Literature Review: AI-Assisted Worldbuilding

Jennifer Ried

Technische Hochschule Ostwestfalen-Lippe

Department of Media Production

December 2025

Contents

Literature Review: AI-Assisted Worldbuilding for Independent Game Development

1. Introduction

The video game industry presents a paradox. While digital distribution has democratized game publishing—any developer can now reach millions of players through platforms like Steam, itch.io, or the Epic Games Store—the production of visually competitive 3D games has become exponentially more resource-intensive. A 2D RPG like *Chrono Trigger* (1995) was created by a team of roughly 50 people. Three decades later, *Elden Ring* (2022) required an estimated 400+ developers working for nearly six years. The transition from 2D sprites to 3D worlds, from MIDI soundtracks to orchestral scores, from text dialogue to fully voiced characters with facial animation, has inflated production requirements by an order of magnitude.

This literature review investigates whether recent advances in artificial intelligence and procedural generation can reverse this trend—whether the "creator-to-creation ratio" that allowed small teams to build expansive game worlds before the 3D era can be restored through computational assistance. The central research question driving this review asks: *Can AI and procedural tools enable solo developers or small teams to create open-world RPG experiences that previously required large studios and massive budgets?*

This question has urgency. The cost of AAA game development continues to escalate—*Grand Theft Auto VI* is reportedly budgeted at over $2 billion. Meanwhile, player expectations shaped by AAA productions apply to all games. The indie developer faces an impossible standard: compete visually with teams of hundreds while working alone or with a handful of collaborators. The hypothesis underlying this thesis is that AI changes this calculus. Not by fully automating game development—the technology is not there—but by amplifying human creative capacity to the point where small teams can produce content at scales previously requiring large organizations.

To answer this question, the review synthesizes research across fifteen distinct but interconnected domains: from foundational machine learning (transformers, diffusion models, neural radiance fields) through procedural content generation (classical algorithms and AI-enhanced techniques) to practical production tools (animation systems, audio generation, game engine integration). A total of 206 papers were analyzed, spanning publication dates from 1985 to 2025, with particular emphasis on work from 2020–2024 when many critical capabilities reached practical maturity. The review is intentionally broad because the thesis project requires integration across all these domains—a character needs a 3D model (Domain 4), rigging and animation (Domain 8a), voice (Domain 8b), dialogue (Domain 8c), and behavior (Domain 5), all placed in a procedurally generated world (Domains 2a, 2b) rendered in a game engine (Domain 8g).

The tiered citation methodology employed here distinguishes between mega-foundational works (Tier 1, with over 10,000 citations in major AI domains), major foundational contributions (Tier 2, 1,000–10,000 citations), field-defining papers (Tier 3, 100–1,000 citations), and emerging work (Tier 4, under 100 citations but published in top venues within the past two years). This approach ensures the review builds on established foundations while capturing cutting-edge developments that have not yet accumulated citations but represent the state of the art.

The review is organized along the content creation pipeline: first examining the AI foundations that enable all subsequent techniques, then tracing how these technologies generate visual content, populate worlds with intelligent characters, integrate with production tools, and ultimately cohere into believable fictional worlds. The final section synthesizes findings across domains to assess feasibility for independent development.

---

2. Foundations of Modern AI

2.1 From Rules to Learning

The history of artificial intelligence divides into two distinct paradigms. The classical era (1956–2010) relied on human experts encoding knowledge as explicit rules—what researchers called symbolic AI or expert systems. Medical diagnosis systems like MYCIN contained hundreds of hand-written conditional statements; chess programs like Deep Blue evaluated positions using carefully crafted heuristics (Shortliffe, 1976; Campbell et al., 2002). These systems worked within narrow, well-defined domains but failed at tasks requiring perception or nuance. Deep Blue could defeat world champions at chess but could not recognize a chess piece in a photograph.

The modern era emerged from a convergence of three developments between 2007 and 2012. NVIDIA's release of CUDA in 2007 enabled graphics cards to perform general-purpose computation, eventually training neural networks 10–50× faster than CPUs. The ImageNet database, containing 14 million labeled images across 20,000 categories, provided the training data that data-hungry neural networks required (Deng et al., 2009). And algorithmic improvements, particularly Hinton et al.'s (2006) demonstration of effective deep network training through unsupervised pre-training, reignited interest in neural approaches.

The pivotal moment came in 2012. Krizhevsky, Sutskever, and Hinton entered their neural network "AlexNet" in the ImageNet Large Scale Visual Recognition Challenge and won by a staggering margin—achieving 15.3% error where the next-best approach managed 26.2%. This was not incremental improvement but a paradigm shift. Within two years, every major technology company had pivoted to neural network approaches (Krizhevsky et al., 2012).

2.2 The Attention Revolution

The next breakthrough came in 2017. Vaswani et al. published "Attention Is All You Need," introducing the Transformer architecture that would reshape natural language processing—and eventually, image and video generation as well. The key innovation was the self-attention mechanism: rather than processing sequences element by element (as recurrent networks did), Transformers could attend to any position in the input simultaneously, discovering which parts of a sequence relate to which other parts (Vaswani et al., 2017).

The practical impact was enormous. BERT (Devlin et al., 2019) demonstrated that pre-training a Transformer on massive text corpora, then fine-tuning on specific tasks, could achieve state-of-the-art results across diverse benchmarks with minimal task-specific architecture. GPT-3 (Brown et al., 2020) scaled this approach to 175 billion parameters and demonstrated emergent capabilities: the model could perform tasks it was never explicitly trained for, simply by being shown a few examples in the prompt. This "few-shot learning" meant that a single model, without modification, could write code, translate languages, answer questions, and generate creative text.

For game development, these capabilities translate directly to NPC dialogue, lore generation, quest design, and procedural narrative. The chain-of-thought prompting technique introduced by Wei et al. (2022) enables complex multi-step reasoning—generating quest logic that accounts for prerequisites, consequences, and player choices. Code generation models like Codex (Chen et al., 2021) and Code Llama (Rozière et al., 2023) can produce functioning game scripts from natural language descriptions.

