficult process characterized by high cognitive load, complex abstraction requirements, and significant dropout rates in higher education. Traditional pedagogical approaches often struggle to bridge the gap between seemingly abstract language syntax and the strategic problem-solving required in software engineering. This paper presents a comprehensive gamified solution: a 2D top-down Role-Playing Game (RPG) inspired by classical adventure games such as *The Legend of Zelda* [30], specifically designed to reinforce existing knowledge. We detail the integration of a secure, Docker-based automated code judge within a narrative-driven environment. Key features include a virtual economy, a social "company" system for collaboration, and a curriculum focused on the C programming language to deepen low-level understanding. The paper evaluates the technical robustness of the prototype and proposes an empirical framework to validate the hypothesis that this immersive approach significantly enhances learning productivity, persistence, and student motivation.

2012 ACM Subject Classification Social and professional topics → Computer science education; Applied computing → Interactive learning environments

Keywords and phrases Gamification, C Programming, Role-Playing Games, Automated Grading, Docker

Digital Object Identifier 10.4230/OASIcs.ICPEC.2026.7

Related Version *Full Version*: `https://github.com/Velt1/whitepaper-gamification/blob/main/Bachelorarbeit_Max_Fritsche.pdf`

1 Introduction

Programming education is facing a global crisis of engagement. As software becomes the backbone of modern infrastructure, the demand for developers grows, yet introductory computer science (CS1) courses report failure rates as high as 30-50% [34, 27]. The primary challenge lies not just in learning syntax, but in developing the strategic knowledge required to decompose complex problems and debug logic under pressure. Current educational models often rely on lecture-based instruction followed by isolated, context-free exercises. These methods lack the immediate, iterative feedback loops that modern learners require to stay motivated. This paper argues that Gamification, the application of game design elements to non-game contexts, can provide a solution by fostering a "Flow" state where the challenge of coding is matched by an immersive narrative and immediate rewards.

1.1 Challenges in Programming Education

As noted by Singh [34] and Medeiros et al. [27], beginners struggle with three distinct levels of knowledge:

1. **Syntactic Knowledge:** The specific rules and grammar of the language (e.g., semicolons, brackets).
2. **Conceptual Knowledge:** Understanding abstract structures such as loops, recursion, or pointers.
3. **Strategic Knowledge:** The ability to apply concepts to solve novel problems, plan algorithms, and debug systematically.

1.1.1 Cognitive and Subject-Specific Hurdles

Students frequently struggle with the step from conceptual knowledge to strategic application. While a student might understand what a "variable" is in isolation, they often fail to integrate multiple concepts into a functioning program. Syntax errors, which lead to frustrating "trial-and-error" loops, often overshadow the actual logic of the problem. Medeiros et al. [27] emphasize that the high degree of abstraction, such as memory addresses and control structures, often lacks analogies in everyday life, making the initial learning curve exceptionally steep.

1.1.2 Behavioral and Emotional Factors

Beyond cognitive barriers, behavioral factors such as self-efficacy and time management play a crucial role. Programming is often perceived as "inherently difficult," which leads to early demotivation. In large cohorts, individual support is limited, and students often feel abandoned when facing complex software setups or cryptic compiler errors. This lack of "psychological safety" inhibits exploration and experimentation.

At the classroom level, these barriers are amplified by feedback latency. In many traditional settings, learners only discover misconceptions after delayed grading cycles, which reduces opportunities for iterative correction and weakens persistence. Empirical work on programming pedagogy therefore emphasizes rapid, formative feedback and repeated low-stakes practice as central design requirements for CS learning environments [26, 12].

1.2 Hypothesis of the Study

The hypothesis of this work is that *the gamification of programming education can be successfully and effectively implemented in the form of a Zelda-like RPG to increase learning productivity and motivation.* To evaluate this hypothesis, we developed a prototype that focuses on the reinforcement of existing skills through practical, narrative-embedded exercises.

2 Theoretical Foundations

2.1 The Importance of Practical Exercise

Programming is a craft that requires active participation. According to the constructivist principle of "Learning by Doing," knowledge is constructed through experience. Iftikhar et al. [16] state that "Learning to code is a skill that can only be developed by virtue of practice." Passive consumption of lectures or textbooks provides the vocabulary, but only the application in various scenarios builds actual competence.

