One Click, One Game: AI-Generated Personalized Game-Based Learning Experiences

Anonymous Author(s)[1]

[1]*Anonymous Institution*

Abstract

Personalized game-based learning holds significant educational promise, but creating custom games for individual learners remains prohibitively expensive. This paper presents Play My Way (PMW), a multi-agent system that autonomously generates personalized educational mini-games without human intervention. The architecture coordinates specialized AI components (a conceptualization LLM, a code generation LLM, and a graphics module) through a placeholder-based orchestration mechanism that enables modular collaboration.

The system contributes two primary technical advances. First, it demonstrates autonomous multi-agent coordination for educational content generation, where specialized components collaborate to produce complete, playable games from a learner's hobby and educational topic. Second, novel graphics generation techniques address consistency in game assets: reference-based character generation for visual coherence, skeleton-guided animation for fluid motion across frames, and iterative inpainting for continuous background expansion.

A preliminary feasibility study with 15 participants generated 133 playable games at $0.11 per game, confirming technical viability and establishing baseline quality metrics that identify clear directions for improvement.

Keywords

Multi-Agent Systems, Generative AI, Game-Based Learning, Procedural Content Generation, Personalized Learning

1. Introduction

Educational games can improve learning outcomes, but most remain generic. Thematic personalization, aligning game content with individual learners' interests, fosters deeper engagement [1]. Yet it faces a fundamental barrier: creating custom games for each learner requires prohibitive development resources [2].

Generative AI provides the building blocks. Recent advances enable AI to generate both code and visual assets, but current approaches still require constant human oversight [3]. True scalability demands systems where AI components collaborate autonomously to produce finished games without human intervention.

This research addresses a core question: **To what extent can a generative AI-powered system autonomously create interest-aligned mini-games for game-based learning?**

Play My Way (PMW) contributes: (1) a *modular multi-agent architecture* where specialized components for conceptualization, code generation, and graphics creation collaborate through placeholder-based orchestration; and (2) *graphics generation techniques* for consistent game assets, including reference-based character generation, skeleton-guided animation, and iterative background expansion.

This work is presented as work in progress. An exploratory pilot with 15 participants generating 133 games demonstrates technical viability and surfaces where fully autonomous generation breaks down. We treat these failure modes as a map of candidate *agency-handoff points* among learner, teacher, and AI agents, an angle we develop in the discussion.

HAI-Agency: Workshop on Orchestrating Human and AI Agency for Proactive and Reflective Learning, co-located with the 27th International Conference on Artificial Intelligence in Education (AIED 2026), Seoul, Korea, June 27 – July 3, 2026.

2. Related Work

Research consistently shows game-based learning improves outcomes. Meta -analyses find positive effects on learning and retention compared to conventional instruction [4], and games that adapt to learner preferences foster greater intrinsic motivation than one-size-fits-all approaches [5]. Yet most adaptive educational games adjust only difficulty or pacing, not thematic content [6]. Deeper personalization, aligning entire games with learner interests, remains rare because it requires pre-authored content from multidisciplinary teams operating within constrained budgets [7, 2, 8]. Scaling personalized game-based learning requires automating game creation itself.

While Procedural Content Generation has traditionally automated content *within* games [9], generative AI now enables automating game creation. LLM accuracy on general code generation benchmarks has risen from 3.6% for early models to 95.1% for recent systems [10], with growing integration into game development research [11]. For visual assets, latent diffusion models generate images from text prompts [12], enabling rapid creation of characters, environments, and objects. However, maintaining visual consistency across multiple generated assets remains under-explored, despite being essential for cohesive game aesthetics. The components for automated game creation exist, but integrating them is challenging.

These capabilities degrade on domain-specific tasks. Top-performing code LLMs succeed only 45% of the time on game-specific logic [13], and LLM-based generation focuses on level design for existing titles rather than complete game synthesis [14]. The dominant paradigm remains *Human-In-The-Loop (HITL)*: developers guide the AI, evaluate outputs, debug errors, and integrate components [3]. Single-model approaches have not achieved the autonomy that scalable personalization requires.