2.3 Diffusion Models for Visual Synthesis

While Transformers revolutionized language, a parallel revolution transformed image generation. Generative Adversarial Networks (GANs) had dominated visual synthesis since their introduction by Goodfellow et al. (2014), but suffered from training instability and mode collapse—the tendency to generate limited variations rather than the full distribution of possible images.

Diffusion models offered an alternative. Ho, Jain, and Abbeel's 2020 paper on Denoising Diffusion Probabilistic Models (DDPM) demonstrated that a simple process—gradually adding noise to images, then training a network to reverse that process—could match or exceed GAN quality while training more stably (Ho et al., 2020). The insight was elegant: rather than generating images directly, the network learns to remove noise. Given pure static, it iteratively denoises until a coherent image emerges.

Rombach et al. (2022) made diffusion practical for high-resolution image generation by introducing latent diffusion models. Instead of operating in pixel space (computationally expensive for large images), latent diffusion compresses images to a learned latent space, performs diffusion there, then decodes the result. This 10–100× speedup enabled models like Stable Diffusion to run on consumer hardware.

The final piece was conditioning—steering generation toward desired outputs. Radford et al.'s CLIP model (2021), trained on 400 million image-text pairs, learned to connect visual and linguistic representations in a shared embedding space. Text-to-image systems like DALL-E 2 (Ramesh et al., 2022) and Imagen (Saharia et al., 2022) used CLIP embeddings (or similar contrastive learning) to guide diffusion toward images matching text prompts. The result: photorealistic image generation from natural language descriptions.

The practical implications for game development are profound. Concept art that required hours of artist iteration can now be explored in minutes. Style guides become trainable LoRA weights rather than written documents. Environment mood boards generate programmatically from text descriptions. The question shifts from "can we create this image" to "can we guide AI to create it consistently." This is less about replacing artists than about changing what artists do—from pixel-level execution to higher-level creative direction.

2.4 Neural 3D Representations

The leap from 2D image synthesis to 3D content generation required new representations for three-dimensional scenes. Mildenhall et al.'s (2020) Neural Radiance Fields (NeRF) represented a breakthrough: encoding a 3D scene as a continuous function that maps spatial coordinates and viewing directions to colors and densities. A multi-layer perceptron learns this function from photographs taken around an object or scene. Once trained, the network can render photorealistic novel views from angles never photographed—the camera position simply becomes an input to the learned function (Mildenhall et al., 2020).

NeRF achieved stunning visual quality but rendered slowly, requiring millions of network queries per image. Kerbl et al.'s 3D Gaussian Splatting (2023) addressed this limitation by representing scenes as explicit collections of 3D Gaussians rather than implicit neural functions. The result: real-time rendering at 30+ frames per second while maintaining high visual quality—the first neural 3D representation suitable for interactive applications like games (Kerbl et al., 2023).

Text-to-3D systems like DreamFusion (Poole et al., 2022) and Magic3D (Lin et al., 2023) closed the loop by applying 2D diffusion models as priors for 3D generation. The core insight, called Score Distillation Sampling, is that a 2D diffusion model already knows "what real images look like" from any viewpoint. By rendering a 3D representation from multiple angles and asking the diffusion model to improve each rendering, the optimization gradually sculpts a 3D object consistent with the text prompt. Magic3D outputs game-engine-ready meshes with textures at 8× higher resolution than DreamFusion—though the "Janus problem" (objects with faces on multiple sides, since each view independently wants to match the prompt) remains an active research challenge.

---

3. Procedural Content Generation

3.1 Classical Algorithms

Procedural content generation (PCG) predates the AI revolution by decades. Ken Perlin's gradient noise algorithm (1985), developed for the film *Tron*, remains foundational to terrain generation in 2025—heightmaps created by summing octaves of Perlin noise at different frequencies produce convincingly natural landscapes (Perlin, 1985). L-systems, formalized by Lindenmayer and extended by Prusinkiewicz (1990), encode plant growth as formal grammars—production rules that iteratively expand symbols into complex branching structures. Every procedural tree in modern games descends from this mathematical biology (Prusinkiewicz & Lindenmayer, 1990).

The PCG research community systematized these techniques through comprehensive taxonomies. Togelius et al.'s 2011 survey classified approaches by what content they generate (levels, rules, narratives) and how quality is evaluated (theory-driven, human-guided, or automatic). Shaker, Togelius, and Nelson's 2016 textbook remains the definitive reference, covering noise functions, cellular automata, L-systems, and search-based generation—the use of evolutionary algorithms to optimize content against fitness functions encoding design goals (Togelius et al., 2011; Shaker et al., 2016).

Wave Function Collapse (WFC), introduced by Maxim Gumin in 2016, deserves special mention despite its non-academic origins. The algorithm, which constrains tile placement based on adjacency rules learned from example inputs, has accumulated over 21,000 GitHub stars and seen adoption across indie games, Unreal Engine plugins, and academic research. It exemplifies how practical innovations sometimes emerge from game development communities rather than research institutions (Karth & Smith, 2017).

The theoretical landscape of PCG also includes cellular automata for cave generation (the classic "drunk walk" algorithms), Voronoi diagrams for region partitioning, and graph grammars for dungeon connectivity. Each technique trades control for variety: more random algorithms produce diverse outputs but are harder to constrain to design intentions; more structured algorithms offer predictability but risk feeling formulaic. The art of PCG lies in layering these techniques—using L-systems for plant placement on terrain shaped by noise functions, within regions defined by Voronoi cells, connected by graph-grammar-generated paths.

3.2 Machine Learning Enhanced PCG

Classical PCG hits a quality ceiling. Perlin noise produces terrain that looks procedural; hand-crafted levels feel more intentional. The insight driving Procedural Content Generation via Machine Learning (PCGML) is that neural networks can learn the implicit design patterns from human-created content and generate new content following those learned distributions.

Summerville et al.'s 2018 survey established the PCGML taxonomy, distinguishing approaches by their generative model: sequence models (LSTMs generating level segments as token sequences), constructive models (GANs and autoencoders synthesizing content directly), and quality-diversity approaches combining ML with evolutionary search (Summerville et al., 2018). The survey's 603+ citations indicate rapid field maturation.