Practice is particularly important for transitioning from conceptual to strategic knowledge. Students may know what loops, arrays, or pointers are, but still struggle to design complete solutions, test edge cases, and debug systematically [27]. Structured repetition across varied tasks strengthens transfer: learners recognize recurring patterns and become faster at selecting and adapting solution strategies.

In addition, repeated programming practice creates a measurable motivational effect when students can observe progress. Immediate verification of attempts, including failed attempts, supports a growth-oriented mindset and reduces the fear of making mistakes [26]. This is a key rationale behind integrating an automated judge into the game loop rather than using delayed manual assessment.

2.2 Gamification as a Didactic Method

Gamification refers to the use of game-design elements in non-game contexts to increase user engagement [36]. Unlike "Games for Learning," where the game is the primary focus, gamification adds a layer of motivation to the existing learning content.

Systematic reviews show that gamification can improve participation, persistence, and perceived enjoyment, but effects depend strongly on instructional fit [11, 8]. In programming education, the strongest benefits are reported when mechanics (e.g., quests, points, progression) are directly tied to meaningful coding actions instead of superficial reward collection [44].

2.2.1 Behaviorist Reinforcement

The core loop of a gamified system is inherently behaviorist. Based on B.F. Skinner's theories, behavior is shaped by its consequences [35]. Immediate feedback, such as points or visual rewards, serves as a positive reinforcer. Our system provides this through an Online Code Judge that evaluates code in seconds, allowing students to learn from mistakes instantly rather than waiting days for manual grading.

2.2.2 Flow Theory and Scaffolding

Optimal learning occurs in the "Flow Zone," a state between boredom and anxiety. In our RPG, scaffolding provides a ladder of increasing difficulty: early quests are simple and confidence-building, while later quests challenge algorithm design and debugging. Recent work in educational gamification confirms that balancing challenge with competence signals is essential for sustained engagement and reduced dropout [22, 3].

2.2.3 Psychological Motivation Frameworks

We apply two major frameworks to ensure deep engagement:

- **Self-Determination Theory (SDT):** SDT identifies three basic needs: *Autonomy* (choice of quests), *Competence* (mastering C code), and *Relatedness* (social interaction in companies) [33].
- **ARCS Model:** Developed by Keller, this model focuses on Attention (narrative), Relevance (real-world code), Confidence (clear goals), and Satisfaction (rewards).

2.2.4 Meaningful Gamification Design

Not every reward mechanic improves learning. If points are disconnected from mastery, students may optimize for score rather than understanding. We therefore follow a meaningful-gamification perspective: rewards are attached to quality indicators such as passing hidden tests, solving tasks with fewer retries, or completing collaborative quests [29]. This alignment helps preserve didactic intent while still using motivational affordances.

2.3 Gamification vs. Traditional Methods

Traditional methods often suffer from high latency in feedback and a "punitive" error culture. Errors are seen as failure (grade deduction) rather than a learning opportunity. In a gamified RPG context, a "Game Over" or a failed test case is an invitation to retry. Lampropoulos et al. [20] show that students in gamified courses achieved 14% higher pass rates and 17% higher average grades compared to traditional groups.

At the same time, prior research warns against overgeneralization. Effects vary by learner profile, course structure, and mechanic design; some studies report neutral outcomes when gamification is bolted on without curricular integration [8]. Our design addresses this by embedding game mechanics directly into the programming workflow itself (quest start, code submission, feedback, revision, completion).

3 State of the Art: Analysis of Existing Platforms

A review of current tools reveals specific strengths and weaknesses in the context of higher education.

3.1 Block-based Environments

Scratch is the industry standard for k-12 and avoids syntax errors through drag-and-drop blocks. While effective for teaching logic, it is often viewed as overly simplistic by university students and abstracts away low-level concerns [28, 42]. For our target cohort, this abstraction reduces friction but can limit transfer to systems-oriented programming tasks.

3.2 Puzzle and Logic Games

Games such as *Lightbot* teach algorithmic thinking through spatial puzzles. They are highly engaging but remain abstract and rarely expose learners to authentic compilation, debugging, and testing workflows. As a result, transfer from puzzle mastery to practical software tasks is not guaranteed [43].

3.3 Code Adventure Games

CodeCombat and *CheckiO* use real code (Python/JS) to control characters in a story. While successful, these platforms often situate programming tasks primarily within a game environment, which may hinder transfer to local development environments. Our approach focuses on C to provide a stronger systems-oriented challenge suitable for computer science majors [19].

