Proceedings of the Twentieth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2024)

Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration

Mahdi Farrokhi Maleki, Richard Zhao

Department of Computer Science, University of Calgary
Calgary, Alberta, Canada, T2N 1N4
mahdi.farrokhimaleki@ucalgary.ca, richard.zhao1@ucalgary.ca

Abstract

Procedural Content Generation (PCG) is defined as the automatic creation of game content using algorithms. PCG has a long history in both the game industry and the academic world. It can increase player engagement and ease the work of game designers. While recent advances in deep learning approaches in PCG have enabled researchers and practitioners to create more sophisticated content, it is the arrival of Large Language Models (LLMs) that truly disrupted the trajectory of PCG advancement.

This survey explores the differences between various algorithms used for PCG, including search-based methods, machine learning-based methods, other frequently used methods (e.g., noise functions), and the newcomer, LLMs. We also provide a detailed discussion on combined methods. Furthermore, we compare these methods based on the type of content they generate and the publication dates of their respective papers. Finally, we identify gaps in the existing academic work and suggest possible directions for future research.

Introduction

The video game industry has been expanding rapidly and even surpassed the combined revenue of the music and movie industries in 2022 (Forbes Magazine 2023). This huge market means there is always a need for new content. However, the process of creating a game is very time-consuming and can take several years (Juego Studio 2023). Procedural Content Generation (PCG) is considered one of the solutions to this problem and can increase replay value, reduce production costs, and minimize effort (Amato 2017). PCG for games has existed since the 1980s, and it was mainly used in roguelike games such as *Beneath Apple Manor* (1978) and the genre's namesake, *Rogue* (A.I.Design 1980).

PCG can be used to create a variety of content, but it is commonly used to create art assets (Kang et al. 2020; Mittermueller, Ye, and Hlavacs 2022), maps and levels (Kreitzer, Ashlock, and Pereira 2019; Kumaran, Mott, and Lester 2019), game mechanics (Machado et al. 2019), and music for games (Makhmutov 2019). The algorithms used can greatly vary depending on the content they are supposed to generate, but we can generally categorize them under a few categories. Most of the algorithms reviewed in this survey fall under one or a combination of three categories: (1) search-based methods, such as Monte Carlo Tree Search (MCTS), which focus mainly on optimizations, (2) learning-based methods, including traditional machine learning and deep learning (DL), such as generative adversarial networks (GAN) and reinforcement learning (RL), which are the recent additions to PCG, and (3) other methods, such as noise functions and generative grammars, that we could not categorize in the earlier categories, Because of the massive interest in DL in recent years, there are many papers published that use DL algorithms as part of PCG. The newest player on the team, Large Language Models (LLMs), comes with very unique characteristics that we will describe in detail.

Related Works

There are a handful of surveys on PCG for games with different focuses and aims that have been published before our work. Some of them discuss the technical details of PCG algorithms (Zhang, Zhang, and Huang 2022), while others focus on content created by PCG (Hendrikx et al. 2013; De Carli et al. 2011). Some papers focus on specific types of PCG; for example, machine learning in PCG, while other works (Summerville et al. 2018) mainly focus on DL algorithms in PCG. Liu et al. (2021) specifically examines puzzle generation using PCG. Two textbooks, one for PCG (Shaker, Togelius, and Nelson 2016) and one for Game AI (Yannakakis and Togelius 2018), cover search-based methods, solver-based methods, constructive generation methods (such as cellular automata and grammar-based methods), fractals, noise, and ad-hoc methods for generating diverse game content.

As of the writing of this paper, the most recent survey on PCG was published in 2022 (Zhang, Zhang, and Huang 2022). This paper includes different categories of PCG; however, it does not discuss generated content in detail nor mention gaps in the field or possible future directions. Based on this information, it is clear that there is a lack of a comprehensive review of PCG for games in recent years. Since the publication of Liu et al. (2021), PCG has grown quickly, and a significant number of papers and articles, especially those discussing DL and LLMs, have been published. New research directions and possibilities have emerged in ways that significantly changed the research field. A review of the

state-of-the-art and the latest applications of DL and LLMs to PCG is needed.