Volz et al. (2018) demonstrated a powerful hybrid: training a GAN to generate Mario levels, then using evolutionary search (CMA-ES) in the GAN's latent space to find levels meeting specific constraints—playability, difficulty, or aesthetic criteria. The neural network provides quality; the search provides control. This pattern recurs across PCGML: combine learned priors with controllable optimization.

PCGRL—Procedural Content Generation via Reinforcement Learning—frames level design as an agent placing tiles in an environment, receiving rewards for producing playable, interesting levels (Khalifa et al., 2020). The agent learns design principles through trial and error. Crucially, for independent developers with limited training data, techniques like Torrado et al.'s (2020) bootstrapping approach enable GANs to generalize from just a handful of example levels.

The 3D frontier remains less explored but viable. Giacomello et al. (2018) applied GANs to DOOM level generation—the first 3D application of PCGML. Wulff-Jensen et al. (2017) generated 3D terrain using GANs trained on digital elevation models. The techniques exist; integration with modern game engines like Unreal Engine 5 is the remaining challenge.

---

4. Visual Content Generation

4.1 From Text to Image

The practical implications of diffusion models for game development became clear with the release of Stable Diffusion in August 2022. An open-source model, running on consumer graphics cards, could generate concept art that previously required hours of artist time. Within months, techniques for fine-tuning and controlling these models proliferated.

LoRA (Low-Rank Adaptation) addressed the challenge of style consistency. Training a full diffusion model requires massive datasets and computational resources; LoRA instead learns small adapter weights that modify a frozen base model's behavior (Hu et al., 2021). An artist can train a LoRA on 20–50 images of their character design style, then generate hundreds of variations matching that style—maintaining visual coherence across an entire game's worth of assets.

DreamBooth and Textual Inversion extended this personalization. DreamBooth learns to associate a specific subject (a particular character, a unique architectural style) with a special token, enabling generation of that subject in novel contexts from just 3–5 reference images (Ruiz et al., 2022). Textual Inversion achieves similar results by learning custom word embeddings rather than modifying model weights (Gal et al., 2022). IP-Adapter enables image-guided style transfer without any training—providing a reference image directly conditions the generation (Ye et al., 2023).

ControlNet revolutionized spatial control. Zhang, Rao, and Agrawala (2023) introduced trainable modules that accept structural guides—edge maps, depth images, pose skeletons—and condition the diffusion process accordingly. This means character pose from a 3D reference, building layout from an architectural sketch, or composition from a rough depth map can drive generation while the AI fills in details matching the text prompt (Zhang et al., 2023).

For game developers, these tools compose into a practical pipeline: generate concept art with text prompts; establish visual style via LoRA; maintain character consistency through DreamBooth; control composition with ControlNet. Node-based interfaces like ComfyUI make this pipeline accessible to non-programmers.

4.2 The 3D Asset Pipeline

Generating 3D assets from text remains more challenging than image synthesis, but progress has been rapid. DreamFusion established the paradigm: use a pre-trained 2D diffusion model to supervise 3D optimization (Poole et al., 2022). Each iteration renders the evolving 3D representation from a random viewpoint, queries the diffusion model for how to improve that rendering, and backpropagates gradients through the renderer to update the 3D representation.

Magic3D improved upon DreamFusion with a two-stage coarse-to-fine approach, achieving 8× higher resolution and producing textured meshes rather than neural radiance fields—directly usable in game engines (Lin et al., 2023). GET3D generates textured 3D shapes in a single forward pass, learning from 2D image collections without 3D supervision (Gao et al., 2022). For rapid prototyping, Point-E and Shap-E trade quality for speed, generating initial 3D shapes in 1–2 minutes versus an hour (Nichol & Jun, 2022; Jun & Nichol, 2023).

3D Gaussian Splatting has emerged as the representation of choice for real-time applications. While NeRF requires expensive network evaluations for each camera ray, Gaussian Splatting renders by projecting and blending explicit 3D primitives—achieving 30+ fps on consumer hardware. Unreal Engine 5 plugins already support Gaussian Splat assets, enabling neural 3D content in game pipelines (Kerbl et al., 2023).

The text-to-3D quality gap is closing. Recent work addresses the Janus problem through multi-view supervision and viewpoint-aware generation. While outputs still require cleanup for production use, the trajectory is clear: text-to-3D is approaching practical maturity.

---

5. Intelligent Characters

5.1 The Emotion Foundation

Believable NPCs require more than pathfinding and dialogue trees. Joseph Bates's 1994 paper "The Role of Emotion in Believable Agents" established the foundational principle: emotion is central to believability, not merely ornamental (Bates, 1994). Drawing from Disney animation's principles of appeal and exaggeration, Bates argued that characters must appear to have inner lives—desires, fears, reactions to events. The Oz Project at Carnegie Mellon, which Bates led, produced a generation of researchers who operationalized these ideas.

This tradition reached its fullest expression in Façade (2003, 2005). Mateas and Stern created Trip and Grace, two AI characters navigating a troubled marriage as the player visits their apartment. The system combined natural language understanding, emotional models, dramatic beats, and reactive planning in the ABL language they developed. Façade proved that emotionally complex, conversationally responsive NPCs were achievable—though the bespoke development effort required was immense (Mateas & Stern, 2003, 2005).

Industry practice standardized on simpler approaches. Behavior Trees, popularized by Halo 2's AI, provide clear authorial control and debuggability. Goal-Oriented Action Planning (GOAP) enables dynamic behavior selection based on world state. The 2020s survey by Uludağlı et al. documents this mature—but fundamentally limited—technology stack (Uludağlı et al., 2020).

5.2 The LLM Revolution

Large language models have transformed what's possible for NPC behavior. Park et al.'s "Generative Agents" paper (2023) demonstrated the paradigm shift: 25 LLM-powered agents inhabiting a simulated town, forming relationships, remembering experiences, and exhibiting emergent social behaviors—organizing a Valentine's Day party that no researcher scripted (Park et al., 2023). The architecture combines a memory stream (timestamped natural language observations), reflection (periodic synthesis of memories into higher-level insights), and planning (generating action sequences based on character goals and recent events).

The citation response—3,753 citations within a year—signals how rapidly the field has embraced this approach. Park et al. (2024) scaled to 1,000 agents accurately replicating human survey responses, demonstrating feasibility at population scales relevant to open-world games.