3.4 Online Code Judges

LeetCode and *HackerRank* are high-performance tools for competitive programming. They offer immediate feedback but little narrative framing. For many beginners, these platforms feel like "dry" testing environments rather than motivating learning spaces. Our concept keeps judge-style rigor but embeds tasks in a persistent world with social and narrative context [14].

4 Conceptual Design of the Learning Game

4.1 Target Group and Learning Goals

The game targets second- and third-semester students who have already completed an introductory course. The goals are:

- **Strategic Thinking:** Solving complex problems through algorithm design.
- **Debugging Competence:** Systematic error identification and correction.
- **Low-Level Understanding:** Mastery of pointers, structs, and manual memory management in C.

This target profile is intentional: the game is designed for reinforcement, not first exposure. Students already know core syntax and can therefore invest cognitive capacity in decomposition, test design, and optimization. The quest sequence is structured to repeatedly exercise these higher-order skills while keeping each individual challenge bounded.

4.2 Programming Language Choice: C as Primary Track

Selecting the learning language is a strategic decision. Python lowers entry barriers and often improves early success rates in introductory settings [41]. However, for learners beyond the very first semester, C provides pedagogical advantages by exposing memory layout, pointer semantics, and explicit resource management [1].

Our curriculum therefore uses C as the default track, with optional Python support for comparative tasks in future iterations. The objective is not language preference, but depth of mental models: learners should understand why an algorithm behaves as observed at runtime, not only that it passes tests. This aligns with the project goal of reinforcing foundational systems thinking.

4.3 The Zelda Metaphor and Movement

We chose a 2D top-down pixel-art style, inspired by *The Legend of Zelda* [30]. This aesthetic is nostalgic, accessible, and reduces cognitive load compared to complex 3D environments.

- **Exploration:** Players move through a spatial world, making the learning path tangible.
- **Interactions:** NPCs act as mentors or "Quest Givers."
- **Contextualization:** Coding tasks are presented as "rebuilding broken bridges" or "deciphering ancient scrolls."

The movement layer is not cosmetic; it is the interface for didactic sequencing. Quests are spatially distributed, creating a natural progression from introductory to advanced regions. Narrative framing has been associated with higher perceived relevance and immersion in learning games [17], while nostalgic aesthetics can lower affective barriers for some learners [24].

4.4 Socio-Economic Ecosystem

The game features a meta-layer to ensure long-term persistence:

- **Virtual Economy:** Students earn "Credits" for solving quests.
- **Company System:** Students can form "Companies" (teams) to pool resources. This mimics real-world development teams and encourages peer-instruction.
- **Properties:** Companies can buy "Offices" or "Labs" in the game world, providing a sense of ownership and social status.
- **Social Feed:** A real-time activity feed shows achievements of friends, fostering a healthy sense of community.

To avoid purely extrinsic behavior, the economy is constrained by educational guardrails: rewards are tied to verified submissions, repeated low-value grinding is dampened, and collaborative achievements are prioritized over raw leaderboard chasing. This balances competition and inclusion, consistent with findings on social comparison and reward design in educational systems [37, 31].

5 Technical Implementation

5.1 System Architecture

The prototype utilizes a distributed, multi-tier architecture:

1. **Unity Client:** A standalone 2D application handling world rendering, movement, and the integrated code editor.
2. **Node.js/Express Backend:** Manages the business logic, quest state, and economy.
3. **Supabase/PostgreSQL:** Provides persistent storage with Row-Level Security (RLS) to ensure data privacy.
4. **Docker Runner Service:** An isolated environment for safe code execution.

The architecture follows established client–server patterns used in serious games [18]. Unity handles interaction and rendering, while Node.js/Express exposes REST endpoints for quests, submissions, economy operations, and social features. Real-time notifications (e.g., job completion, feed updates) are delivered via Server-Sent Events (SSE), which reduces complexity compared to full bidirectional channels for this use case [39].

5.2 Backend and Security Design

The Node.js server acts as the central coordinator. For security, we utilize JSON Web Tokens (JWT) for authentication. Every interaction, from buying an item to starting a quest, is validated against the user's role and state. The database schema includes specialized tables for `users`, `quests`, `transactions`, `companies`, and `feed_items`.