Multi-agent systems offer a path forward. CREA outperforms single-model baselines for visual art by coordinating planner, generator, and critic agents [15]; MegaAgent shows software-engineering agents can autonomously decompose and parallelize tasks without predefined procedures [16]. Both illustrate how distributing responsibilities across specialized components enables higher automation [17, 18]. Yet research on LLM-based game agents focuses on agents that *play* games, not agents that *create* them [19, 20].

What remains missing are systems where specialized AI components collaborate autonomously to produce complete educational games. PMW addresses this gap through a modular architecture where specialized components coordinate via placeholder-based communication rather than sequential processing with human checkpoints.

3. System Design

Most adaptive learning systems personalize by adjusting difficulty or pacing. PMW takes a different approach: it adapts entire games to learners' interests, generating unique experiences from gameplay mechanics to visual themes. Learners select their hobbies through a simple interface (Fig. 1a) and receive personalized mini-games within minutes. A learner passionate about archery, for instance, receives a bow-and-arrow targeting game themed around their interest (Fig. 1b). These hobby-aligned games embed curriculum-aligned multiple-choice questions from a question database, reducing the psychological distance between learner and material.

This personalization requires generating complete games autonomously. A playable game needs code, graphics, and educational content working together. Coordinating these components is the core technical challenge, which PMW addresses through two design principles, modular specialization and placeholder-based communication, structured into a two-phase architecture (Fig. 2): Creative Generation produces game concepts and executable code with placeholders, and Asset Generation fulfills these references through graphics generation and question retrieval (detailed in Sections 4 and 5).

(a) Hobby selection interface

(b) Generated archery-themed game

Figure 1: PMW interface and example output: users select hobbies and topics, receiving personalized mini-games within minutes.

3.1. Modular Specialization

Rather than relying on a single model for all aspects of game creation, PMW delegates distinct responsibilities to specialized components (Fig. 2). The Game Idea LLM conceptualizes game designs based on the learner's hobby; the Code LLM translates these concepts into executable browser-based games; the Graphics Module produces visual assets tailored to the game's theme. This separation mirrors how human game studios organize expertise across designers, programmers, and artists. It also enables targeted improvements: upgrading one component requires no changes to others.

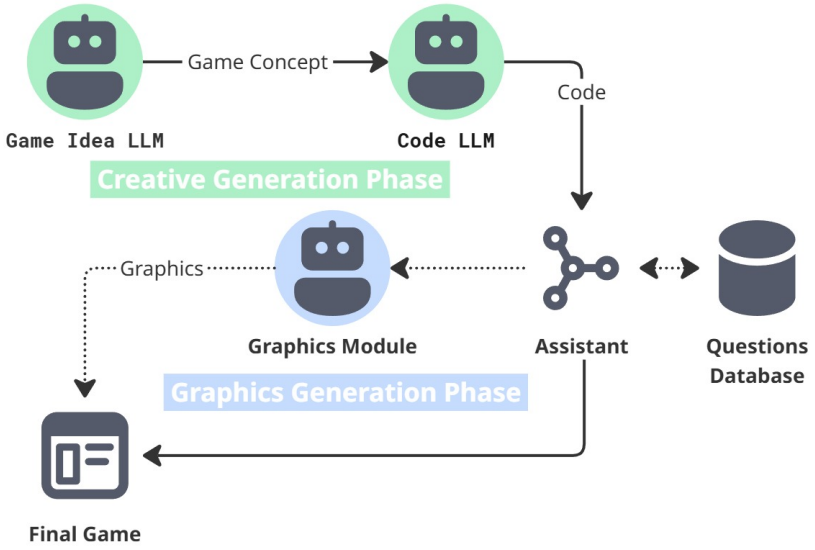

Figure 2: PMW's two-phase architecture: Creative Generation (green) produces game concepts and code with placeholders; Asset Generation (blue) fulfills placeholder references through the Graphics Module and Questions Database. The Assistant orchestrates asset fulfillment by coordinating between modules.