For independent developers, cost matters. "Lyfe Agents" achieved 10–100× cost reduction versus the original Generative Agents implementation while maintaining real-time interaction capability (Lyfe Agents, 2023). This makes LLM-powered NPCs economically viable for indie projects.

The practical architecture for an RPG likely combines approaches: major characters receive full generative agent treatment with memory and reflection; quest-critical NPCs use hybrid systems with scripted cores and LLM dialogue expansion; background townspeople employ cheaper, simpler language models or procedural templates. The days of writing thousands of dialogue lines manually may be ending.

5.3 Quest and Dialogue Generation

Beyond character behavior, the narrative backbone of RPGs—quests and dialogue—presents distinct generation challenges. A quest is not merely a sequence of objectives but a narrative structure with prerequisites, consequences, player agency, and emotional beats. Traditional dialogue trees branch exponentially; a conversation with five choices at each of three exchanges yields 125 unique paths. Manual authoring at this scale is prohibitive for indie development.

Early approaches to procedural quest generation drew from grammar-based methods. Dormans and Bakkes (2011) proposed generating quests through graph transformation systems, where narrative templates expand according to grammatical rules—the "fetch quest" archetype expands into travel, acquisition, and return subgoals, each recursively elaborated. Kybartas and Bidarra's 2017 survey documented a decade of procedural narrative research, categorizing approaches as structure-based (following dramatic templates), simulation-based (emergent from agent interactions), or optimization-based (searching for narratively interesting sequences) (Kybartas & Bidarra, 2017).

Planning-based approaches bring AI reasoning to quest generation. Ammanabrolu et al.'s work on text-based game generation (2020–2024) used knowledge graph grounding to ensure quest coherence—NPCs that know about artifacts actually possess them; locations mentioned in dialogue exist in the world. Their CALM framework (Conversational Agentic Language Model) demonstrated that LLMs with explicit world state tracking could participate meaningfully in interactive fiction, suggesting architectures for quest-aware dialogue systems (Ammanabrolu et al., 2020).

The integration of LLMs transformed what is possible. Fan et al. (2024) proposed using chain-of-thought prompting for multi-step quest generation, where the model explicitly reasons through narrative structure before generating quest details. The approach generates quests that respect established lore by conditioning generation on a world knowledge base—something impossible for traditional procedural approaches that lack semantic understanding.

Dialogue generation has seen similar transformation. Yu et al.'s survey of dialogue systems (2023) documented the shift from retrieval-based (selecting from pre-written responses) through generation-based (producing novel text) to hybrid architectures combining both. For RPG NPCs, hybrid approaches offer control: designers define canonical responses for critical information while LLMs fill conversational space around those anchors.

Controllability remains the core challenge. Players expect NPCs to know specific information (quest details, lore) while exhibiting personality and maintaining conversation history. Techniques from constrained language generation—controlled generation through prompt engineering, fine-tuning on character-specific corpora, or using knowledge-grounded generation with retrieval augmentation—address these requirements (Li et al., 2022). The practical implementation involves NPC "personas" embedding character backstory, world knowledge relevant to that character, and interaction history into the prompt context.

5.4 Multi-Agent Development Systems

A distinct but related application of multi-agent research targets the game development process itself rather than in-game characters. If LLM agents can simulate believable townspeople, can they simulate a development team?

ChatDev (Qian et al., 2023) introduced the paradigm: LLM agents assuming roles (CEO, CTO, programmer, designer, tester) collaborating through structured dialogue to produce functional software from requirements. The system achieved 86.66% executability on test cases, producing working code through simulated discussion, code review, and iteration. For game development, this suggests AI-assisted pipelines where agents specialize in narrative design, level layout, balance testing, or bug identification.

The architecture extends to creative collaboration. MetaGPT (Hong et al., 2023) formalized multi-agent workflows with structured operating procedures—standardized formats for requirements documents, design specifications, and code reviews that agents produce and consume. This reduces the "telephone game" degradation where information corrupts through agent chains. Applied to worldbuilding, specialized agents might handle geography, political systems, cultural development, and historical events, coordinating through shared world documents.

Scaling multi-agent simulations revealed both capabilities and limits. Li et al.'s (2024) experiments with up to 10 million agents demonstrated emergent phenomena—social networks, information propagation, coordination—but also showed computational costs that limit real-time applications. For game development workflows (non-real-time), these costs are manageable; for in-game simulation, efficiency remains critical.

The multi-agent paradigm suggests a future where independent developers orchestrate AI collaborators rather than doing everything manually. A designer might specify world requirements, receive generated lore from a “historian” agent, geography from a “cartographer” agent, political structures from a “sociologist” agent, and quest hooks from a “narrative designer” agent—all coordinating through natural language and structured knowledge representations. While this vision remains nascent, early implementations demonstrate feasibility.

---

6. Production Pipeline Integration

6.1 Animation and Rigging

Character animation has historically required specialized talent and expensive motion capture. Recent ML advances are democratizing this pipeline. Holden, Saito, and Komura’s 2016 work introduced neural motion synthesis—learning motion manifolds from example clips to enable style transfer and blending. Phase-Functioned Neural Networks (PFNN) achieved real-time character control at 60+ fps by conditioning network weights on the locomotion cycle phase (Holden et al., 2016, 2017).

Motion Matching, introduced by Clavet at GDC 2016 and adopted by *For Honor*, *Ghost of Tsushima*, and *The Last of Us Part II*, searches motion capture databases at runtime to find the best matching pose for current inputs. Learned Motion Matching (Holden et al., 2020) compresses these databases 70×—from 590MB to 8.5MB—making large motion libraries practical. Unreal Engine 5.4 includes native Motion Matching support, bringing AAA animation quality to all developers without licensing fees (Clavet, 2016; Holden et al., 2020).

Text-to-motion is the emerging frontier. MDM (Human Motion Diffusion Model) generates character animation from natural language descriptions: “a person walks forward then jumps” produces the corresponding motion sequence (Tevet et al., 2023). MoMask achieves current state-of-the-art performance with FID scores of 0.045 on standard benchmarks (Guo et al., 2024). Motion-Agent integrates these capabilities with conversational refinement—describing, critiquing, and iterating on animations through dialogue (Wu et al., 2024).