Authorization is enforced at multiple layers. API middleware validates tokens and role constraints, while Row-Level Security in PostgreSQL limits data access at the storage layer. This defense-in-depth approach reduces blast radius in case of endpoint misconfiguration. The modular backend structure also simplifies testing and maintenance as feature sets grow [4].

5.3 The Dockerized Code Runner

Executing arbitrary user code (especially in C) is a massive security risk. We implemented a "Sandbox" model:

- **Isolation:** Each run occurs in a fresh Docker container with no network access.
- **Resource Limits:** Containers are capped at 256MB RAM and 5 seconds of CPU time.
- **Redis Queue:** To handle spikes in submissions, jobs are pushed to a Redis queue and processed by a pool of "pre-warmed" containers.
- **Feedback Loop:** The runner captures `stdout`, `stderr`, and exit codes, sending them back to Unity via Server-Sent Events (SSE).

Containerization is supported by prior security research as an effective baseline for isolating untrusted workloads [5, 38]. In our runner flow, submissions are converted into queue jobs, executed with strict CPU/RAM/time limits, and then mapped to deterministic verdicts (`accepted`, `wrong answer`, `time limit exceeded`, `runtime error`). This provides both operational safety and pedagogically useful feedback granularity.

5.4 Quest Lifecycle and Data Flow

Each quest follows a consistent state machine: `available` → `started` → `submitted` → `validated` → `completed/failed`. This explicit lifecycle enables reliable progress tracking and simplifies analytics.

From a data perspective, the client requests quest metadata, submits source code, and receives asynchronous status updates. The backend stores attempts and verdicts, updates progression metrics, and emits feed events when milestones are reached. This makes it possible to audit learning trajectories (e.g., retries per quest, time-to-success) and later correlate them with learning outcomes in formal evaluation.

5.5 Unity Movement and Animation System

Technical excellence in the frontend is key for immersion.

- **Physics:** A `Rigidbody2D` based movement system ensures smooth collisions.
- **Animations:** A state-machine-based Animation Controller handles transitions between "Idle" and "Walk" for four directions.
- **Input:** We implemented the new Unity Input System to support both Keyboard and Gamepads.

6 Technical Validation

6.1 Code Judge Reliability

The judge was tested with several scenarios:

- **Infinite Loops:** A program with `while(1);` was correctly terminated by the Docker engine after 5 seconds.
- **Memory Abuse:** A program attempting to allocate 2GB of RAM was killed by the OOM-killer.
- **Syntax Parsing:** Compiler errors from `gcc` were correctly passed through to the game terminal.

In addition to these baseline checks, robust online assessment requires diverse test suites per task, including edge conditions and hidden cases. Prior online-judge research suggests that broader case coverage improves reliability and discourages overfitting to visible examples [23, 21]. Our validation strategy therefore treats test design as a first-class didactic artifact, not merely a technical detail.

6.2 Movement and Interaction Tests

We validated the interaction triggers. An NPC only opens the "Quest Menu" when the player is within a specific 2D trigger zone and presses the interaction key. This prevents overlapping UI elements and ensures a clean user experience.

We additionally verified animation transition stability (Idle/Walk), dead-zone handling for noisy input, and collision consistency across scene boundaries. These details matter because interaction friction can directly reduce task engagement and distort educational outcomes during user studies.

7 Proposed Empirical Evaluation

To confirm the pedagogical impact, we propose a longitudinal study design.

7.1 Study Design

A cohort of 60 bachelor's students will be split into an *Experimental Group* (RPG) and a *Control Group* (Traditional).

- **Period:** 6 weeks of intervention.
- **Topics:** Identical for both groups (Arrays, Pointers, Structs, Linked Lists).
- **Metrics:** Pre- and post-tests for strategic knowledge, time-on-task, and completion rates.

To improve internal validity, both groups should receive the same instructional content, assignment workload, and time budget; only the delivery format differs. Random assignment is preferred, and effect sizes should be reported in addition to significance tests. This aligns with methodological recommendations for evaluating motivational interventions in education [40].

7.2 Evaluation Dimensions and Instruments

The evaluation should combine technical, behavioral, and learning-focused measures:

- **Learning Outcomes:** Difference between pre- and post-test scores, plus transfer tasks that require applying concepts to unseen problems.
- **Engagement:** Number of completed quests, voluntary extra attempts, active time, and return frequency.
- **Motivation:** Post-intervention surveys and qualitative feedback on autonomy, competence, and enjoyment.
- **System Quality:** Failure rates, average runner latency, and usability observations.

