Immersive Media and User Experience

jimmux.com

Research Article

A Comparative Study of AI Tools and Techniques for Assisting Solo Artists and Programmers in Developing Video Games from Scratch

Sanjib Bashir Ahmad[1] and Siti Azreena Binti Mubin[2]

[1]*School of Computing, Asia Pacific University, Kuala Lumpur, Malaysia; E-mail: TP086597@mail.apu.edu.my*

[2]*School of Computing, Asia Pacific University, Kuala Lumpur, Malaysia; E-mail: siti.azreena@apu.edu.my*

Cite This: https://doi.org/10.65638/2978-8811.2026.2.04

ABSTRACT: Game development is a complex activity that requires the combined creative and technical skills in image-making, animation, audio, programming, and game design. Solo game development is hard, as it takes tremendous effort to be good at one skill, l*et al*one many. This paper investigates the extent to which solo developers can use AI-assisted workflows to develop playable game prototypes. Using AI for game asset generation and technical support, the production time for developing games can be significantly reduced. Combined with human skills and creativity, these generated assets can be transformed into high-quality production value. In programming and technical development, AI can be successfully used to debug, explain code, and support the creation of robust build systems and codebases. Other classical AI systems, such as knowledge engineering, have been used to develop a game subsystem logic using ontologies. Through multiple case studies and repeated game development, the study rigorously tests AI tools and techniques. Solo development challenges the perception that software and games can only be developed by institutions or teams. Solo development can foster new development practices and avenues for individual expression, thus enabling the creation of novel games that not only entertain but also carry meaning.

Keywords: AI, Solo developer, Game development, Art, Animation, Programming, Open source.

1. INTRODUCTION

Game development as a solo developer presents unique challenges, as a single person must perform a wide range of artistic and technical tasks to create games. Scoped game features, selective use of art techniques, robust code, and minimal library dependencies are major concerns in solo development.

AI is more than just Machine Learning (ML) and Large Language Model (LLM) powered generative tools based on neural networks. Although popularized by LLMs, neural network research dates back to 1943, when McCulloch and Pitts first introduced simple models of neurons and how a network of them can perform computations [1].

For the purpose of this study, AI applications in game development are grouped into three categories: generative AI, game AI, and developer-assist AI. Generative AI refers to tools that support the creation of visual assets, animation, music, sound, and design ideas. Game AI refers to techniques used inside games, such as pathfinding, NPC behavior, dynamic difficulty adjustment, and decision-making systems. Developer-assist AI refers to tools that support programming, debugging, build configuration, documentation lookup, and workflow planning.

Solo developers are rare in modern software and game development, where small indie teams are more common, typically ranging from two to ten people [2]. In

Received: April 26, 2026
Accepted: June 02, 2026
Published: June 04, 2026

multi-member teams, tasks are divided into skill-based groups, but solo developers have to do everything themselves.

Providing tools for solo developers can empower millions worldwide who are interested in creating games, not just for commercial motives, resulting in passion projects with artistic expression, experimental mechanics, narrative originality, or community engagement [3].

Game development consists of six major areas where AI can help: (a) generate images, (b) generate video for animation, (c) generate audio, (d) create infrastructure, (e) support programming, and (f) debug errors and warnings.

Beyond development efficiency, AI-assisted game development also has implications for immersive media, interaction design, and player experience. In games, the quality of player interaction is shaped not only by technical implementation but also by responsiveness, visual coherence, perceived polish, engagement, and immersion. Therefore, this study not only evaluates AI assistance in the production pipeline but also examines whether the resulting solo-developed prototype can offer a positive user gaming experience. This connection between AI-assisted solo development and player-perceived game quality remains underexplored, particularly in small-scale independent production contexts.

1.1. Problem Statement

Game development is a complex, multidisciplinary endeavor that requires both engineering and creative skills [4]. AI can help solo developers create games more easily by providing tools to generate game assets, support animation, programming, and debugging. Tools like Alpha3D, Leonardo AI, and Luma AI can help generate game assets [5]. Alpha3D supports creating 3D assets, Leonardo AI supports image and visual asset generation, and Luma AI supports video-based generation and animation workflows. These tools, among many, are examples of how AI can support common production needs in solo game development.

Using generated content and AI-assisted development, developers can reduce time and resource requirements to create early-stage prototype games, as found in a review of 3,091 posts by [6]. The efficiency of game development can be maximized by streamlining art and programming tasks with AI tools [7]. AI can also be used to support project management, planning, feature control, testing, and scope management, which are among the top problems in game development [8].

Although existing studies discuss generative AI tools, game AI techniques, and AI-assisted creative workflows [5] [6] [9], there remains limited empirical research on how these tools are used by solo developers across an actual game-development process. In particular, there is a lack of studies that combine development case studies with playtesting and evaluation of the resulting game prototype. This creates a gap in understanding whether AI-assisted workflows can help solo developers not only produce game assets and code more efficiently, but also create playable games that are perceived positively in terms of usability, visual quality, engagement, and overall user experience.

1.2. Research Questions

RQ1. What are the currently available AI tools and techniques that solo developers can use to create games?

RQ2. How can solo developers use AI tools and techniques in combination with their own creative and technical skills to create high-quality games?

RQ3. How does the use of AI-assisted tools affect the visual quality, performance, and user experience (UX) of the developed games, both in terms of visual content and performance?

1.3. Aim and Objectives

Aim: To evaluate the effectiveness of AI tools and techniques in supporting solo development through the production of multiple game development case-studies and user experience evaluation.

Objectives:

RO1. To identify, investigate, and categorize currently available state-of-the-art AI tools and techniques in video game development for generating images, animation, music, and programming and debugging support that can assist solo developers in creating video games.

RO2. To develop various small games, game engines, build systems, simulations, and a finished prototype using AI tools and techniques in combination with human creativity and artistic talent.

RO3. To evaluate the impact of AI-assisted development on game quality, performance, and player experience through user testing and structured feedback.

1.4. Scope of the Research

There are two applications of AI in video games: AI in games that exhibit intelligent behavior in games, and AI for games, which refers to the tools used to create games [10]. The scope of this study is limited to AI for video games. Furthermore, AI should be used strictly in an assistive role; otherwise, it can harm human creativity by making developers dependent on it [11].

AI-generated code, while useful for rapid prototyping, can raise concerns in production and maintenance, as it may lead to additional code cleanup and repair due to inconsistent style or hidden errors [6]. The emergence of services such as Code Janitor [12], which offers AI-code fixing for outputs from tools such as ChatGPT, GitHub Copilot, and Claude, illustrates the growing recognition of this maintenance concern. This research aligns with the craftsmanship of human engineering by writing hand-crafted code during the game development phase rather than relying on AI-generated code, while treating AI tools and techniques as assistive devices.

1.5. Significance of the Research

This research is significant as it contributes to the identification and application of AI tools and techniques suited to the needs and constraints of solo game development. Empowering solo developers to create video games leads to personal creative freedom and expression. As noted by [3], indie game development extends beyond profit-oriented goals, emphasizing individual artistic vision, experimental mechanics, narrative originality, and community engagement. More solo developers can have the freedom to create serious games and socially beneficial purposes, including skill acquisition and simulation-based training [13], reducing gender-based violence [14], and gamifying sociological research [15].