Automatic rigging removes another bottleneck. RigNet learns end-to-end rigging from mesh geometry, producing skeletons and skinning weights without manual intervention (Xu et al., 2020). HumanRig (2024) specifically targets humanoid characters, achieving production-quality results on AI-generated 3D meshes—closing the loop from text-to-3D to game-ready animated character.

6.2 Audio Generation

The audio pipeline has seen parallel transformation. MusicGen generates background music from text descriptions—"epic orchestral battle theme with brass and timpani"—in a single open-source model (Copet et al., 2023). AudioLDM produces sound effects from descriptions: "sword clashing against shield," "footsteps on gravel," "wind howling through ruins" (Liu et al., 2023). AudioLDM 2 unifies speech, music, and effects in a single model (Liu et al., 2024).

Voice synthesis has reached emotional expressiveness. XTTS supports 17 languages with voice cloning from 6-second samples, generating distinct NPC voices for an entire game from minimal reference audio. The MIT license enables commercial use. Bark adds non-verbal expressiveness—laughter, sighs, hesitations that make dialogue feel human (Casanova et al., 2024; Shen et al., 2023).

NVIDIA's Audio2Face closes the facial animation loop. Given dialogue audio, it generates 52+ blend shapes with emotion detection from voice tone—real-time, streaming, compatible with standard animation rigs (NVIDIA, 2024). Combined with text-to-speech, this enables procedural generation of fully voiced, facially-animated dialogue without recording a single line.

6.3 Houdini and Unreal Engine Integration

Two tools dominate production workflows for procedural world generation: Houdini for content creation and Unreal Engine 5 for runtime rendering.

Houdini's procedural paradigm—node-based networks defining operations rather than storing static results—aligns naturally with AI-enhanced generation. The official Houdini 20 release added ONNX support for running arbitrary neural networks, PCA for dimensionality reduction, and Python ML nodes for distributed processing. Community tools like MLOPS integrate Stable Diffusion directly, enabling AI texture and image generation within Houdini graphs (SideFX, 2023, 2024).

Research on neural terrain generation is replacing Perlin noise with learned approaches. The Terrain Diffusion Network (Hu et al., 2023) uses sketch-guided synthesis with climatic awareness—understanding how erosion, tectonics, and weather patterns shape landscapes. A 2025 paper proposes diffusion as a direct successor to Perlin noise, offering infinite seamless generation with constant-time random access (Goslin, 2025). Infinite Texture generates arbitrarily large seamless textures from text descriptions (Wang et al., 2024).

Unreal Engine 5's PCG Framework, introduced in version 5.2, provides native node-based procedural generation. Combined with World Partition (streaming large worlds), Nanite (virtualized geometry handling billions of triangles), and Lumen (real-time global illumination), the engine supports open-world development that was previously studio-exclusive (Epic Games, 2023).

The ML Deformer, available since UE 5.2, uses neural networks to approximate complex muscle and cloth simulation at runtime—training offline on physics-accurate simulations, then running efficiently in-game. Motion Matching became native in UE 5.4. The Neural Network Engine (NNE), maturing through versions 5.3–5.5, enables arbitrary ONNX model inference at runtime—custom style transfer, learned behaviors, or any other neural functionality (Epic Games, 2024).

Third-party integration extends these capabilities. Convai provides conversational AI for NPCs with long-term memory, environmental awareness, and action execution—NPCs that remember previous conversations and perform in-game actions based on dialogue. This single plugin addresses one of the most labor-intensive aspects of RPG development: populating worlds with responsive characters (Convai, 2024).

6.4 The Integration Challenge

While individual AI capabilities have matured rapidly, integrating them into coherent production pipelines remains the primary challenge for independent developers. The tools exist in isolation: text-to-image generates concept art; text-to-3D produces assets; auto-rigging creates skeletons; text-to-motion generates animations; text-to-speech produces dialogue; Music generation creates soundtracks. But connecting these capabilities—ensuring a character's generated appearance matches their dialogue tone matches their animation style matches their role in generated quests—requires careful orchestration.

Several architectural patterns are emerging. The **hub-and-spoke model** centers on a game engine (typically Unreal or Unity) with plugins connecting to external AI services. Convai, for example, handles NPC dialogue while the engine manages gameplay. This approach leverages mature game development workflows but introduces latency and dependency on external services.

The **pipeline model** processes assets through sequential stages: concept art   3D model   rigging   animation   integration. Each stage can employ different AI tools, with human checkpoints for quality control. Houdini excels at this approach, with its procedural nodes encapsulating AI operations that can be chained, cached, and version-controlled.

The **generative database model** treats AI as a content generation backend. Rather than pre-generating all content, the system generates on-demand from specifications. An NPC's dialogue generates at conversation time from their persona description; a building's interior generates when the player enters; quest variations generate based on player history. This approach maximizes variety while minimizing storage, but requires careful latency management.

For independent developers, the practical approach likely combines all three: a pipeline for asset creation (manageable batches with review), hub-and-spoke for runtime AI (dialogue, behavior), and generative database for variation (procedural details, ambient content). The specific mix depends on project scope, available compute, and quality requirements.

6.5 Quality Assurance in AI-Assisted Development

Traditional game QA assumes human-authored content with predictable failure modes. AI-generated content introduces novel challenges: hallucinated lore that contradicts established facts; generated NPCs that behave inappropriately; procedural quests with impossible objectives; visual assets with artifact or quality inconsistencies.

Emerging approaches address these challenges at multiple levels. **Constraint satisfaction** bounds generation outputs—NPCs must reference only items that exist; quests must have completable objectives; geography must be navigable. Knowledge graphs and world state representations enable these constraints, ensuring generated content respects established facts.

Statistical monitoring detects distribution shifts. If generated dialogue suddenly skews negative or visual outputs exhibit new artifacts, automated monitoring flags the regression. Techniques from ML operations (MLOps) apply: tracking model versions, dataset versions, and output statistics over time.

Automated testing extends to generated content. Generated quests can be played by AI agents verifying completability. Generated dialogue can be analyzed for offensive content, contradictions, or out-of-character statements. Generated assets can be validated against technical specifications (polygon counts, UV mapping, bone weights).