This multidimensional instrument set is aligned with prior evaluation designs in gamified programming education [9, 15, 13].

Comparable empirical studies have reported positive effects for gamified programming interventions on both engagement and performance, especially when narrative progression and immediate feedback are combined [15, 32, 25, 6].

7.3 Didactic Importance of Quests

The quest structure provides narrative framing. Instead of "Write a function for Bubble Sort," the quest is "Organize the Kingdom's Library." This contextualization makes the problem-solving process feel meaningful. Qualitative feedback from previous studies [10] suggests that students describe such courses as "gripping" and "fesselnd."

From a didactic standpoint, quests operationalize three mechanisms simultaneously: explicit goals, constrained challenge, and immediate feedback loops. Learners know what success means, can iterate quickly, and observe progression over time. This can reduce frustration associated with abstract or decontextualized problem sheets, while preserving rigorous assessment through automated testing.

8 Limitations and Threats to Validity

Despite promising technical results, several limitations must be acknowledged. First, this paper reports a prototype and an evaluation framework, not a completed longitudinal classroom deployment. External validity therefore depends on future studies across institutions and cohorts.

Second, motivation effects can be novelty-driven. Students may initially engage more due to the game format, with attenuation over time. Long-term retention measurements and follow-up testing are required to distinguish temporary engagement spikes from durable learning gains [2, 7].

Third, there are potential selection and implementation biases. Learners already interested in games may respond differently from those preferring conventional study formats. Instructor guidance quality and quest calibration can further moderate outcomes. To mitigate these risks, future studies should pre-register hypotheses, document implementation fidelity, and include subgroup analyses.

Finally, infrastructure assumptions (stable internet access, compatible devices, and backend capacity) may influence adoption in school contexts. Technical robustness and pedagogical value must therefore be considered jointly when transferring the approach from university pilots to broader educational settings.

9 Conclusion and Future Work

9.1 Summary of Results

The "CodeQuest RPG" prototype demonstrates that gamification can be successfully integrated into the rigorous context of university programming education. Technically, we have shown that Docker-based code execution is safe and responsive. Pedagogically, the combination of narrative exploration and immediate feedback addresses the primary pain points of beginners: demotivation and the struggle with abstraction.

9.2 Outlook and School Transfer

For future development, we envision:

- **Multi-Language Support:** Expanding the runner to support Python and Rust.
- **Collaborative Dungeons:** Multi-player puzzles where code from two players must be synchronized to succeed.
- **School Integration:** Adapting the narrative for the German school system (Oberstufe) to promote "Informatische Grundbildung" in a playful yet serious way.

References

1 D. G. Balreira, Thiago L. T. Silveira, and Juliano Araujo Wickboldt. Investigating the impact of adopting python and c languages for introductory engineering programming courses. *Computer Applications in Engineering Education*, 31:47–62, 2022. doi:10.1002/cae.22570.

2 P. Buckley and Elaine Doyle. Gamification and student motivation. *Interactive Learning Environments*, 24(6):1162–1175, 2016. doi:10.1080/10494820.2014.964263.

3 G. Challco, I. Bittencourt, M. Reis, Jário Santos, and Seiji Isotani. Gamiflow: Towards a flow theory-based gamification framework for learning scenarios. In *Artificial Intelligence in Education*, pages 415–421, 2023. doi:10.1007/978-3-031-36336-8_65.

4 O. Chaplia and H. Klym. An approach for automated code deployment between multiple node.js microservices. In *2023 IEEE 13th International Conference on Electronics and Information Technologies (ELIT)*, pages 202–205, 2023. doi:10.1109/ELIT61488.2023.10310996.

5 Théo Combe, Antony Martin, and Roberto Di Pietro. To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5):54–62, 2016. doi:10.1109/MCC.2016.100.

6 J. Costa. Using game concepts to improve programming learning: A multi-level meta-analysis. *Computer Applications in Engineering Education*, 31:1098–1110, 2023. doi:10.1002/cae.22630.

7 Karen D. Cuervo-Cely, Felipe Restrepo-Calle, and J. J. Ramírez-Echeverry. Effect of gamification on the motivation of computer programming students. *Journal of Information Technology Education: Research*, 21:1–23, 2022. doi:10.28945/4917.