3.2. Placeholder-Based Communication

Components operate independently through a declarative interface. The Code LLM produces functional games with placeholders: structural references indicating where assets from other modules should appear. The Assistant orchestration layer scans for these references, routes requests to appropriate modules, and assembles the final product. This decoupling allows each component to focus on its specialty without tight integration dependencies.

4. Creative Generation Phase

The Creative Generation Phase transforms a learner's hobby into a game concept and executable code, coordinating the Game Idea LLM and Code LLM. This pipeline can fail when LLMs generate plausible-sounding but broken outputs, what researchers call hallucinations [21]. Structured prompt engineering addresses this challenge.

4.1. Prompt Engineering Framework

PMW structures prompts using the RACE framework [22], which organizes instructions into four components: **Role** defines the model's identity; **Action** specifies the task; **Context** provides constraints and boundaries; **Example** anchors generation with a concrete reference. This structure confines creative scope to implementable outputs. The trade-off is reduced creative diversity: generated games tend to adopt mechanics similar to provided examples. This predictability helps the Code LLM reliably parse and implement the concepts.

4.2. Game Concept Generation

The Game Idea LLM (GPT-4o [23]) assumes the role of professional game designer under specific constraints. Prompts guide toward short, replayable browser games integrating multiple-choice questions, preventing overambitious concepts. Figure 3 shows example output for the hobby "basketball": a structured concept that specifies mechanics, visual theme, and educational integration points.

```
{
    "concept": "Basketball shooting with quiz integration",
    "controls": "WASD movement, mouse-click to shoot",
    "objective": "Score baskets to trigger quiz questions",
    "education_integration": {
      "trigger": "Every 2 successful baskets",
      "topic": "history",
      "questions": 50
    },
    "scoring": {"correct": "2x multiplier", "incorrect": "-50 points"},
    "visual_style": "2D pixel art, player in jersey, court background"
  }
```

Figure 3: Structured game concept from the Game Idea LLM for hobby "basketball."

4.3. Code Generation

The Code LLM translates game concepts into browser-based games. The target format is self-contained HTML/CSS/JavaScript, requiring no external libraries or server-side dependencies. This ensures generated games run immediately in any modern browser without installation. For this component, o3-mini [24] was selected as it achieves state-of-the-art results on SWE-bench Verified, an industry-standard code generation benchmark [25].

The Code LLM cannot generate images or curate educational questions directly. Instead, it expresses these needs through placeholder functions: a call such as `getGraphic("basketball court with hoops")` requests an image from the Graphics Module, while `getQuestions("history", 50)` requests quiz items from the Questions Database. The Assistant orchestration layer parses the generated code, identifies these placeholders, routes each request to the appropriate module, and substitutes the returned assets back into the code, producing a self-contained game. This separation allows the Code LLM to focus entirely on game logic and structure.

4.4. Constraint Evolution

Constraints are explicit rules embedded in prompts that limit LLM generation, prohibiting error-prone patterns and enforcing compatibility with downstream components. They evolve through development-time refinement: when multiple generations exhibited the same error, explicit prohibitions were added to prevent recurrence, accumulating coverage of common failure modes without model fine-tuning.

A second mechanism incorporates individual learner preferences. Learners can update preference profiles at any time, indicating preferred mechanics, pacing, or visual styles; these append to prompts for subsequent generations, refining outputs per learner over time. Together the two mechanisms enable improvement both globally and individually without retraining underlying models.

5. Graphics Generation Phase

Educational games require visuals that bring concepts to life. A soccer-themed quiz needs fields, goals, and animated players. Yet generating these assets consistently and automatically poses a challenge, particularly within browser-based 2D constraints.

PMW adopts pixel art for its well-defined constraints and suitability for rapid iteration. General-purpose models like DALL-E [26] struggle with pixel art, which demands precise control over discrete grids and transparency that these models cannot reliably provide. PixelLab [27] addresses these limitations as a purpose-built solution offering direct control over sprite-specific features. The following subsections describe three workflows for generating consistent game assets.