Review Structure

The structure of the paper is as follows:

First, we give an overview of the different types of content that can be generated using PCG. Then, we go through each category and explain the most commonly used methods published in academic articles. While some cutting-edge deep learning methods are applied on their own, others are applied in combination with more traditional methods, or in an interactive setting (Liu et al. 2021). Because of this, we added another category called combined methods. The reason behind this choice is that we saw a new trend in combining different algorithms (especially GANs, RL, and LLMs), and we believe that this new category could shape the future of PCG for games. Following this, using the data we gathered by analyzing recently published papers, we discuss the research trends, areas of focus in PCG, gaps in academic research, and possible solutions. We end the paper with an outlook on possible future directions and a conclusion.

For reviewing the mentioned articles, we had several criteria for inclusion and exclusion, which we explain below:

1. To ensure a deeper focus on recently published papers, we selected related papers on PCG for games published in one of five different well-known conference series: the Artificial Intelligence and Interactive Digital Entertainment (AIIDE) conference series, the International Conference on the Foundations of Digital Games (FDG) series, the Advances in Computer Games (ACG) conference series, the Computer-Human Interaction in Play (CHI-PLAY) conference series, and the IEEE Conference on Games (CoG) series, during the most recent five years (2019-2023). Based on these criteria, we obtained 207 articles aggregated in Figure 1. The distribution of selected papers is as follows: FDG: 40%, CoG: 33%, AIIDE: 24%, CHI PLAY: 2%, ACG: 1%.

   For works before this period, we only discuss influential or interesting papers with a high citation count in the mentioned survey. For the latter, we used Google Scholar for academic papers and Google for grey literature. The phrase used in Google Scholar to find these articles was ('Procedural Content Generation' OR PCG) AND (game OR games), and the phrase used in Google was 'Procedural Content Generation for games'.
2. We only included papers focusing on generating content, so articles similar to Osborn et al. (2019), which is about evaluating the generated content and not generating it, are not part of this survey.
3. We excluded research that focused solely on the behaviors of non-player characters (NPCs). While some behavior generation techniques may overlap with PCG, we feel that NPC behaviors are a separate topic that deserves its own analysis.
4. While we focused on papers that talked about content generation in video games, we also mentioned articles that used PCG in tabletop games and board games.
5. It is important to note that content generation has uses outside of designing and developing games for humans to experience. In addition to creating content in games meant for humans to play, PCG is also being utilized in scientific research to create game-like benchmarks and playgrounds for reinforcement learning and other forms of AI (Liu et al. 2021). PCG is also being used in other formats, such as movies (Massive software 2024) and art (Aminian, Parsa 2023). In this paper, we only focus on game content generation.

Novelty of the Work

The differences between our work and previous surveys are that (i) we conduct a comprehensive review of both PCG methods and targeted content, (ii) at the time of writing, we are the first review paper to include LLMs as part of the methods used in PCG, and describe why LLMs are unique from all previous approaches, (iii) we add a section for combined methods and discuss why this category is important, and (iv) by analyzing papers published in the most recent 5 years in different conferences, we create a timeline for the methods used and find the current trend in academic research.

Content Types

Almost everything, from sounds to the game narrative, can be generated nowadays, but the generated content for games can vary a lot depending on the algorithms used. For example, LLMs are mostly used to create game narratives (Huang and Sun 2023), and GANs are considered a better solution to generate images (Kang et al. 2020) and 2D levels (Abraham and Stephenson 2023).

There are several ways that we can categorize generated content. For example, it can be divided into online and offline generation. Online generation means that the content generation is performed during the runtime of the game, while offline generation means it is created during game development. Additionally, content can be categorized as necessary or optional (Togelius et al. 2011). For this survey, we use categories similar to the ones presented by Hendrikx et al. (2013) with some modifications. We divide content created for games into five different categories, each consisting of multiple items.

1. **Game bits:** This category consists of the smallest pieces (units) used in games. Any type of generated texture, sound, vegetation, structures and buildings, and object properties (e.g. can it interact with basic physics?) goes into this list. We also included art and images (Coutinho and Chaimowicz 2022) created by PCG algorithms in this category.
2. **Game Space:** Game space is the physical environment that the game takes place and is created from game bits. PCG algorithms are commonly used to generate maps and roads for games. Researchers have even tried to create entire game worlds with them (Prins et al. 2023).
3. **Game Scenarios:** Game scenarios are parts of the game that are tied to the narrative. This includes generated conversations, stories, and quests. We include puzzles and