8 Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. Gamification in education: A systematic mapping study. *Educational Technology & Society*, 18(3):75–88, 2015. URL: https://www.researchgate.net/publication/270273830_Gamification_in_Education_A_Systematic_Mapping_Study.

9 José Figueiredo and Francisco J. García-Peñalvo. Increasing student motivation in computer programming with gamification. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 997–1000, 2020. doi:10.1109/EDUCON45650.2020.9125283.

10 Nuno H. Flores and Rui Pinto. Quest-based gamification in a software development lab course: A case study. In *9th International Conference on Higher Education Advances (HEAd'23)*, 2023. doi:10.4995/HEAd23.2023.16110.

11 Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work? a literature review of empirical studies on gamification. In *2014 47th Hawaii International Conference on System Sciences*, pages 3025–3034. IEEE, 2014. doi:10.1109/HICSS.2014.377.

12 John Hattie and Helen Timperley. The power of feedback. *Review of Educational Research*, 77(1):81–112, 2007. doi:10.3102/003465430298487.

13 C. J. Hellín, Francisco Calles-Esteban, Adrián Valledor, Josefa Gómez, Salvador Otón-Tortosa, and A. Tayebi. Enhancing student motivation and engagement through a gamified learning environment. *Sustainability*, 2023. doi:10.3390/su151914119.

14 Zhang Hua, Miao Zhang, Fanchao Meng, Xuequan Zhou, and Dianhui Chu. Application of the online judge technology in programming experimental teaching. In *Proceedings of the 2020 5th International Conference on Modern Management and Education Technology (MMET 2020)*, 2020. doi:10.2991/assehr.k.201023.059.

15 M. B. Ibáñez, Angela Di-Serio, and Carlos Delgado-Kloos. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on Learning Technologies*, 7(3):291–301, 2014. doi:10.1109/TLT.2014.2329293.

16 Sidra Iftikhar, Ana-Elena Guerrero-Roldán, and Enric Mor. Practice promotes learning: Analyzing students' acceptance of a learning-by-doing online programming learning tool. *Applied Sciences*, 12(24):12613, 2022. doi:10.3390/app122412613.

17 H. Jarrah, Doha Adel Bilal, Mona Halim, M. Helali, R. AlAli, Ali Atwa Ali Alfandi, and M. Khasawneh. The impact of storytelling and narrative variables on skill acquisition in gamified learning. *International Journal of Data and Network Science*, 2024. doi:10.5267/j.ijdns.2023.11.018.

18 Abdul Malik Khan, Ivan Arsov, Marius Preda, Samir Chabridon, and Anne-Marie Beugnard. Adaptable client-server architecture for mobile multiplayer games. In *SIMUTools*, page 11, 2010. doi:10.4108/ICST.SIMUTOOLS2010.8704.

19 Chrysoula Kroustalli and Stelios Xinogalos. Studying the effects of teaching programming to lower secondary school students with a serious game: A case study with python and codecombat. *Education and Information Technologies*, 26, 2021. doi:10.1007/s10639-021-10596-y.

20 Georgios Lampropoulos and Antonis Sidiropoulos. Impact of gamification on students' learning outcomes and academic performance: A longitudinal study comparing online, traditional, and gamified learning. *Education Sciences*, 14(4):367, 2024. doi:10.3390/educsci14040367.

21 Rodziah Latih, Marini Abu Bakar, Norleyza Jailani, N. M. Ali, Syahanim Mohd Salleh, and A. Zin. Pc2 to support instant feedback and good programming practice. In *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 1–5, 2017. doi:10.1109/ICEEI.2017.8312410.

22 Lei Li and Qiang Sun. Research on gamification learning model based on the flow theory. In *2023 5th International Conference on Computer Science and Technologies in Education (CSTE)*, pages 93–96, 2023. doi:10.1109/CSTE59648.2023.00023.

23 Zheng Li, Deli Yu, Yonghao Wu, and Yong Liu. Using fine-grained test cases for improving novice program fault localization. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 178–183, 2020. doi:10.1109/COMPSAC48688.2020.00032.