2. LITERATURE REVIEW

The literature review is organized into five thematic areas: (a) AI for game-development workflows, including asset generation, music, animation, and developer-support tools, (b) AI in gameplay systems, including NPC behavior, dynamic difficulty adjustment, MCTS, and personalization, (c) player experience, immersion, engagement, and UX in AI-assisted game prototypes, (d) AI risks and ethical concerns surrounding AI-assisted creativity and authorship in game development, (e) AI role in solo game development and the positive impact developers can have by creating games that benefit society.

The video game industry accounts for a significant share of the global entertainment economy, which has expanded rapidly due to technological innovation, increased digital connectivity, and the globalization of media markets [16]. The global gaming market has already surpassed both the film and music industries in growth and is projected to reach USD 521 billion by 2027 [3]. In this huge market, Steam remains one of the most popular platforms for game distribution for solo developers, with 8,588 indie games released in 2024 alone [17].

A wide range of AI tools and techniques have long been employed in game development, including rule-based systems, tree search algorithms, fuzzy logic, neural networks, genetic algorithms, and computer vision. These approaches are well established rather than being novelties [18].

2.1. AI for Game Development

The use of AI in game development encompasses generating visual, audio, and game content, as well as using various tools within game engines to support asset generation.

Visual Elements in Games

Generative AI-driven visualization tools and genetic algorithms have been adopted across multiple stages of game development, including procedural-level design and content generation. Machine-learning–based animation tools such as PoseNet and FaceMesh enable the generation of realistic character poses and facial expressions, thereby enhancing animation workflows [18].

Leading game engines increasingly integrate AI-driven tools, such as Unity Muse and Unity Sentis, as embedded AI solutions [6], Caffe framework, and UnrealCV plugin for 3D environment visualization [18]. Reference [19] further highlights AI toolsets available in Unity ML-Agents, CryEngine, and Unreal Engine to enhance gameplay systems and developer productivity.

Music and Sound in Games

AI techniques have also been successfully applied to game audio production. Gaussian Mixture Models (GMMs) and Generative Adversarial Networks (GANs) have been used to generate game music, as demonstrated by [20]. Reference [20] also highlights the use of AI-based audio tools such as DeepBach and JamBot, which can generate polyphonic melodies and derive subtracks from a primary audio track.

AI for Game Content

To create richer and more diverse game content, [21] explored the use of GPT-based language models to generate quests for Role-Playing Games (RPGs). Hand-authored narrative content is time-consuming and difficult to scale; language models can help alleviate this burden by generating narrative-rich, contextually coherent content.

2.2. AI in Gameplay Systems

The adoption of AI-driven game mechanics and algorithms enables solo developers to access capabilities once reserved for AAA studios. Because these techniques and tools are widely available, both independent developers and large studios can operate on a more level playing field with respect to core gameplay systems. Popular use cases include NPC and enemy behavior, search path, dynamic game difficulty, and evolving storyline.

AI-Driven Enemy and NPC Behavior

Machine learning models and neural networks allow Non-Playable Characters (NPCs) and enemies to learn from and respond dynamically to player actions and decisions [18]. A* and Dijkstra's algorithm are commonly implemented to support NPC navigation and enemy interactions [19]. Together, these AI techniques enable the creation of rich, engaging game worlds that enhance emotional immersion and player experience. Reference [19] also notes the use of Natural Language Processing (NLP) to generate realistic and emotionally expressive NPC dialogue.

Monte Carlo Tree Search (MCTS) Based on Surrogate Model

To address the need for real-time video games that require rapid decision-making and immediate feedback, [22] proposed a surrogate-model–based Monte Carlo Tree Search (MCTS) specifically for real-time environments. This approach has been empirically validated in games such as The Rumble Fish 2 and StarCraft II. Through experimentation, comprising 300 training rounds, 100 agent-versus-agent duels, and 100 test combat scenarios, [22] demonstrated that the surrogate model–based MCTS outperformed the traditional MCTS algorithm.

AI for Dynamic Difficulty Adjustment

Dynamic Difficulty Adjustment (DDA) refers to the adaptive modification of game difficulty in response to player performance. Reference [23] proposed a DDA framework based on reinforcement learning (RL) agents that dynamically adjust difficulty levels to help players maintain a flow state during gameplay, which was evaluated using an arcade shooter game, in which RL agents dynamically adjusted the difficulty to maintain an approximately 50% win–loss ratio.

Personalizing Gameplay Through AI

AI enables a high degree of personalization in gameplay, tailoring user experiences to individual player preferences by analyzing player behavior and choices [24], resulting in diversity and personalized gameplay.

AI vs. AI Gameplay: Spectator Games

Games can also be played with AI agents, giving rise to the emergence of spectator-focused games featuring AI vs. AI competition [25]. Multiplayer RTS titles such as Age of Empires and Starcraft allow players to configure matches in which AI opponents replace human participants. Reference [26] further explored this phenomenon on the Twitch platform, where AI-driven gameplay attracted large audiences.

Post-Gameplay: LLM-Based Fictional Believability in Characters

Reference [27] explores the concept of fostering relationships between players and fictional game characters as a means of increasing emotional depth and long-term engagement by leveraging AI agents powered by large language models (LLMs). In their study, [27] brought the characters Jerry and Catherine to life through conversational interactions with player participants.

2.3. AI, Immersion, and Game User Experience (UX)

Player experience is important when evaluating whether AI-assisted game development and the resulting prototype produce a satisfying level of immersion and a positive user experience (UX). In game studies and HCI, game user experience may include usability, playability, enjoyment, engagement, visual aesthetics, emotional response, and perceived satisfaction [28].

Similarly, the GameFlow model identifies concentration, challenge, player skill, control, clear goals, feedback, immersion, and social interaction as important elements of player enjoyment [29]. Flow theory is particularly relevant to games because player engagement is often

shaped by the balance between challenge and skill; when difficulty is too low, players may become bored, whereas excessive difficulty may lead to frustration or anxiety [29] [30].

Immersion and presence are also important in immersive media and game evaluation, as they can be subjectively measured through questionnaires that examine players' sense of involvement, interaction clarity, feedback, visual clarity, and the perceived believability of the game environment [31] [32].

Evaluation of AI-assisted game development should not be limited to development efficiency or production capability alone, but should also consider how players experience the resulting game in terms of usability, engagement, immersion, and overall satisfaction.

2.4. Risks and Ethical Use of AI

Numerous scholars have raised concerns about the ethical use of AI, particularly regarding originality and creative ownership [9]. When used in an assistive capacity, AI can promote creativity and value by enabling developers who lack comprehensive artistic or technical skill sets to complete their games and projects that might otherwise remain unfinished [6]. Claims that AI has already compromised human creativity [11] are not fully conclusive, as the generated content alone does not guarantee value or acceptance.

A study involving 42 online surveys and nine interviews with game developers, conducted by [9] indicates that AI is most advantageous when used to assist with or automate repetitive tasks, rather than to replace core creative activities.

2.5. Game Development Impact as a Solo Developer

Solo developers are uniquely positioned to act as agents of positive social change, as their work may be driven by values such as openness, collaboration, and activism, enabling the creation of games with societal impact [6]. This allows solo developers to explore themes and narratives that larger studios might consider commercially risky or unprofitable.

Reference [14] demonstrates how video games can reduce bias toward victims of gender-based violence and advocates the development of "games for good." Similar initiatives in the past have successfully addressed racial and gender biases by fostering empathy and perspective-taking through interactive gameplay.

3. METHODOLOGY

3.1. Research Design

This study adopts a mixed-methods research design combining: (a) game development using AI tools and techniques showcased as case studies, and (b) empirical evaluation through playtesting the prototype game and a user survey.