5.1. Reference-Based Character Consistency

Without structural guides, AI-generated characters exhibit inconsistent proportions across frames, making them unsuitable for animation. PMW addresses this through reference-based generation that combines a predefined reference image with a text prompt. Figure 4 shows how a single reference pose combined with different prompts creates visually consistent characters. The reference constrains structure; the prompt dictates theme. This produces animation-ready sprites with predictable skeletal structure and ensures visual coherence across all characters within a game.

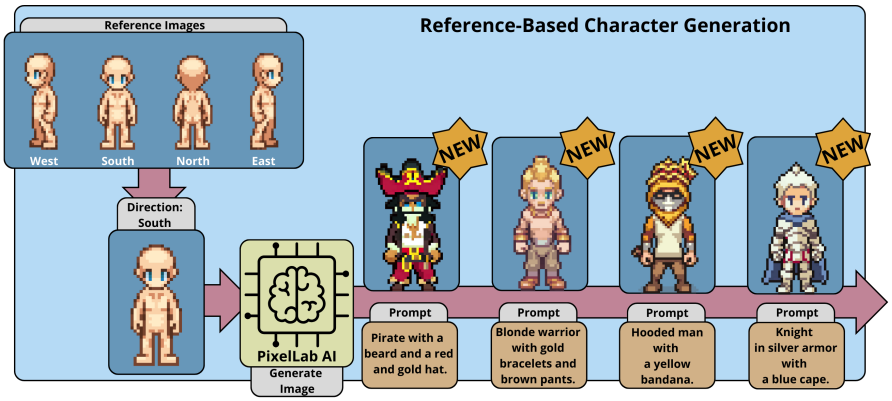

Figure 4: Reference-based generation: a single reference pose combined with text prompts creates consistent characters suitable for animation.

5.2. Skeleton-Based Animation

Static characters are insufficient for most mechanics; walking, jumping, and swimming all require animation. AI generates each frame independently, causing character appearance to shift unpredictably between outputs, breaking the visual identity that animation requires.

PMW employs a two-phase skeleton-guided animation process (Fig. 5). The first phase defines reference animations: a walk cycle is manually designed for a standard character, and skeletal postures

are extracted for each keyframe and stored in an animation library. These abstract poses define the animation independent of character appearance.

The second phase generates a unique character for each learner using the reference-based techniques. The character's skeletal posture is then estimated and combined with the stored animation poses to deform the character into each keyframe, producing the final animation. Since frames are generated independently, they can be processed in parallel, reducing total generation time.

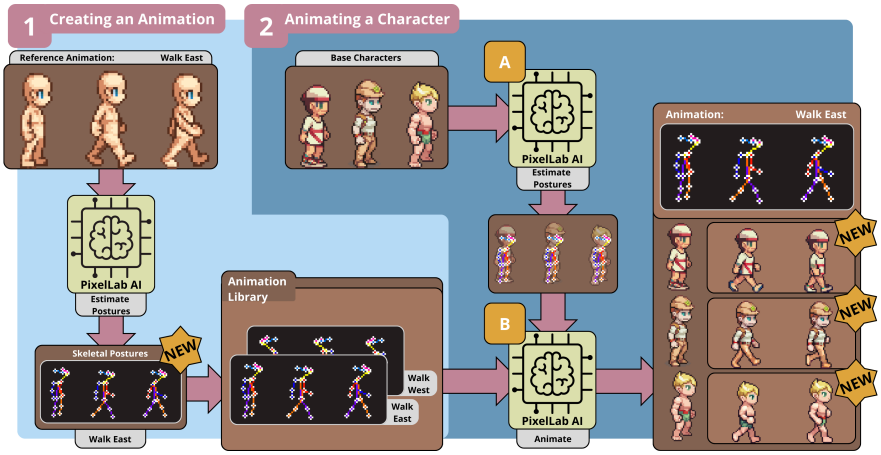

Figure 5: Two-phase skeletal animation: Phase 1 (left) creates reusable animations by storing skeletal poses from a reference. Phase 2 (right) estimates a new character's posture (A) and combines it with stored poses to generate animated frames (B).