Human-in-the-loop curation remains essential for quality-critical content. The most practical AI-assisted workflow generates candidates for human selection and refinement rather than deploying AI output directly. A concept artist reviews ten generated variations, selecting and refining the best; a writer reviews generated dialogue, editing for voice and consistency; a level designer evaluates generated layouts, adjusting for playability.

The economic question is where the quality-cost curve crosses. At what point does human curation of AI-generated content cost more than manual creation? Current evidence suggests the crossover has occurred for high-volume, moderate-quality content (dialogue variations, background assets, ambient audio) but not yet for hero content (main characters, key story moments, signature visuals). For independent developers, this translates to a hybrid strategy: AI for breadth, human craft for depth.

---

7. Worldbuilding Theory

7.1 Subcreation and Believability

Procedural techniques can generate content; worldbuilding theory explains what makes that content feel like a coherent world rather than a collection of assets. J.R.R. Tolkien’s 1947 essay “On Fairy-Stories” introduced the concept of subcreation—the artist as “subcreator” bringing forth secondary worlds governed by their own internal laws. The key to believability, Tolkien argued, is consistency: within its own frame of reference, the fantasy world must operate by its own rules without contradiction (Tolkien, 1947).

Mark J.P. Wolf extended Tolkien's framework in *Building Imaginary Worlds* (2012), analyzing how fictional worlds across media achieve depth through infrastructure: maps, timelines, languages, cultures, histories. Wolf distinguishes between invention (new ideas), completeness (how much is developed), and consistency (how well elements cohere). For procedural generation, consistency is the critical challenge—ensuring generated content respects the world's established patterns (Wolf, 2012).

Janet Murray's *Hamlet on the Holodeck* (1997) established core properties of digital narrative environments: procedural (governed by encoded rules), participatory (responding to user input), spatial (navigable as virtual space), and encyclopedic (holding vast stores of information). Her triad of immersion, agency, and transformation describes what players seek from virtual worlds—the sensation of transportation, the power to take meaningful action, and the opportunity to become someone else (Murray, 1997).

7.2 Transmedia and Environmental Storytelling

Henry Jenkins's concept of transmedia storytelling describes how narratives unfold across multiple media platforms, each making distinctive contributions—the game expanding on the film which references the novel which inspired the comic. For worldbuilding, Jenkins emphasizes that worlds are not just containers for stories but systems that can generate stories. This "negative capability"—leaving gaps that invite exploration and speculation—is essential to player engagement (Jenkins, 2008; Jenkins, 2014).

Environmental storytelling, as analyzed by Fernández-Vara (2011), uses spatial design to communicate narrative: a bloodstain on the floor, a child's abandoned toy, architecture that reveals social hierarchies. The player becomes detective and archaeologist, piecing together history from spatial evidence. This technique is particularly relevant for procedural generation—embedding narrative into generated spaces through consistent environmental logic rather than explicit exposition (Fernández-Vara, 2011).

Carson (2000) articulated the Disney approach to environmental storytelling in theme park design: spaces that tell stories through architecture, props, and progression. The guest experience is authored spatially rather than temporally. Games like *Dark Souls* and *Elden Ring* exemplify this approach—lore embedded in item descriptions, architecture that reveals history, placement of enemies that tells stories without cutscenes. Procedural generation can support this if generation rules encode narrative logic: ruins contain evidence of their former purpose; creature placement reflects ecosystem relationships; treasure locations reward exploration.

These theoretical frameworks impose constraints on procedural world generation. Generated content must maintain internal consistency (Tolkien's subcreation principle). It must reward exploration with discoverable meaning (Murray's encyclopedic property). It must embed narrative in spatial details (environmental storytelling). And it must leave intentional gaps for player imagination (Jenkins's negative capability). AI systems that generate content without this theoretical grounding risk producing worlds that feel hollow despite visual richness.

The tension between procedural generation and authored worlds is not opposition but collaboration. The goal is not worlds that feel procedurally generated but worlds that feel intentionally crafted—even when machines produced them. This requires generation systems that understand design intent, not just statistical patterns. The theoretical frameworks above provide the vocabulary for specifying that intent.

---

8. The Frontier: Generative Worlds

8.1 World Models and Dream Training

The ultimate expression of AI-assisted worldbuilding is the fully generative game environment—worlds that exist not as authored assets but as outputs of neural networks. The theoretical foundation is the "world model": a learned representation of environment dynamics that can predict future states given actions.

Ha and Schmidhuber's 2018 "World Models" paper demonstrated the concept: train a variational autoencoder to compress observations, a recurrent network to predict future states, and a controller to select actions. The crucial insight was that agents could train entirely in the model's "dreams"—hallucinated rollouts of environment dynamics—then transfer learned policies to reality (Ha & Schmidhuber, 2018).

DreamerV3 (2023) scaled this approach to complex environments, achieving diamond collection in Minecraft from scratch with a single configuration working across 150+

diverse tasks. The system learns to imagine coherent multi-step trajectories, optimizing policies against these imagined futures (Hafner et al., 2023).

8.2 Neural Game Engines

GameGAN (2020) demonstrated that GANs could learn game rules from observation alone—recreating Pac-Man as a pure neural simulation without game code (Kim et al., 2020). GameNGen (2024) extended this to complex 3D games, running DOOM at 20 fps on TPU as a diffusion model—each frame generated anew based on previous frames and player inputs (Valevski et al., 2024).

Genie (Bruce et al., 2024) represents the paradigm shift to foundation world models. An 11-billion parameter model trained unsupervised on internet video, Genie can generate playable 2D environments from single images, text descriptions, or sketches. The model infers latent actions without explicit labels, learning general world dynamics that transfer across visual domains. This is not just PCG—it is learned physics, aesthetics, and interactivity in a single generative model (Bruce et al., 2024).

Oasis (Decart, 2024) brought these capabilities to real-time performance, generating Minecraft gameplay at 20 fps—100× faster than Sora. The architecture combines a spatial autoencoder with diffusion transformers, achieving interactive frame rates that make neural world generation viable for actual play rather than just research demonstration (Decart, 2024).

8.3 Limitations and Trajectory

Current systems have significant limitations. Oasis “forgets” world layouts—walking in a circle may not return to the same location. Resolution caps at 720–1080p. Complex action spaces (an RPG’s inventory, crafting, and dialogue systems) exceed current control fidelity. Training requires substantial gameplay footage that may not exist for novel game concepts.