3.2. Game Development

The development process was documented to record quantitative data from: (a) days spent on each game development, (b) specific use cases of AI tools, (c) prompt evolution process to generate satisfactory game assets with cohesive style, (d) programming support use cases of AI, (e) assistance to setup build systems, editor configuration, and (f) debugging support uses to resolve errors and warnings.

The development process directly addresses the RQ2 and RO2. The final prototype game was used for playtesting. All developed games serve as artifacts of how AI tools and techniques were used to create games.

3.3. Playtesting and Survey Participants

The playtest involved 30 valid participants, selected from personal and professional contacts. Participants included both casual and experienced players, and no prior knowledge of AI tools was required. There was no mention of AI tooling for game development in the questionnaire to avoid any possible AI bias. Because the sample was based on convenience sampling, the results should be interpreted as early-stage prototype feedback rather than as a generalizable evaluation of all player groups. This is a possible sampling bias because most players were familiar with 2D platformers and thus may have adapted more quickly to the controls and level structure.

3.4. Questionnaire

The structured survey questionnaire was designed to evaluate player experience, usability, learning curve, and design effectiveness. It is aimed at aligning with Human-Computer Interaction (HCI), Game Design, and User Experience (UX). The questionnaire collected the following information: Section A: Participant Background,

Section B: Feedback, Controls & Responsiveness, Section C: Level Design & Difficulty, Section D: Challenge & Learning, Section E: Visuals & Theme, and Section F: Overall Experience. The questionnaire combines Likert-scale questions for quantitative data and open-ended responses for qualitative data. Sections B – F comprise Likert-scale questions, totaling 18. Section G consists of two open-ended questions.

The questionnaire sections were aligned with UX constructs. Controls and responsiveness were treated as indicators of usability; level design and hazard clarity were treated as indicators of interaction clarity; challenge and learning were linked to engagement and flow; visuals and theme were linked to aesthetics and immersion; and overall experience was used to assess perceived enjoyment and polish.

3.5. Ethical Considerations

This study did not collect any personal or indirect information that could be used to identify any participant, thereby ensuring full anonymity. Participation was completely voluntary, and no compensation was offered. Participants could withdraw at any time by not completing the survey. Participants were informed that the playtest and survey were used for an academic study.

4. DEVELOPMENT

The AI tools employed in this study for game asset creation and programming were intended to enhance and assist human creativity and development. Under no circumstances was AI used to replace the human aspect of innovation. The first two games were developed using Unreal Engine, and the next two using C++, SDL, and SFML graphics libraries. The C++ games were written from scratch, and the final game included a custom game engine. C++ was chosen as it's widely regarded in the industry for its high performance and low footprint. It has been in use for more than 20 years, with a developer community of approximately 4.5 million [33]. Knowledge engineering was used to develop the ontology demo using the popular hit game Stardew Valley as a case study. The Protégé software was used to develop the ontology.

4.1. Game Design Document

AI can generate Game Design Documents (GDDs) that serve as inspiration or for brainstorming new ideas. When generating GDDs, exploring the following points can be helpful for game development: (a) Level Purpose: objectives of the level, establishing stakes like enemies and obstacles, and combat rules, (b) Narrative: backstory that provides game direction and meaning, (c) Environment: visuals, color palette, elements acting as a cohesive whole, and (d) Core Mechanics: game controls, how actions should perform, edge cases.

4.2. Prompt Evolution

During the asset generation process, the term Prompt Evolution is introduced as a trial-and-error technique that involves continuously refining the prompt until a satisfactory asset is produced that matches the game's style and has high production quality. It's a continuous loop of incremental prompt refinement until generated assets match expectations.

There are various situations in which a prompt may not generate the expected image. The prompt is then refined to steer the AI in the right direction. Sometimes, a keyword for a specific art style may yield good results. The usage of the style is then continued, and other keywords are added or removed. Other times, instead of style, specifying art media such as watercolor or drawing may yield better results. There is no fixed formula for achieving good results, as it depends on various factors, such as game developers' preferences, the random seed, the AI tool, the model, and the tool's current version. The same fixed prompt may produce different results because of randomness.

Figure **1** below shows the Prompt Evolution Cycle, in which an initial prompt is first modified based on the prompt variables, then on the constants of art style and essential keywords. As the cycle goes on, based on the results, the developer may experiment with different offshoots and new pathways. The last step involves settling on the final image and generating a video from it for use in the animation phase. Example prompts and images are provided below to further clarify the process of prompt evolution.

Table **1** below shows the steps performed during the prompt evolution phase to generate the visual game assets. As can be seen in the table, it took eight cycles before arriving at the final results. Some cycles ended up being totally discarded as the result took new styles and pathways.

The gameplay videos of the developed games may be viewed to assess the image quality and animations used

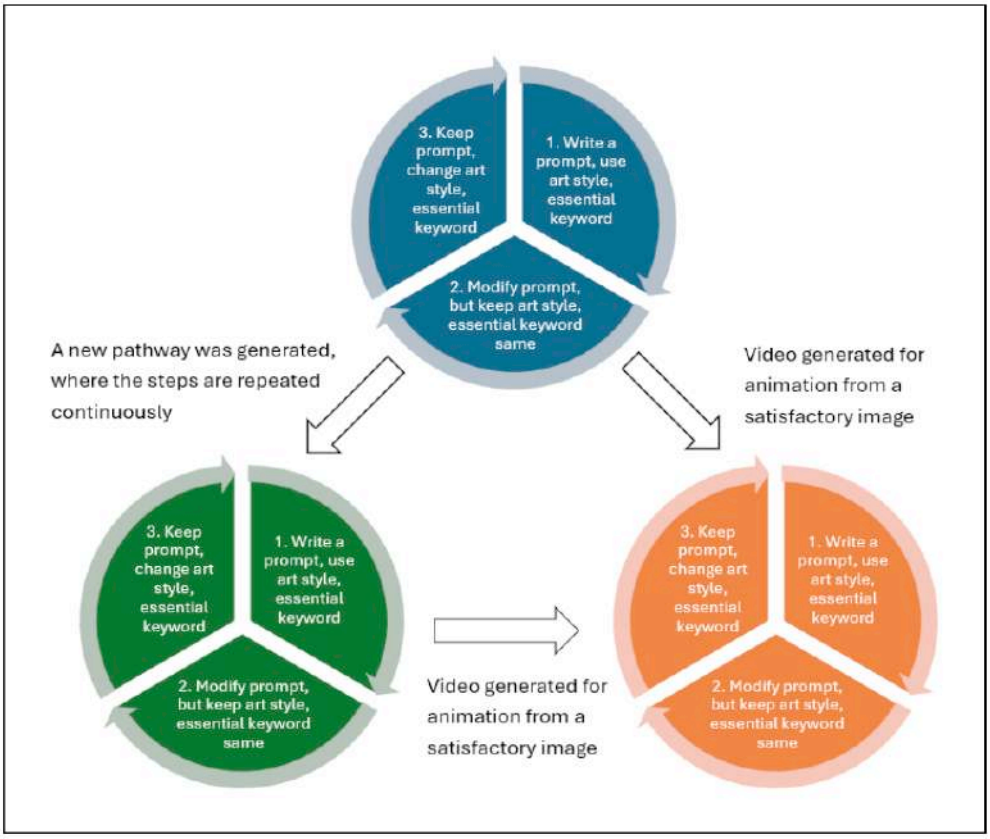

Figure 1: Prompt Evolution Cycle.