5.3. Background Expansion via Inpainting

Side-scrollers and endless runners demand backgrounds extending beyond typical image bounds. PixelLab's maximum resolution of 400×400 pixels makes single-step generation of wide scrolling backgrounds impossible.

PMW uses iterative inpainting [28] to expand backgrounds without visible joins (Fig. 6). A background image is generated to serve as the base. The expansion process then copies a strip of pixels from this image's edge to a new canvas, masks the copied region for preservation, and synthesizes new content in the adjacent blank area. Each newly generated segment becomes the reference for the next expansion, repeating until reaching desired dimensions. This produces continuous environments like racing tracks or soccer fields that extend beyond single-image generation limits.

6. Pilot Study

The pilot is exploratory rather than confirmatory: it is a first attempt to characterize *where* fully autonomous generation falls short and what those failure modes imply for human-in-the-loop design. We use the GUESS-18 subscales as a diagnostic instrument, treating low scores as evidence pointing at specific generator weaknesses rather than as a pass/fail verdict. Fifteen participants, recruited through online platforms, used the system for seven consecutive days, generating mini-games based on their hobbies. Ethics approval was granted on November 14, 2024.

6.1. Methodology

After each game, participants provided immediate 1–5 quality ratings. Every two days, they completed a questionnaire adapted from GUESS-18 [29], an 18-item instrument assessing game user experience on a 7-point Likert scale. Two subscales, Social Connectivity and Audio Aesthetics, were excluded as irrelevant to single-player games without audio; additional items assessed perceived educational

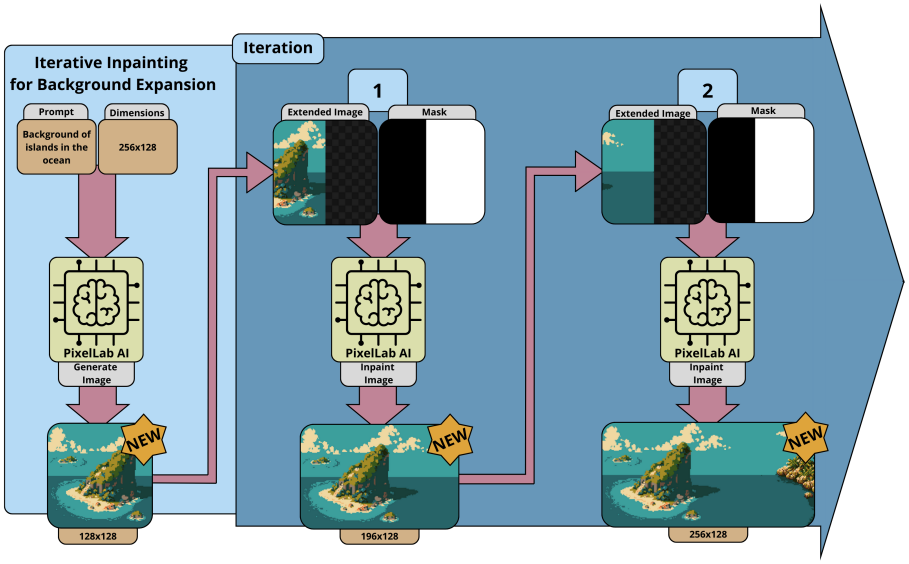

Figure 6: A background is iteratively expanded using inpainting. Each generated segment becomes the reference for the next, producing continuous wide environments.

value and personalization alignment. All interactions were logged automatically, capturing generation frequency, session duration, ratings, and technical outcomes.

6.2. Results

The system generated 133 mini-games across 15 participants at an average cost of $0.11 per game, demonstrating economic scalability. Generation averaged 4.7 minutes and ranged from 39 seconds to 11.8 minutes.