Yet the trajectory is unmistakable. From World Models (2018) to Genie (2024), each year brings larger models, faster inference, and broader capability. The question is not whether neural networks can generate interactive worlds—GameNGen and Oasis prove they can—but when they become practical for production use. For this thesis’s central question about indie RPG feasibility, the answer appears to be: not yet, but perhaps within 2–3 years.

The intermediate path—using generative AI for content creation while maintaining traditional game architecture for runtime—is already viable. Neural networks generate assets, dialogues, and level layouts that export to conventional game engines. The frontier research points to where this is heading, but today's practical approach combines neural generation with traditional game development: AI for authoring, engines for playing.

Interestingly, the research trajectory suggests eventual convergence. As neural world models improve, the distinction between "AI generates content for a game" and "AI is the game" blurs. Oasis already renders frames in response to inputs; Genie generates playable environments from images. The game engine of 2030 may be closer to a foundation world model than to Unreal or Unity. For independent developers, this trajectory suggests investing in AI literacy and creative direction skills rather than traditional engine expertise—the tools will continue to abstract away implementation detail.

---

9. Synthesis

9.1 Cross-Domain Integration

The fifteen domains reviewed here do not exist in isolation. They form a pipeline from conception to playable game, each stage enabled by AI capabilities that barely existed five years ago.

The pipeline begins with **foundations** (Domain 1): transformer architectures enabling language understanding and generation; diffusion models enabling visual synthesis. These foundations enable **procedural generation** (Domains 2a, 2b): classical algorithms enhanced by machine learning, generating terrain, levels, and structures that respect learned design patterns. **Visual content generation** (Domains 3, 4) produces the assets populating these worlds—concept art, textures, and increasingly 3D models from text descriptions.

Intelligent characters (Domains 5, 8c, 8d) animate these worlds. Beyond basic behavior, NPCs now exhibit memory, personality, and emergent social behavior through LLM-powered generative agents. Quest and dialogue generation (Section 5.3) addresses narrative content at scale, while multi-agent development systems (Section 5.4) suggest AI-assisted workflows where developer time shifts from content creation to creative direction.

The **production pipeline** (Domains 8a, 8b, 8e, 8f, 8g) integrates these capabilities—auto-rigging AI-generated characters, generating motion from text descriptions, synthesizing voice and facial animation, constructing worlds procedurally in Houdini, and rendering in Unreal Engine 5. The integration challenge (Section 6.4) and quality assurance requirements (Section 6.5) highlight that orchestration and curation, not individual capabilities, now limit what independent developers can achieve.

Worldbuilding theory (Domain 7) ensures coherent secondary worlds rather than arbitrary asset collections—Tolkien's consistency, Murray's encyclopedic depth, and Jenkins's negative capability constraining what generated content must achieve. And the **generative frontier** (Domain 6) points toward neural networks that encode entire interactive environments, from World Models' learned dynamics to Genie's playable environments from single images.

9.2 Answering the Research Question

Can AI and procedural tools restore the creator-to-creation ratio that existed before games went 3D? The evidence from 206 papers across fifteen domains suggests a qualified yes—with important caveats about what "restore" means and when full capability arrives.

The strongest evidence comes from production-ready capabilities:

- **Animation**: Motion Matching is native to UE5.4; text-to-motion generates clips in minutes that would require hours or expensive motion capture
- **Audio**: MusicGen, AudioLDM, and XTTS are open-source and produce professional-quality music, effects, and voice
- **Dialogue**: LLM-powered NPCs with memory (Convai, Generative Agents architectures) can generate contextual conversation exceeding what manual authoring achieves at scale
- **Level generation**: PCGML techniques produce playable content from limited example data
- **Engine integration**: UE5's PCG Framework, ML Deformer, and NNE enable AI-assisted content in production engines

The weaker evidence concerns capabilities that are emerging but not yet production-ready:

- **3D asset generation**: Text-to-3D produces results requiring cleanup; fully automated 3D character pipelines remain 1–2 years out
- **Neural world generation**: Impressive demonstrations (DOOM, Minecraft) but not yet practical for novel games
- **Quality consistency**: Generated content requires human curation; fully autonomous pipelines are not yet reliable

9.3 The Independent Developer Path

For the thesis's specific context—creating a Skyrim-scale open-world RPG as a solo or small-team project—the analysis suggests a viable path using 2024–2025 tools:

Immediately practical (production-ready):

- Concept art via Stable Diffusion with LoRA/ControlNet for style consistency
- NPC dialogue through LLM integration (Convai or custom Generative Agents)
- Background music via MusicGen, sound effects via AudioLDM
- Character animation via UE5 Motion Matching with Mixamo or text-to-motion libraries
- World layout via Houdini procedural terrain exported to UE5 PCG Framework

Near-term practical (6–12 months):

- 3D character generation with HumanRig auto-rigging pipeline
- Voice synthesis at scale with XTTS and Audio2Face lip-sync
- Quest generation with knowledge-graph-constrained LLM approaches

Emerging (1–2 years):

- Text-to-3D at production quality without manual cleanup
- Multi-agent development workflows (ChatDev-style role simulation)
- Neural terrain enhancement replacing Perlin noise

The creator-to-creation ratio is being restored, but unevenly. A solo developer in 2025 can produce more content than a 50-person team in 1995—but producing *Skyrim*-quality content still requires either compromises (stylized rather than photorealistic art) or additional capability maturation. The tools exist; mastering their integration is the remaining challenge.

9.4 Research Gaps and Future Work

Several gaps emerged from this review warranting further investigation:

1. **Integration architecture**: While individual AI capabilities are well-documented, systematic frameworks for composing them into production pipelines are lacking. How should text-to-3D, auto-rigging, and motion retargeting chain together? What quality gates and human checkpoints are needed? The literature describes components; constructing functioning workflows remains practitioner knowledge passed through tutorials and Discord servers rather than academic publication.