Table 1: Prompt Evolution Steps with Example from Game Case: Rebel Mouse.

Step Nr.	Step Definition	Example Prompt Used
1	Write an initial prompt describing the image wanted with 2 special keywords: spritesheet, art style	spritesheet mouse with gun shooting, [artist 1] style
2	Modify the prompt, introduce new elements, states, and positions, keeping spritesheet directive and art style same	spritesheet mouse idle, shooting gun, [artist 1] style
3	Modify prompt, new state, element, spritesheet directive, but new artist style	spritesheet mouse idle, walk, shoot gun, [artist 2] style
4	Modify prompt with new art style, and character state, style same as previous	pixel art spritesheet mouse standing idle, [artist 2] style
5	Style creator used by selecting a previous style that is preferred, prompt text simplified	create run animation sprites [based on style creator]
6	Style creator used. Repeat an existing prompt to see if randomness provides a different result	spritesheet mouse idle, walk, shoot gun, [artist 2] style
A satisfactory image is generated by this stage. Next stages include video generation for character animation.		
7	Generate a video for character animation. The desired image is selected, and the prompt used is a single word to animate the image.	run
8	Another video is generated for the use in character animation later, using a more extended prompt	spritesheet mouse idle, walk, shoot gun, [artist 2] style

in the games [34]. Figure **2** below shows the actual images generated for the game "Rebel Mouse". Figure **3** shows the final image that was chosen for the game.

4.3. Asset Generation Pipeline

To create game assets, the following pipeline was used: (a) idea for game assets and style, (b) asset generation

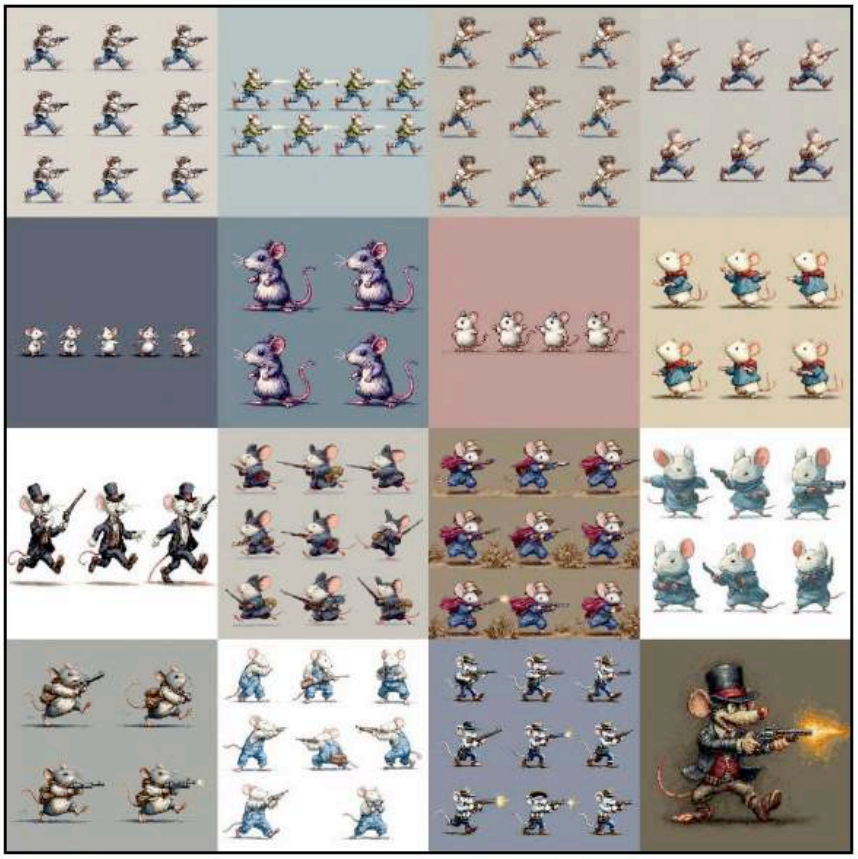

Figure 2: Images Generated During Prompt Evolution.

Figure 3: Final Image Chosen for Rebel Mouse Game.

and prompt evolution by iteratively refining the prompt until the right asset and style are generated, and (c) editing generated assets in graphics or audio software for image, animation, and sound.

4.4. Image Generation

AI tools can be used to generate images for players, NPCs, and enemies; backgrounds and tiles; game objects; obstacles; and decorations.

Character generation is challenging because the same character must be created in multiple poses, like idle, run, and jump. Currently, generative AI tools do not support character preservation across multiple sessions. In a single run, multiple poses of the same character must be generated satisfactorily. The limitation is that generated images must be edited using a new prompt, or multiple poses must be specified in a single prompt.

4.5. Videos and Animation

Once a satisfactory image is generated that meets the specified quality and style requirements, it is animated. The resulting video is edited in graphics software to select animation keyframes and transform them into a spritesheet. Animation is an important aspect of game development, as it can clearly distinguish well-polished games from mediocre ones. A considerable amount of time is also required to produce smooth, high-quality animations. In addition to character animation, game objects can be animated to provide more polish. In the final prototype, smoke from the broken bus and the finishing-line red flag were animated.

4.6. Sound and Music

It's possible to generate high-quality, production-level, full-length musical compositions. LLMs have also enabled the training of custom voices or the subscription to celebrity voices. This has great potential in games. Generated sound or music needs to be edited in an audio software to clip a specific part for use in the game. The clips can then be used as background music, trigger events, loops, and movement sounds.

4.7. Infrastructure, Editor Support, and Build System

AI can assist in setting up and configuring build systems and editors. As open-source software was used, the infrastructure can be deployed without any major changes across multiple operating systems and platforms. GCC C++ compiler, Make, SDL, SFML graphics libraries, ImGui, Lua support with sol2 and an LSP server were installed.

LSP Compliant Editor Support

The LSP configuration support is not limited to any specific editor. Any LSP-compliant editor can be used to support C++ development. LSP servers can turn a simple editor without any C++ support into a full IDE-like editor with support for syntax highlighting, code suggestions, and warnings.

Build System

There are various open-source build systems available for cross-platform development, such as Make, CMake, and Ninja. Make is the simplest, hence it was used for the four games developed from scratch in C++. Using AI assistance, the generated Makefile supports incremental builds via object files, thereby significantly reducing compilation time.

Software development, deployment, and a smooth automated build process have been the subject of study for almost fifty years due to the variety of tools and methods used in software development and evolution [35]. Developing a robust build system, particularly for C++ game development, can be highly beneficial because it's a one-time investment with benefits that apply to all future projects. Significant effort was spent to create a robust cross-platform C++ development environment.

4.8. Programming

AI was used to support programming strictly in the following areas: (a) explain code, (b) code analysis for best practices, and (c) API and documentation lookup. AI was not used to generate code in any of the games developed. Programming is perceived as an enjoyable activity; therefore, using AI to generate code is counterintuitive.

Also, AI-generated code may create maintainability issues when used without sufficient review, testing, or architectural control, thus making it difficult to fix, as the problem may not be evident on first use, and may create additional maintenance work [6]. In small game projects, this risk can be significant because the solo developer is responsible for understanding and maintaining the entire codebase.

4.9. Debugging

AI tools can be very helpful for debugging errors and warnings, as they greatly reduce development time. C++ compilation errors can be cryptic, and runtime errors are

Figure 4: Samurai Toad.

difficult to trace. Compiling in small incremental steps is a good strategy for catching errors early. AI tools were used and proved effective in detecting and correcting these errors.