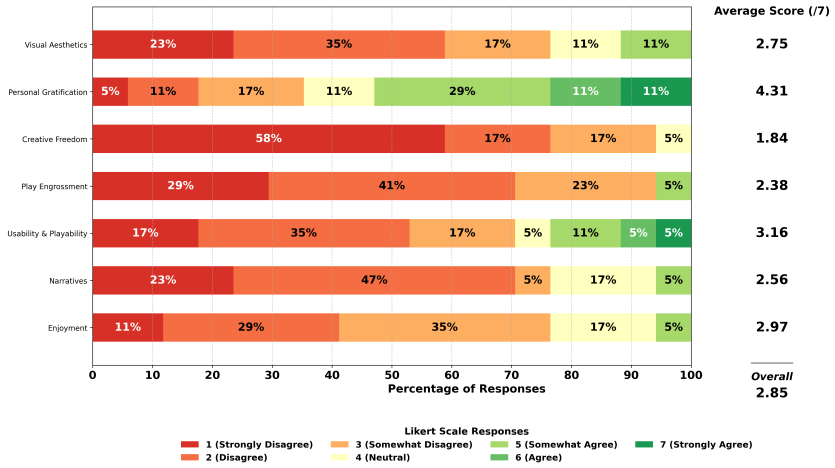

Figure 7: GUESS-18 results across seven dimensions. No subscale achieved the moderate quality threshold of 5/7. Personal Gratification scored highest (4.31/7); Creative Freedom scored lowest (1.84/7).

No GUESS-18 subscale reached the 5/7 moderate threshold (Fig. 7). Each dimension, read as a diagnostic, points at a distinct weakness in the autonomous pipeline. Personal Gratification scored highest (4.31/7), suggesting personalized themes do provide some intrinsic reward. Usability & Playability (3.16/7) and Enjoyment (2.97/7) indicate games were accessible but insufficiently polished. Visual Aesthetics (2.75/7), Narratives (2.56/7), and Play Engrossment (2.38/7) point at limitations in graphical coherence, storytelling depth, and player immersion. Creative Freedom, lowest at 1.84/7, signals that mechanics felt repetitive, a direct consequence of example-anchored prompting (§4.2).

Engagement declined over the study period: seven of fifteen participants had stopped generating games by Day 4, suggesting novelty attracted initial interest but quality did not sustain participation. The system logged a 15.2% failure rate, 20 of 133 games could not be accessed by participants. More than half of participants reported technical issues in games that loaded successfully. These included spatial logic errors, where game mechanics assumed wrong positions for elements; visual mismatches, where code expected blank backgrounds but received pre-filled ones; and UI scaling problems that rendered interfaces unusable.

7. Discussion

The pilot confirms technical feasibility: the system autonomously produced 133 playable, hobby-aligned games at $0.11 per game. It also makes visible *where the autonomy boundary fails*. Quality fell short across every GUESS-18 dimension, but the value of those numbers is not the verdict; it is the map they sketch of the breakdowns a fully autonomous pipeline cannot resolve on its own. We use the rest of this section to read those breakdowns (§7.1) and translate them into proposals for where humans should re-enter the loop (§7.2).

7.1. Why the Games Fell Short

Four failure modes recurred across the 133 generated games.

First, LLMs cannot reliably reason about screen coordinates. Code frequently assumed obstacles would spawn from one edge while graphics placed them elsewhere, or positioned interactive elements where players could not reach them. These spatial reasoning failures made games frustrating or unplayable.

Second, the Code LLM operates blind to generated graphics, writing code that assumes certain visual layouts without seeing what the Graphics Module produces. This caused mismatches: for instance, backgrounds already containing elements would receive duplicate sprites from code expecting blank canvases.

Third, example-driven generation caused games to inherit the quiz-interruption structure from the provided example unless participants explicitly overrode it in their preferences, breaking gameplay flow.

Last, the pipeline produced complete games in one pass, prioritizing speed over depth. This constrained polish and emergent mechanics, yielding functional but shallow prototypes.

7.2. The Path Forward: Where Should the Human Sit?

Taken together, the four failure modes argue against the framing that motivated this work. Full autonomy is not the right destination; the more productive question is *where* in the pipeline a human (learner, teacher, or reviewing AI agent) adds the most value relative to the cost of their attention. Each failure mode in §7.1 suggests a different placement.