2. **Consistency maintenance**: Worldbuilding theory emphasizes coherence, but procedural techniques generate content without global awareness. How can generated quests respect established lore? How can NPC memories stay consistent with world state? Preliminary work on knowledge-graph-constrained generation addresses this, but comprehensive solutions that scale to open-world complexity do not yet exist. This is perhaps the most critical gap for applying current AI to RPG development.

3. **Evaluation metrics**: Academic benchmarks (FID scores, perplexity) do not capture game-relevant quality. What constitutes "good enough" for an indie RPG? How do player experience and production efficiency trade off? The game industry lacks standardized quality metrics for AI-generated content, making it difficult to compare approaches or set production targets. Research on perceived quality of generated assets versus hand-crafted equivalents would be valuable.

4. **Economic modeling**: While this review assessed technical feasibility, economic feasibility—compute costs, training data requirements, development time—requires empirical investigation with actual projects. How many GPU-hours does it cost to generate a village's worth of NPCs? How much human curation time does AI output require? These questions have answers only in unpublished studio practice.

5. **Skill transition**: What skills do developers need to effectively use AI-assisted pipelines? Traditional game development education emphasizes implementation (programming,

3D modeling, animation). AI-assisted development may emphasize different competencies: prompt engineering, curation, direction, and taste. No research maps this skill transition or proposes curricula.

6. **Legal and ethical dimensions**: Generated content raises unresolved questions about training data rights, output ownership, and disclosure obligations. How should developers attribute AI-generated assets? What disclosure do players expect? The legal landscape is evolving faster than research can document.

These gaps define the territory for the thesis's subsequent empirical work: building the AI-assisted worldbuilding pipeline this literature review describes, and measuring what it actually takes to create expansive game worlds with contemporary tools. The 206 papers analyzed here establish what is technically possible; the remaining work is discovering what is practically achievable.

References

Note: Full citations are provided in the domain-specific BibTeX files in `data/exports/`. Key foundational works cited above include:

Ammanabrolu, P., et al. (2020). Graph constrained reinforcement learning for natural language action spaces. *ICLR 2020*.

Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM, 37*(7), 122–125.

Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*.

Bruce, J., Dennis, M., Edwards, A., et al. (2024). Genie: Generative interactive environments. *arXiv preprint arXiv:2402.15391*.

Chen, M., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Clavet, S. (2016). Motion matching and the road to next-gen animation. *Game Developers Conference 2016*.

Copet, J., et al. (2023). Simple and controllable music generation. *Advances in Neural Information Processing Systems, 36*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT 2019*.

Dormans, J., & Bakkes, S. (2011). Procedural adventure game generation: A design agent approach. *PCG Workshop, FDG 2011*.

Fernández-Vara, C. (2011). Game spaces speak volumes: Indexical storytelling. *DiGRA Conference 2011*.

Gal, R., et al. (2022). An image is worth one word: Personalizing text-to-image generation using textual inversion. *ICLR 2023*.

Gao, J., et al. (2022). GET3D: A generative model of high quality 3D textured shapes learned from images. *NeurIPS 2022*.

Goodfellow, I., et al. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems, 27*.

Guo, C., et al. (2024). MoMask: Generative masked modeling of 3D human motions. *CVPR 2024*.

Ha, D., & Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

Hafner, D., et al. (2023). Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems, 33*.

Holden, D., Saito, J., & Komura, T. (2016). A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics, 35*(4).

Holden, D., et al. (2020). Learned motion matching. *ACM Transactions on Graphics, 39*(4).

Hong, S., et al. (2023). MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Hu, E., et al. (2021). LoRA: Low-rank adaptation of large language models. *ICLR 2022*.

Jenkins, H. (2008). *Convergence Culture: Where Old and New Media Collide*. NYU Press.

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics, 42*(4).

Khalifa, A., et al. (2020). PCGRL: Procedural content generation via reinforcement learning. *AIIDE 2020*.

Kim, S., et al. (2020). Learning to simulate dynamic environments with world models. *CVPR 2020*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*.

Kybartas, B., & Bidarra, R. (2017). A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games, 9*(3).

Lin, C. H., Gao, J., Tang, L., et al. (2023). Magic3D: High-resolution text-to-3D content creation. *CVPR 2023*.

Liu, H., et al. (2023). AudioLDM: Text-to-audio generation with latent diffusion models. *ICML 2023*.

Mateas, M., & Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. *Game Developers Conference 2003*.

Mildenhall, B., Srinivasan, P. P., Tancik, M., et al. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. *ECCV 2020*.

Murray, J. H. (1997). *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. MIT Press.

Park, J. S., O'Brien, J. C., Cai, C. J., et al. (2023). Generative agents: Interactive simulacra of human behavior. *ACM UIST 2023*.

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics, 19*(3), 287–296.

Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2022). DreamFusion: Text-to-3D using 2D diffusion. *ICLR 2023*.

Prusinkiewicz, P., & Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer.

Qian, C., et al. (2023). Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Radford, A., et al. (2021). Learning transferable visual models from natural language supervision. *ICML 2021*.

Rombach, R., Blattmann, A., Lorenz, D., et al. (2022). High-resolution image synthesis with latent diffusion models. *CVPR 2022*.

Ruiz, N., et al. (2022). DreamBooth: Fine-tuning text-to-image diffusion models for subject-driven generation. *CVPR 2023*.

Shaker, N., Togelius, J., & Nelson, M. J. (2016). *Procedural Content Generation in Games*. Springer.

Summerville, A., Snodgrass, S., Guzdial, M., et al. (2018). Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games, 10*(3).

Tevet, G., et al. (2023). Human motion diffusion model. *ICLR 2023*.

Togelius, J., et al. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(3).

Tolkien, J. R. R. (1947). On fairy-stories. In *Essays Presented to Charles Williams*.

Valevski, D., et al. (2024). Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.

Volz, V., et al. (2018). Evolving Mario levels in the latent space of a deep convolutional generative adversarial network. *GECCO 2018*.

Wei, J., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS 2022*.

Wolf, M. J. P. (2012). *Building Imaginary Worlds: The Theory and History of Subcreation*. Routledge.

Xu, Y., et al. (2020). RigNet: Neural rigging for articulated characters. *ACM Transactions on Graphics, 39*(4).

Ye, H., et al. (2023). IP-Adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*.

Zhang, L., Rao, A., & Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. *ICCV 2023*.