It is also useful to adopt modern C++ features to prevent errors and warnings, as they enforce sound programming practices, resulting in cleaner, more concise code. For example, the KDE project has been steadily upgrading its codebase to adopt more modern C++ features [36].

5. CASE STUDIES

The goal of creating many games in a short four-month time frame was to rigorously test the game development process using AI assistance, especially for games developed in C++ without a ready-made game engine.

5.1. Samurai Toad

A simple battle draw game played against the game AI, developed using Unreal Engine and Blueprints. Whoever draws first wins, but drawing too fast or too late causes the player to lose [34].

This is the first game developed during the study. It focuses on using generative AI tools for game art. 108 images were generated over the course of 27 prompts before it was possible to generate two satisfactory images with the two characters in idle, attack, and defeat

Figure 5: Catch Me.

Figure 6: Rebel Mouse.

poses. The background was relatively faster to generate, requiring 3 prompts and 12 images. Two additional game assets were created manually in a graphics editor.

5.2. Catch Me

A keyboard-based runner game where player acceleration and speed are maintained by rapidly pressing two buttons. Played against a simple in-game AI. This 2D game was also created using Unreal Engine and Blueprints [34].

AI was used to create art for the player and the game AI character, as well as animations and backgrounds. Generated background is a single image that was tiled manually in a graphics editor, then chained together to create a seamless background. The animation sequence was manually created from an AI-generated video. The idle pose was created from the run pose, so the character's limbs were manually sliced and reassembled for the idle pose.

5.3. Rebel Mouse

A 2D platformer game created in C++ and the SDL3 graphics library. The player is a mouse who shoots down monsters that burrow into the ground when killed. The mouse can be controlled to jump, run, and shoot [34].

As in the previous two games, the game images and videos were generated iteratively using AI, with many prompts. The generated images and videos were then manually edited in a graphics editing software to create the final animation keyframes, spritesheets, and background images.

5.4. Ontology Case Study of Stardew Valley

A small ontology was developed to simulate the gifting game mechanic and NPC friendship system found in the popular commercial game Stardew Valley [34].

Ontologies are regarded as a powerful knowledge representation technique in AI systems. They enable the definition of classes, objects, and data properties, and axioms using logic-based formalisms for representing concepts and their relationships, supporting automated reasoning and consistency checking [37]. The created ontology can answer real gameplay questions, such as where a particular NPC lives, when their birthday is, what gifts they love, and how those gifts can be crafted. To validate the ontology, functional SPARQL queries were developed and executed to successfully retrieve such questions. The implementation demonstrates the possibility of the ontology acting as a backend that a game engine written in C++, Java, or Python could query via available APIs.

5.5. Final Prototype: Gubi Goes to APU

The final prototype game is a 2D side-scrolling platformer developed using C++ and the SFML graphics library. AI tools and techniques were used to create GDDs, art

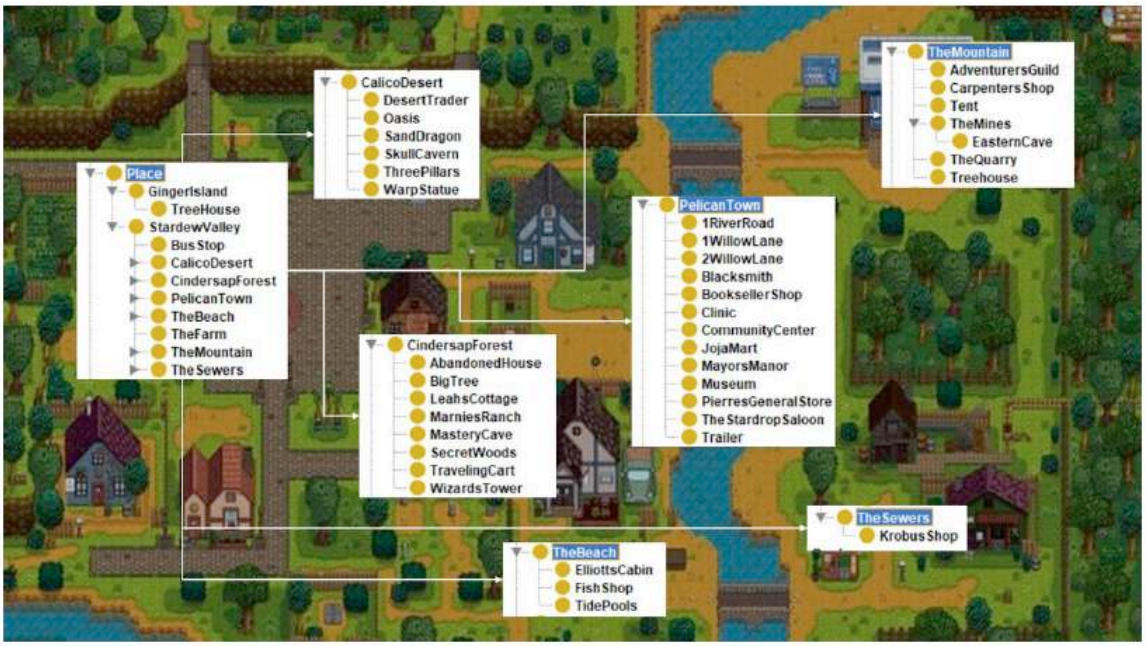

Figure 7: Ontology Case Study.

assets, animation videos, and support for programming and debugging [34].

The game's backstory is that Gubi's school bus breaks down, and he must walk to school along the road. There is debris on the road that Gubi must jump over. There are large sinkholes on the road from falling satellite debris, which Gubi must avoid. There are small moving enemies and obstacles, such as fan rotor blades, that Gubi must jump over. The level is completed when Gubi reaches the end marked by a red animated flagpole.

6. DISCUSSION AND FINDINGS

The findings suggest that AI-assisted workflows were most effective in supporting rapid prototyping, visual asset production, animation preparation, debugging, and technical configuration. Across the case studies, AI tools helped fill knowledge and skill gaps by producing high-quality game assets and reducing friction in tasks that commonly slow down development, such as resolving technical errors. However, the findings also show that human judgment remained essential for game design,

Figure 8: Gubi Goes to APU.

asset selection, manual editing, programming decisions, level design, and final integration.

Overall, the survey results indicate generally positive player perceptions of the prototype, particularly regarding usability, visual presentation, level clarity, and enjoyment. At the same time, lower scores related to perceived danger, goal communication, and content depth suggest that the prototype would benefit from further design refinement. These findings indicate that AI-assisted workflows can support the development of a playable, well-received early prototype, but they do not eliminate the need for human-led iteration and design improvements.

6.1. Development Findings

Results of the various game development data are summarized in Table **2** below: (a) development time, (b) AI use for game assets, and (c) AI use for technical support. The data shows that AI assistance was used most consistently in visual asset creation, video-based animation preparation, and debugging. The strongest benefit was observed in generating character images, backgrounds, and animation sources, as these tasks are time-consuming and require substantial creative effort, and in rapid-iteration tasks such as resolving programming errors. AI did not replace human development work for software architecture and game engine implementation. All original creative ideas for the game were created solely by humans. Manual effort remained necessary for selecting usable assets, editing visuals, cleaning up images, creating sprite sheets, and, most importantly, creating animation keyframes. Animation keyframes required significant effort, as the visuals had to be manually selected, timed, and tested to ensure smooth, realistic animation. In addition, level design, library integration, and gameplay refinement and polish were performed manually. It can be concluded that AI functioned as an accelerative support mechanism rather than an autonomous development system.