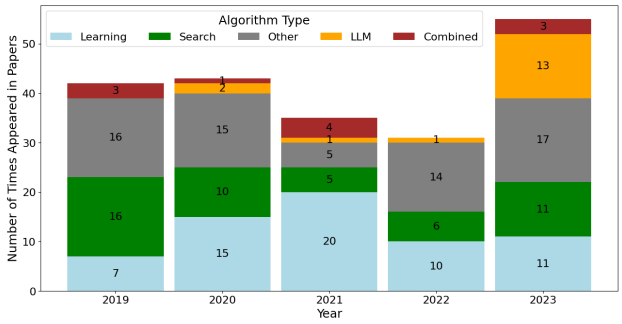

Figure 1: A timeline showing the types of algorithms appeared in PCG-related research papers during the most recent five years.

interactive level elements in game scenarios because they are part of the scenarios that contribute to the story. This is especially true in the case of generated levels for side-scrolling 2D games (Kumaran, Mott, and Lester 2019). Levels for music games, such as Guitar Hero or Dance Dance Revolution, can be seen as 2D levels as well (Liu et al. 2021).

4. **Game Design:** This category consists of any mechanics or rules created for games (what can the player do and what are the goals?) It also includes generated system design, which we interpret as systems used in the games. Generating spawn points for first-person shooter games for game design purposes (Ballabio and Loiacono 2019) is a perfect example of a generated design system.
5. **Derived content:** Derived content includes everything that is not essential to the game but can help the player better immerse in the game world. This category contains background NPC interactions, news found within the game, and chatter of different characters that are not part of the game's story or a quest.

PCG Algorithms

Due to the different types and roles of content in games, diverse PCG methods have been adapted for procedural content generation. In this section, we present different algorithms that exist and can be used to generate items. We categorized them into five main categories: Search-based, Machine Learning-based, Other, LLMs, and Combined Methods. The category called Combined Methods includes papers that use several methods in their study, either in parallel or in one integrated system.

It is worth mentioning that there were a few methods that we could not place under specific subgroups, so we decided to mention them at the start of each category.

Search-Based Methods

The term 'search-based PCG' was coined by Togelius in his paper on the taxonomy of PCG (Togelius et al. 2010). A search-based PCG refers to a special case of generate-and-test PCG. A generate-and-test PCG does not directly dish out content it generates but instead tests the content first using a test function. Depending on the test result, the PCG can either accept the content or reject it and create new content.

A search-based PCG is a test-and-generate PCG that satisfies two criteria (Prasetya and Maulidevi 2016). First, instead of simply accepting or rejecting content, the test function of a search-based PCG assigns a real value that measures the acceptability of the content. This value is often called fitness, and the function that produces it is called a fitness function. Second, in the creation of new, better content, a search-based PCG uses the previously rejected content as the creation base, and the new content is a slightly modified version of the old content. Search-based methods have been used to generate a variety of content, such as puzzles, race tracks, levels, terrains, and maps (Togelius et al. 2010; Prasetya and Maulidevi 2016).

In many cases, search-based algorithms use some form of evolutionary algorithm as the main search mechanism, as evolutionary computation has so far been the method of choice among search-based PCG practitioners. However, search-based PCG does not need to be married to evolutionary computation; other heuristic and stochastic search mechanisms are viable as well (Togelius et al. 2010). In this section, we discuss evolutionary algorithms, Wavefunction Collapse (WFC), Monte Carlo Tree Search (MCTS), simulated annealing, and particle swarm optimization.

1. **Evolutionary Algorithm**: In an evolutionary algorithm, a population of candidate content instances is held in memory. Each generation, these candidates are evaluated by the evaluation function and ranked. The worst candidates are discarded and replaced with copies of the good candidates, except that the copies have been randomly modified (i.e., mutated) and/or recombined (Togelius et al. 2010). One of the most famous examples of evolutionary algorithms is genetic algorithms. Genetic algorithms are used in many instances (Baek et al. 2022; Mitsis et al. 2020; Drageset et al. 2019) to create playable levels and game bits. This algorithm can also be used to create more traditional types of games. For example, Botea and Bulitko (2023) use this algorithm to create Romanian crossword puzzles. Other evolutionary algorithms are also used to create soundtracks for games (Makhmutov 2019), maps and game bits (Makhmutov 2019), roads (Song and Whitehead 2019), and levels for board games (Gerhold and Tijben 2023).
2. **Planning Algorithms**: Planning, in general, is a problem-solving technique consisting of a planning problem (i.e., initial state and goal specification) and a planning domain (i.e., objects, predicates, and action operators). Given the input of a planning problem, a sound planner produces a solution or a plan, which is a sequence of actions that achieve all the specified goal conditions without any causal threats (Song et al. 2020). In recent years, planning-based algorithms have often been used to generate stories in games because planning-based narrative generation is effective at producing stories with a logically sound flow of events (Siler 2022).
3. **Wave Function Collapse (WFC)**: WFC is a texture

synthesis algorithm. Compared to earlier texture synthesis algorithms, WFC guarantees that the output contains only those NxN patterns that are present in the input. This makes WFC perfect for level generation in games and pixel art, and less suited for large full-color textures (Efros and Leung 1999). The WFC algorithm also supports constraints, allowing it to be easily combined with other generative algorithms or manual creation (Gumin 2021).