Spatial reasoning and code↔graphics grounding are tractable without humans: a reviewing agent that can preview assets before code finalizes, or iterative refinement loops, are likely sufficient and preserve the cost structure that makes the system interesting in the first place. **Mechanic repetition** (Creative Freedom 1.84/7) is harder. Example-anchored prompting trades diversity for code reliability, and breaking that trade-off probably needs a teacher in the loop curating or extending the example pool per learning context. **Educational depth** (the shallow integration of quizzes over rather than within mechanics) is the placement we are most confident about: this is where teacher review of generated concepts before they reach learners pays off most, because pedagogical fit is the judgment AI is currently worst at making and humans are best at. Finally, **learner agency** over preference profiles is already present but underused; making it more granular and visible is a low-cost way to absorb the personalization-alignment failures the pilot surfaced. Across these four placements the pattern is consistent: cheap, high-volume, low-stakes decisions stay with the AI; rare, high-stakes,

judgment-heavy decisions move to teachers or learners. The pilot is a first attempt to map which decisions fall on which side of that line.

7.3. Limitations

The evaluation has notable limitations: small sample (N=15), high attrition, and one-week duration. The low per-game cost also raises ethical considerations: systems prioritizing quantity over quality risk producing superficially personalized content lacking educational depth. Despite these constraints, if quality and reliability improve, AI-driven personalization could democratize access to customized learning experiences.

8. Conclusion

This paper presented Play My Way, a multi-agent architecture that autonomously generates personalized educational mini-games by coordinating a Game Idea LLM, a Code LLM, and a Graphics Module through placeholder-based orchestration. Alongside the architecture, the work contributes graphics generation techniques (reference-based character consistency, skeleton-guided animation, iterative inpainting) that address visual coherence in automated game asset creation.

An exploratory pilot confirmed that the pipeline runs end-to-end and economically, and mapped the failure modes that fully autonomous generation cannot resolve on its own. Each failure mode points at a candidate human-in-the-loop checkpoint among learner, teacher, and reviewing AI agents. Whether AI can act alone is no longer the operative question for scalable personalized game-based learning; *where* agency sits is.

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude in order to: Improve writing style; Generate code. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] R. M. Ryan, E. L. Deci, Self-Determination Theory: Basic Psychological Needs in Motivation, Development, and Wellness, Guilford Publications, 2018.

[2] F. Mehm, J. Konert, S. Göbel, R. Steinmetz, An Authoring Tool for Adaptive Digital Educational Games, in: 21st Century Learning for 21st Century Skills, volume 7563, Springer Berlin Heidelberg, 2012, pp. 236–249. doi:10.1007/978-3-642-33263-0_19.

[3] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, L. He, A survey of human-in-the-loop for machine learning, Future Generation Computer Systems 135 (2022) 364–381. doi:10.1016/j.future.2022.05.014.

[4] P. Wouters, C. van Nimwegen, H. van Oostendorp, E. D. van der Spek, A meta-analysis of the cognitive and motivational effects of serious games, Journal of Educational Psychology 105 (2013) 249–265. doi:10.1037/a0031311.

[5] Y. Xiao, K. Hew, Personalized gamification versus one-size-fits-all gamification in fully online learning: Effects on student motivational, behavioral and cognitive outcomes, Learning and Individual Differences 113 (2024). doi:10.1016/j.lindif.2024.102470.

[6] A. García-Alonso, C. Casado-Lumbreras, S. Sastre-Merino, School engagement in the gamified digital environment: A systematic review and future research agenda, Frontiers in Psychology 13 (2022) 844995. doi:10.3389/fpsyg.2022.844995.

[7] B. R. Maxim, B. Ridgway, Use of interdisciplinary teams in game development, in: 2007 37th Annual Frontiers In Education Conference, 2007, pp. T2H–1–T2H–5. doi:10.1109/FIE.2007.4417928.

[8] Carlier, Stéphanie and Naessens, Vince and De Backere, Femke and De Turck, Filip, A software engineering framework for reusable design of personalized serious games for health : development study, JMIR SERIOUS GAMES 11 (2023) 16. URL: http://doi.org/10.2196/40054.