Furthermore, within 106 days, five demos were developed, two of which were written from scratch in C++, demonstrating that AI-assisted development is feasible for solo developers, both for asset generation and technical support. Without AI support, producing the same number of prototypes would likely have been significantly more time-consuming, particularly for asset generation, animation preparation, debugging, and technical troubleshooting.

6.2. Survey Results

The survey evaluates the prototype game for user-perceived game quality, performance, visuals, and player experience through a 5-point Likert scale: (a) Strongly Disagree, (b) Disagree, (c) Neutral, (d) Agree, and (e) Strongly Agree.

All the questions were required. Feedback was received and recorded from 30 participants over 8 days. There were 32 submissions; 2 were spam and were discarded. After the target of 30 feedback was received, the survey was closed.

To support a structured analysis, questions were grouped together as follows: (a) Controls & Responsiveness: responsiveness, jump precision, sense of control, (b) Level Design: level clarity, obstacle clarity, hazard fairness, and enemy predictability, (c) Challenge & Learning: difficulty, player improvement, understanding failures, (d) Visuals & Theme: environment, story, and visuals consistency, meaningful goal, and (e) Overall Experience: game enjoyment, feel for polish, play more.

These five groups can be interpreted to connect with UX and immersive media evaluation as follows: (a) Controls & Responsiveness: usability and interaction quality indicators, (b) Level Design: interaction clarity, spatial understanding and flow support, (c) Challenge & Learning: engagement and flow indicators, (d) Visuals & Theme: immersion and aesthetic-coherence indicators,

Table 2: Game Development Data.

Game Name	Development Time	Game Asset	Technical Support
Samurai Toad	21 days	Image	Debugging
Catch Me	21 days	Image, video, sound	Debugging
Rebel Mouse	28 days	Image, video	Programming, debugging, and build support
Ontology Case Study	10 days	None	Programming
Gubi Goes to APU	26 days	Image, video	Programming, debugging

Table 3: UX and Immersive Media Connection With Likert-Scale Question Groups.

Group Name	Survey Items	UX and Immersive Media Construct
Controls & Responsiveness	Responsiveness, jump precision, sense of control	Usability and interaction quality
Level Design	Level clarity, obstacle clarity, hazard fairness, and enemy predictability	Interaction clarity, spatial understanding, flow support
Challenge & Learning	Difficulty, player improvement, and understanding failures	Engagement and flow
Visuals & Theme	Environment, story, and visuals consistency, meaningful goal	Immersion and aesthetic coherence
Overall Experience	Game enjoyment, feel for polish, play more	Perceived enjoyment, satisfaction, game polish

and (e) Overall Experience: perceived enjoyment, satisfaction and game polish.

Table **3** above summarizes the group structure and its connections to UX and immersive media indicators, where Level Design can be perceived as a bridge between usability and flow because clear level design helps players remain engaged instead of confused [29].

6.3. Quantitative Analysis

The sample is largely composed of regular players with moderate to high familiarity with 2D platformers. This is useful for early prototype feedback, but may be slightly biased towards higher perceived usability, as experienced players adapt more quickly to a new game. Tables **4** and **5** below present the survey data.

The survey findings can be interpreted through three UX-related constructs: usability, engagement, and immersion. The Controls and Responsiveness items primarily relate to usability, as they assess whether players can control Gubi effectively and interact with the game without unnecessary friction. The positive scores for control responsiveness SQ3 (M = 4.27), jump precision SQ4 (M = 4.07), and sense of control SQ5 (M = 4.3) suggest that the prototype provided a usable interaction experience for an early-stage game.

The Level Design and Challenge & Learning items relate to engagement and flow. In flow theory, engagement is supported when the level of challenge is reasonably balanced with player skill. The high scores for level understanding SQ6 (M = 4.43) and understanding failure SQ12 (M = 4.47) suggest that players were able to understand the level structure, recognize obstacles SQ7 (M = 4.27), sinkholes and falling space debris SQ8 (M = 4.03), and learn from repeated attempts SQ11 (M = 4.27). This indicates that the prototype supported basic flow conditions by reducing confusion and allowing players to focus on improvement. However, the lower score for difficulty appropriateness SQ10 (M = 3.97) suggests that challenge balancing still requires refinement.

The lower scores for danger SQ13 (M = 3.97) and end-goal communication SQ15 (M = 3.93) may be explained by the prototype's limited scope, short level length, and lack of the next level, where the story could have further evolved and continued the narrative.

6.4. Qualitative Analysis

Qualitative responses support the findings by showing that many participants highlighted smooth controls, responsiveness, and the visual art style, while some

Table 4: Survey Play Tester Demographics.

Question	Choice	Answers
SQ1. How often do you play video games?	Plays daily	15
	Plays weekly	8
	Plays monthly	5
	Plays rarely	2
SQ2. How familiar are you with 2D platformer games?	Very familiar	15
	Somewhat familiar	13
	Not familiar	2

Table 5: Likert Scores for Section B – F.

Group	Likert Scores	Mean	Median	Std. Deviation
Controls & Responsiveness	SQ3. The controls felt responsive	4.27	4	0.83
	SQ4. Jumping felt precise	4.07	4	0.87
	SQ5. I felt in control of Gubi	4.3	4.5	0.84
Level Design	SQ6. The level was easy to understand	4.43	4.5	0.63
	SQ7. Obstacles were visually clear	4.27	4.5	0.87
	SQ8. Sinkholes and falling space junk were fair	4.03	4	0.72
	SQ9. Enemy movement was predictable	4.1	4	0.84
Challenge & Learning	SQ10. The difficulty felt appropriate	3.97	4	0.72
	SQ11. I improved as I played	4.27	4	0.83
	SQ12. When I failed, I understood why	4.47	4.5	0.57
Visuals & Theme	SQ13. The environment communicated a sense of purpose and a little danger	3.97	4	0.81
	SQ14. The story context matched the visual cues (bus, stadium, buildings, enemies, obstacles)	4.13	4	0.86
	SQ15. Reaching APU at the end felt meaningful	3.93	4	1.11
Overall Experience	SQ16. I enjoyed playing this level	4.4	5	0.81
	SQ17. The game felt polished	4.07	4	0.87
	SQ18. I would like to play another level	4.3	5	0.99

noted that the level was not sufficiently engaging and that the difficulty, enemies, and content depth required further refinement. Tables **6** and **7** below show the thematic coding of all the responses received for SQ19 and SQ20.

Table 6: Thematic Coding "SQ19. What Did You Enjoy Most?"

Theme	Count
Visuals, style, art, atmosphere	16
Controls and responsiveness	6
Level flow and design	4
Accessibility & Learning Curve	4
Polish	2

The responses for SQ19 were mainly concentrated around the “visuals, style, art, and atmosphere” theme, suggesting that AI-assisted visual generation was one of the most noticeable features of the prototype and an important element in the development pipeline. It further aligns with the positive quantitative scores for visual consistency SQ14 and overall enjoyment SQ16. Similarly, positive comments on controls and responsiveness support the positive mean scores for the controls-related items SQ3, SQ4, and SQ5, while comments on level flow and design correspond with the level-design findings SQ6, SQ7, SQ8 and SQ9.