4. **Monte Carlo Tree Search (MCTS)**: Monte Carlo Tree Search (MCTS) is a heuristic search algorithm that involves searching combinatorial spaces represented by trees. In such trees, nodes denote states whereas edges denote transitions (actions) from one state to another (Świechowski et al. 2023).
5. **Simulated Annealing:** Simulated annealing is a process where a setup is randomly tweaked and compared to the previous setup using the cost function. If it is better, the new setup is kept. Otherwise, the decision to keep the new setup is made randomly, with the probability of keeping the old setup proportional to how much better it is (Russell and Norvig 2016).
6. **Particle Swarm Optimization (PSO)**: PSO is a general-purpose optimization technique developed by Eberhart and Kennedy (1995). This technique was inspired by the concept of swarms in nature, such as bird flocking, fish schooling, or insect swarming. The idea is that individual members of the swarm can profit from the discoveries and previous experiences of all other members of the swarm during the search for the optimum solution (Duro and de Oliveira 2008).

Machine Learning-Based Methods

Machine learning methods have gained a lot of popularity during the last decade (Summerville et al. 2018). Research on neural networks under the name deep learning has precipitated a massive increase in the capabilities and application of methods for learning models from big data (Schmidhuber 2015; Goodfellow, Bengio, and Courville 2016). They have provided us with new ways for generating audio, images, 3D objects, network layouts, and other content types (Goodfellow et al. 2014) across a range of domains, including games. For example, long short-term memory (LSTMs) are mostly used for time-dependent sequential data (e.g., action sequences, agent paths, charts for rhythm) and language models (Risi and Togelius 2020), while generative adversarial networks (GANs) have been applied to generate artifacts, such as images, music, and speech (Goodfellow et al. 2014). Outside of content generation, machine learning algorithms have been widely used in testing generated maps using other algorithms (Liu et al. 2021). Many other machine learning methods can also be utilized in a generative role, including n-grams, Markov models, autoencoders, and others (Boulanger-Lewandowski, Bengio, and Vincent 2012; Fine, Singer, and Tishby 1998; Gregor et al. 2015; Summerville et al. 2018).

In this section, we talk about different machine learning methods such as Autoencoders (including deep variational autoencoders), Recurrent Neural Networks (including LSTMs), Generative Adversarial Networks (GANs), Markov Models, Reinforcement Learning methods, and Transformers.

The most recent entry, LLMs, are gaining a lot of attention among researchers. While state-of-the-art LLMs use transformers as their underlying architecture, their model-as-a-service (MaaS) nature puts them into a different category.

It is worth mentioning that not all articles regarding deep learning methods use complex neural network models. Merino et al. (2023) use neural networks to create sprites and game maps for 2D games. Bhaumik et al. (2023) and Kumaran et al. (2019) use neural network models to generate maps and levels for 2D games.

1. **Simple Neural Networks**: Some works on PCG algorithms rely on the use of relatively simple neural networks. For example, papers such as Chen etl al. (2020), take images as inputs and output game levels using neural networks.
2. **Recurrent Neural Networks & LSTMs**: A recurrent neural network (RNN) is a type of artificial neural network that uses sequential data or time series data. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output (Medsker, Jain et al. 2001). LSTMs are similar neural networks introduced to eliminate the vanishing gradient problem in RNNs. LSTMs work to solve that problem by introducing additional nodes that act as a memory mechanism, telling the network when to remember and when to forget. As mentioned before, LSTMs and RNNs are mostly used to create sequential data. Summerville and Mateas (2016) deals with Mario levels as a sequence of data and combines Markov Chains and LSTMs to create new levels.
3. **Generative Adversarial Networks (GANs)**: GANs are very popular for content generation purposes. They usually consist of two networks, a generator and a discriminator, that are trained iteratively to allow the generator to create more realistic content while the discriminator gets better at distinguishing generated content from real data (Goodfellow et al. 2014; Liu et al. 2021).