[9] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, J. Togelius, Deep Learning for Procedural Content Generation, Neural Computing and Applications 33 (2021) 19–37. doi:10.1007/s00521-020-05383-8.

[10] J. Jiang, F. Wang, J. Shen, S. Kim, S. Kim, A Survey on Large Language Models for Code Generation, 2024. doi:10.1145/3747588.

[11] M. F. Maleki, R. Zhao, Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 20 (2024) 167–178. doi:10.1609/aiide.v20i1.31877.

[12] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-Resolution Image Synthesis with Latent Diffusion Models, 2022. doi:10.48550/arXiv.2112.10752.

[13] W. Zhang, J. Yang, R. Tao, L. Chai, S. Guo, J. Wu, X. Chen, G. Cui, N. Ding, X. Xu, H. Wei, B. Zhou, V-GameGym: Visual Game Generation for Code Large Language Models, 2025. doi:10.48550/arXiv.2509.20136.

[14] C. Hu, Y. Zhao, J. Liu, Game Generation via Large Language Models, 2024. doi:10.48550/arXiv.2404.08706.

[15] K. Venkatesh, C. Dunlop, P. Yanardag, CREA: A Collaborative Multi-Agent Framework for Creative Content Generation with Diffusion Models, 2025. doi:10.48550/arXiv.2504.05306.

[16] Q. Wang, T. Wang, Z. Tang, Q. Li, N. Chen, J. Liang, B. He, MegaAgent: A Large-Scale Autonomous LLM-based Multi-Agent System Without Predefined SOPs, 2025. doi:10.48550/arXiv.2408.09955.

[17] Y. Dong, X. Jiang, J. Qian, T. Wang, K. Zhang, Z. Jin, G. Li, A Survey on Code Generation with LLM-based Agents, 2025. doi:10.48550/arXiv.2508.00083.

[18] J. He, C. Treude, D. Lo, LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead, 2025. doi:10.48550/arXiv.2404.04834.

[19] S. Hu, T. Huang, G. Liu, R. R. Kompella, F. Ilhan, S. F. Tekin, Y. Xu, Z. Yahn, L. Liu, A Survey on Large Language Model-Based Game Agents, 2025. doi:10.48550/arXiv.2404.02039.

[20] X. Xu, Y. Wang, C. Xu, Z. Ding, J. Jiang, Z. Ding, B. F. Karlsson, A Survey on Game Playing Agents and Large Models: Methods, Applications, and Challenges, 2024. doi:10.48550/arXiv.2403.10249.

[21] Z. Zhang, Y. Wang, C. Wang, J. Chen, Z. Zheng, LLM Hallucinations in Practical Code Generation: Phenomena, Mechanism, and Mitigation, 2025. doi:10.48550/arXiv.2409.20550.

[22] RACE model ChatGPT: better prompts with structure, 2024. URL: https://www.clickforest.com/en/blog/race-model-chatgpt-prompts.

[23] OpenAI, GPT-4o, 2024. URL: https://openai.com/index/hello-gpt-4o/.

[24] OpenAI, Openai o3-mini, 2025. URL: https://openai.com/index/openai-o3-mini/.

[25] Stanford Institute for Human-Centered AI, The 2025 AI Index Report, 2025. URL: https://hai.stanford.edu/ai-index/2025-ai-index-report.

[26] OpenAI, DALL·E 3, 2023. URL: https://openai.com/index/dall-e-3/.

[27] PixelLab - AI Generator for Pixel Art Game Assets, 2024. URL: https://www.pixellab.ai/.

[28] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, L. V. Gool, RePaint: Inpainting using Denoising Diffusion Probabilistic Models, 2022. doi:10.48550/arXiv.2201.09865.

[29] Validation of the GUESS-18: A Short Version of the Game User Experience Satisfaction Scale (GUESS), 2019. URL: https://research.google/pubs/validation-of-the-guess-18-a-short-version-of-the-game-user-experience-satisfaction-scale-guess/.