Since visual assets were among the primary uses of AI in the case studies, this result indicates that players responded positively to an area where AI assistance had a direct impact. However, the final quality of these visuals also depended on human creativity, including idea generation, prompt refinement, art-style referencing, asset selection, editing, and integration into the game.

Table 7: Thematic Coding "SQ20. What Should be Improved?"

Theme	Count
More content, longer levels	9
Enemies, obstacles, hazards	8
Polish	5
Visuals	4
Controls, jumping	3
Stasis	2
AI Resistance	2

Game improvement-related responses for SQ20 were mainly related to longer levels, additional content, enemy variety, obstacles, and hazards. These responses align with the relatively lower quantitative scores for difficulty appropriateness SQ10, environmental danger and challenge SQ13, and the need for narrative continuation

or goal communication SQ15. This suggests that while the prototype was generally received positively, participants still expected greater content depth, stronger challenge progression, and clearer explanation at the end of the current level.

Suggestions regarding jump controls and polish appear to qualify the positive results for SQ17, in which participants indicated that the game felt polished. Rather than contradicting the quantitative finding, these comments suggest that the prototype achieved an acceptable level of polish for its current stage, while further improvements are possible.

The AI Resistance theme refers to some participants' concerns or hesitation toward the use of AI-generated content in game development. Although this theme appeared only twice, which is less frequently than other comments, it is nonetheless important because it reflects broader ethical and perceptual concerns surrounding AI-assisted creativity. Some players may evaluate AI-assisted games based on their personal views about originality, authorship, and human creative involvement.

Overall, the two qualitative questions and the quantitative findings point to a consistent interpretation of the prototype's strengths and weaknesses. The prototype was perceived positively in terms of visuals, controls, and basic playability, while content depth, challenge progression, polish, and perceptions of AI-assisted creativity remain areas for further refinement.

6.5. Discussion

This study aims to evaluate whether AI-assisted development can enable a solo developer to produce a playable, high-quality video game prototype that delivers a positive player experience. The findings provide meaningful evidence supporting this objective while also highlighting areas where human refinement remains essential.

The high ratings for control responsiveness, clarity, and art-enjoyment suggest that the AI-assisted development approach was effective in producing a functionally sound, visually rich, and engaging prototype. Players were able to understand the level structure, anticipate hazards, and recover from failure with minimal frustration, which are key indicators of basic game quality.

From an immersive media perspective, the positive responses to visuals and control responsiveness indicate that players were able to engage with the prototype at both aesthetic and usability levels. Using a UX lens, the controls relate to usability, the challenge and learning responses relate to flow, and the visual-theme responses relate to immersion and aesthetic coherence. However, lower scores related to danger, goal communication, and content length show that immersion depends not only on asset quality but also on level pacing, narrative clarity, interaction depth, and emotional engagement.

The survey's quantitative data mostly show a slight positive skew. This is a promising indicator for early platformer-type prototype games developed by indie developers, as positive user feedback tendencies are interpreted positively in UX research. Furthermore, the structured player feedback data, with Likert-scale values, is also a reliable indicator of game satisfaction and UX quality [28].

The generally positive results may partly reflect sample characteristics and testing context. Many participants were regular players or familiar with 2D platformers, which may have made it easier for them to understand the controls and level structure. In addition, convenience sampling through personal and professional contacts may have introduced a positive response bias, as respondents were known to the researchers. Therefore, the findings should be interpreted as promising early-stage feedback rather than conclusive evidence of broad player acceptance.

These findings indicate that AI-assisted tools, when combined with human design and creativity, can support the development of early-game prototypes with strong usability performance, particularly in terms of control responsiveness, the player's sense of control, and positive feedback on the visual art style.

This study contributes to existing literature in three ways: (a) it provides practice-based evidence and game artifacts of how AI tools and techniques can support solo game development across asset generation, animation preparation, technical troubleshooting, and programming support, (b) it documents multiple development case studies rather than discussing AI tools only at a conceptual level, and (c) it connects AI-assisted development with player-facing evaluation by examining usability, visual quality, engagement, and perceived experience through playtesting and survey feedback.

7. CONCLUSIONS AND FUTURE WORK

This study evaluated how AI tools and techniques can support solo game development through the development of multiple demo games and case studies, and the playtesting of a final 2D platformer game

prototype. The findings indicate that AI assistance can support prototyping, visual asset production, animation preparation, debugging, and technical setup, while human creativity, design, and intelligence remain essential for design direction, programming game engine architecture, animation keyframe generation, asset refinement, and gameplay integration. The player feedback further suggests that an AI-assisted workflow can produce an early prototype with positive perceived usability, visual appeal, and overall enjoyment, although further refinement is required in level depth, challenge design, immersion, and narrative communication.

Future work can include larger-scale user studies with more diverse player groups to validate the findings across different experience levels and gameplay preferences. Further research can also use formal immersion and engagement measurement instruments to evaluate the influence of AI-generated assets on player experience. Comparative studies of AI-assisted vs. non-AI-assisted development would be informative and interesting to conduct.

In addition, future work can also include creating an AI adoption framework for game development targeted specifically for solo programmers and artists based on the development experience gained from this research. While in-game AI was not explored in this paper, it can be further studied in relation to solo development, particularly what in-game AI algorithms can be used for easy and quick implementation to enhance game features.

While generative AI can greatly reduce development time by generating game assets or providing technical assistance, further work on other AI tools, such as expert systems and knowledge engineering, can lead to novel approaches in software development by abstracting logic and programming complexity and reducing cognitive load among solo developers.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHORS' CONTRIBUTIONS

Sanjib Bashir Ahmad conceived, designed the study and developed the game prototype. Siti Azreena Mubin assisted with data collection, reviewed and approved the final version of the manuscript.

FUNDING

No funding was received for this research work.

REFERENCES

[1] N. J. Nilsson, The Quest for Artificial Intelligence. Stanford University, 2009.

[2] M. U. Fahme and A. Adiba, "Building Your Dream Team: How Indie Teams Can Form, and Thrive Together," (in eng), F1000Res, vol. 13, p. 42, 2024, doi: 10.12688/f1000research.139274.1.

[3] E. Goh, O. Al-Tabbaa, and Z. Khan, "Unravelling the complexity of the Video Game Industry: An integrative framework and future research directions," Telematics and Informatics Reports, vol. 12, p. 100100, 2023/12/01/ 2023, doi: https://doi.org/10.1016/j.teler.2023.100100.

[4] J. Chueca, J. Verón, J. Font, F. Pérez, and C. Cetina, "The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games," Information and Software Technology, vol. 165, p. 107330, 2024/01/01/ 2024, doi: https://doi.org/10.1016/j.infsof.2023.107330.

[5] A. Begemann and J. Hutson, "Empirical insights into AI-assisted game development: A case study on the integration of generative AI tools in creative pipelines," 2024, generative AI; game development pipeline; conceptualization; 3D modeling; ethical considerations vol. 5, no. 2, 2024–07–02 2024, doi: 10.54517/m.v5i2.2568.

[6] R. Panchanadikar and G. Freeman, ""I'm a Solo Developer but AI is My New Ill-Informed Co-Worker": Envisioning and Designing Generative AI to Support Indie Game Development," Proceedings of the ACM on Human-Computer Interaction, vol. 8, no. CHI PLAY, pp. 1–26, 2024, doi: 10.1145/3677082.

[7] S. Hemraj, "Maximizing Efficiency in Game Development Through Art Styles, AI Integration, and Creative Expression," (in En), Loading, vol. 17, no. 27, pp. 56–78, 2025, doi: https://doi.org/10.7202/1118424ar.