24 Péter Makai. Video games as objects and vehicles of nostalgia. *Humanities*, 7(4):123, 2018. doi:10.3390/h7040123.

25 Beatriz Marín, Jonathan Frez, J. A. Cruz-Lemus, and Manuel Genero. An empirical investigation on the benefits of gamification in programming courses. *ACM Transactions on Computing Education*, 19(1):1–22, 2018. doi:10.1145/3231709.

26 S. Marwan, Bita Akram, Tiffany Barnes, and Thomas Price. Adaptive immediate feedback for block-based programming: Design and evaluation. *IEEE Transactions on Learning Technologies*, 15:406–420, 2022. doi:10.1109/TLT.2022.3180984.

27 Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90, 2019. doi:10.1109/TE.2018.2864133.

28 Hugo Montiel and Marcela Georgina Gómez Zermeño. Educational challenges for computational thinking in k-12 education: A systematic literature review of scratch as an innovative programming tool. *Computers*, 10(6):69, 2021. doi:10.3390/computers10060069.

29 Scott Nicholson. A user-centered theoretical framework for meaningful gamification. In *Games+Learning+Society 8.0*, 2013. doi:10.9776/13222.

30 Nintendo. The legend of zelda. https://www.zelda.com/, 1986. Accessed 2026-04-11.

31 Sungjin Park and Sangkyun Kim. Leaderboard design principles to enhance learning and motivation in a gamified educational environment: Development study. *JMIR Serious Games*, 9(1), 2021. doi:10.2196/14746.

32 Arturo Rojas-López, Elvira G. Rincón-Flores, Juan Mena, Francisco J. García-Peñalvo, and María Soledad Ramírez-Montoya. Engagement in the course of programming in higher education through the use of gamification. *Universal Access in the Information Society*, 18:583–597, 2019. doi:10.1007/s10209-019-00680-z.

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

34 Sarita Singh. Identifying learning challenges faced by novice/beginner computer programming students: An action research approach. In *QuASoQ/SEED@APSEC*, 2022. URL: https://api.semanticscholar.org/CorpusID:255968169.

35 Burrhus Frederic Skinner. Teaching machines. *Science*, 128(3330):969–977, 1958. doi:10.1126/science.128.3330.969.

36 Rodrigo Smiderle, Sandro J. Rigo, Liane B. Marques, Emanuelle H. T. de Oliveira, and Patrícia A. Jaques. The impact of gamification on students' learning, engagement and behavior based on their personality traits. *Smart Learning Environments*, 7(1):3, 2020. doi:10.1186/s40561-019-0098-x.

37 Sergey Sosnovsky, Qixiang Fang, Bert de Vries, Stefan Luehof, and Frans Wiegant. Towards adaptive social comparison for education. In *Intelligent Tutoring Systems*, pages 421–426, 2020. doi:10.1007/978-3-030-57717-9_38.

38 F. Špaček, Radomír Sohlich, and Tomáš Dulík. Docker as platform for assignments evaluation. *Procedia Engineering*, 100:1665–1671, 2015. doi:10.1016/j.proeng.2015.01.541.

39 L. D. L. Torre, J. Chacón, D. Chaos, S. Dormido, and José Sánchez. Using server-sent events for event-based control in networked control systems. *IFAC-PapersOnLine*, 2019. doi:10.1016/j.ifacol.2019.08.218.

40 Horst Treiblmaier and Lisa Putz. Gamification as a moderator for the impact of intrinsic motivation: Findings from a multigroup field experiment. *Learning and Motivation*, 71:101655, 2020. doi:10.1016/j.lmot.2020.101655.

41 Jacques Wainer and Eduardo C. Xavier. A controlled experiment on python vs c for an introductory programming course. *ACM Transactions on Computing Education*, 18(3):1–16, 2018. doi:10.1145/3152894.

42 Fu-Hsiang Wen, Tienhua Wu, and W. Hsu. Toward improving student motivation and performance in introductory programming learning by scratch: The role of achievement emotions. *Science Progress*, 106, 2023. doi:10.1177/00368504231205985.

43 Mirac Yallihep and Birgul Kutlu. Mobile serious games: Effects on students' understanding of programming concepts and attitudes towards information technology. *Education and Information Technologies*, 25:1237–1254, 2019. doi:10.1007/s10639-019-10008-2.

44 Zehui Zhan, Luyao He, Yao Tong, Xinya Liang, Shihao Guo, and Xixin Lan. The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, 3:100096, 2022. doi:10.1016/j.caeai.2022.100096.