[8] C. Politowski, F. Petrillo, G. C. Ullmann, J. d. A. Werly, and Y.-G. Guéhéneuc, "Dataset of Video Game Development Problems," presented at the Proceedings of the 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 2020. [Online]. Available: https://doi.org/10.1145/3379597.3387486.

[9] S. Alharthi, "Generative AI in Game Design: Enhancing Creativity or Constraining Innovation?," Journal of Intelligence (J. Intell.), vol. 13, 05/20 2025, doi: 10.3390/jintelligence13060060.

[10] P. Roberts, Artificial Intelligence in Games. CRC Press, 2023.

[11] R. J. Sternberg, "Do Not Worry That Generative AI May Compromise Human Creativity or Intelligence in the Future: It Already Has," Journal of Intelligence, vol. 12, no. 7, doi: 10.3390/jintelligence12070069.

[12] Code Janitor. "Professional AI code fixing service." code-janitor.dev. https://code-janitor.dev/ (accessed 2026).

[13] J. Pérez, M. Castro, and G. López, "Serious Games and AI: Challenges and Opportunities for Computational Social Science," IEEE Access, vol. 11, pp. 62051–62061, 2023, doi: 10.1109/ACCESS.2023.3286695.

[14] Z. M. C. v. B. Sweeney Jing Li, "Video games for good: Active perspective-taking fosters empathy and reduces implicit bias toward gendered violence victims," Entertainment Computing, vol. 53, p. 100928, 2025/03/01/ 2025, doi: https://doi.org/10.1016/j.entcom.2025.100928.

[15] A. Gazis and E. Katsiri, "Gamifying Sociological Surveys Through Serious Games—A Data Analysis Approach Applied to Multiple-Choice Question Responses Datasets," (in English), Computers, vol. 14, no. 6, p. 224, 2025, doi: https://doi.org/10.3390/computers14060224.

[16] M. J. P. Wolf, Sources: Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming. Reference & User Services Quarterly, 2013.

[17] SteamDB. "Steam Game Releases Summary for Indie." SteamDB. https://steamdb.info/stats/releases/?tagid=492 (accessed 2026).

[18] Y. Wu, A. Yi, C. Ma, and L. Chen, "Artificial intelligence for video game visualization, advancements, benefits and challenges," Mathematical Biosciences and Engineering, vol. 20, no. 8, pp. 15345–15373, 2023, doi: 10.3934/mbe.2023686.

[19] V. Chergarova, M. Tomeo, H. Morgan, and S. Andrade, "Integrating Artificial Intelligence (AI) into Game Development to Elevate Diverse Gameplay Elements," Issues In Information Systems, vol. Volume 25, no. Issue 2, pp. 408–417, 2024, doi: 10.48009/2_iis_2024_132.

[20] Z. Xiang and Y. Guo, "Controlling Melody Structures in Automatic Game Soundtrack Compositions With Adversarial Learning Guided Gaussian Mixture Models," IEEE Transactions on Games, vol. 13, no. 2, pp. 193–204, 2021, doi: 10.1109/TG.2020.3035593.

[21] S. Värtinen, P. Hämäläinen, and C. Guckelsberger, "Generating Role-Playing Game Quests With GPT Language Models," IEEE Transactions on Games, vol. 16, no. 1, pp. 127–139, 2024, doi: 10.1109/TG.2022.3228480.

[22] M.-J. Kim, D. Lee, J. S. Kim, and C. W. Ahn, "Surrogate-assisted Monte Carlo Tree Search for real-time video games," Engineering Applications of Artificial Intelligence, vol. 133, p. 108152, 2024/07/01/ 2024, doi: https://doi.org/10.1016/j.engappai.2024.108152.

[23] L. Climent, A. Longhi, A. Arbelaez, and M. Mancini, "A framework for designing Reinforcement Learning agents with Dynamic Difficulty Adjustment in single-player action video games," Entertainment Computing, vol. 50, p. 100686, 2024/05/01/ 2024, doi: https://doi.org/10.1016/j.entcom.2024.100686.

[24] W. Yu and Z. Xia, "The Enhancement of Video Games by AI," Academic Journal of Science and Technology, vol. 15, pp. 210–214, 05/19 2025, doi: 10.54097/mdsa4882.

[25] R. Summerley and B. McDonald, "Perceived Foolishness: How Does the Saltybet Community Construct," Games and Culture, p. 46, March 27, 2024 2024, doi: 10.1177/15554120241238262.

[26] M. R. Johnson and N. J. Jackson, "Twitch, Fish, Pokémon and Plumbers: Game live streaming by nonhuman actors," Convergence, vol. 28, no. 2, pp. 431–450, 2022, doi: 10.1177/13548565221074804.

[27] Y. Sun et al., "Bring game characters to the social space: Developing storytelling community AI agents driven by LLMs," Entertainment Computing, vol. 54, p. 100948, 2025/06/01/ 2025, doi: https://doi.org/10.1016/j.entcom.2025.100948.

[28] M. H. Phan, J. R. Keebler, and B. S. Chaparro, "The Development and Validation of the Game User Experience Satisfaction Scale (GUESS)," Human Factors, vol. 58, no. 8, pp. 1217–1247, 2016, doi: 10.1177/0018720816669646.

[29] P. Sweetser and P. Wyeth, "GameFlow: a model for evaluating player enjoyment in games," Comput. Entertain., vol. 3, no. 3, p. 3, 2005, doi: 10.1145/1077246.1077253.

[30] C. J. Larche and M. J. Dixon, "The relationship between the skill-challenge balance, game expertise, flow and the urge to keep playing complex mobile games," (in English), Journal of Behavioral Addictions, vol. 9, no. 3, pp. 606–616, 12 Oct. 2020 2020, doi: https://doi.org/10.1556/2006.2020.00070.

[31] C. Jennett et al., "Measuring and defining the experience of immersion in games," International Journal of Human-Computer Studies, vol. 66, no. 9, pp. 641–661, 2008/09/01/ 2008, doi: https://doi.org/10.1016/j.ijhcs.2008.04.004.

[32] B. G. Witmer and M. J. Singer, "Measuring Presence in Virtual Environments: A Presence Questionnaire," Presence: Teleoperators and Virtual Environments, vol. 7, no. 3, pp. 225–240, 1998, doi: 10.1162/105474698565686.

[33] B. Stroustrup, "Thriving in a crowded and changing world: C++ 2006–2020," Proc. ACM Program. Lang., vol. 4, no. HOPL, p. Article 70, 2020, doi: 10.1145/3386320.

[34] S. Ahmad. "Supporting Resources for "A Comparative Study of AI Tools and Techniques for Assisting Solo Artists and Programmers in Developing Video Games from Scratch"." Github. https://github.com/sanjib/study_ai_solo_game_dev (accessed 2026).

[35] T. G. G. Dalibor Fonović, "A Quantitative Study of C/C++ FOSS Software Buildability," Proceedings of the Ninth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, 2022.

[36] Fausto Carvalho, Rafael Campos Nunes, Rodrigo Bonifácio, João Saraiva, and P. Accioly, "Embracing modern C++ features: An empirical assessment on the KDE community," Journal of Software: Evolution and Process, vol. 36, no. 5, 2024, doi: https://doi.org/10.1002/smr.2605.

[37] N. Noy and D. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Knowledge Systems Laboratory, vol. 32, 01/01 2001.