   GANs are perfect for generating content represented by pixel-based images or 2D arrays of tiles, such as levels, maps, landscapes, and sprites. By reviewing the gathered papers, it can be seen that in the work using GANs for level generation, game levels are tackled as images only during training, while the constraints for validating levels are not considered at all.
4. **Markov Models**: The Markov chain algorithm is a typical constructive method. In this approach, content is generated on-the-fly (Kernighan and Pike 2006) according to the conditional probabilities of the next state in a sequence based on the current state. This state can incorporate multiple previous states via the construction of n-grams. An n-gram model (or n-gram for short) is simply an n-dimensional array where the probability of each state is determined by the n states that precede it (Summerville et al. 2018). This is the simplest form of Markov

chains, also called 1D Markov chains.

There are also multidimensional Markov chains (MdMCs) (Ching, Zhang, and Ng 2007), where the state represents a surrounding neighborhood and not just a single linear dimension. A multidimensional Markov chain differs from a standard Markov chain in that it allows for dependencies in multiple directions and from multiple states, whereas a standard Markov chain only allows for dependence on the previous state alone (Summerville et al. 2018). In addition to their standard MdMC approach, Snodgrass and Ontañón developed a Markov random field (MRF) approach (Goodfellow et al. 2014) that performed better than the standard MdMC model in Kid Icarus, a domain where platform placement is pivotal to playability (Snodgrass and Ontanón 2016). This method has generated novel but recognizable game names (Browne 2008), natural language conversations, poetry, jazz improvisation (Pachet 2004), and content in a variety of other creative domains (Bentley, Kumar et al. 1999).

5. **Autoencoders**: An autoencoder is a type of neural network architecture designed to efficiently compress (encode) input data down to its essential features, then reconstruct (decode) the original input from this compressed representation. One of the famous autoencoders used for PCG is variational autoencoders (VAEs). VAEs are generative models that learn compressed representations of their training data as probability distributions, which are used to generate new sample data by creating variations of those learned representations (Ng et al. 2011). These algorithms are mostly used for generating 2D maps and levels.

   Snodgrass and Sarkar (2020) use VAEs to generate level structures and a search-based approach to blend details from various platformers, while Sarkar et al. (Sarkar et al. 2020) directly trains VAEs on levels from several platforming games and interpolates the latent vectors between domains for blending (Liu et al. 2021). Sometimes, trained autoencoders may be used to repair unplayable levels. Davoodi et al. (2022) train an autoencoder to repair manually designed levels for different games by re-iterating it over the decoder while using a trained discriminator from a GAN model to determine the stopping criteria (Liu et al. 2021).

6. **Reinforcement Learning**: RL problems involve learning how to map situations to actions that maximize a numerical reward signal (Sutton and Barto 2018). Most articles that use RL develop agents to play generated levels, which indirectly serve as content evaluators.

   To generate content using RL, the generation task is usually transformed into a Markov decision process (MDP), where a model is trained to iteratively select actions that would maximize expected future content quality. This transformation is not an easy task, and there is no standard way of handling it. Most RL PCG approaches require an adaptation of the input to be used during generation (Liu et al. 2021). There are also some interesting cases where RL can be used in the absence of data, provided a system for the learning agent to interact with can be set up (Barriga 2019).

7. **Transformers**: Transformers are a type of neural network architecture that transform an input sequence into an output sequence by learning context and tracking relationships between sequence components (Khan et al. 2022). They rely on a self-attention mechanism (Vaswani et al. 2017), which facilitates the capture of intricate semantic relationships within high-dimensional feature spaces. For example, PCGPT (Mohaghegh et al. 2023) used transformers to iteratively generate complex and diverse game maps in the Sokoban game.

Other Methods

There are some methods that do not belong in the previous categories. We added this list for frequently used methods in PCG that we could not fit in any other category. It includes pseudo-random number generators (PRNGs), generative grammars, generative graphs, and fractals.

It is worth mentioning that these categories are not the only means of generating content for games. There are some articles about innovative ways of PCG. For example, Wootton (2020a) uses quantum blur effects to create maps and levels for various games. Wootton (2020b) also uses quantum computation combined with graphs to generate maps.

(a) **Pseudo-random Number Generator (PRNG):** One of the simplest and earliest approaches to procedural game content generation is based on pseudo-random number generation (PRNG). PRNG is an algorithm for generating a sequence of numbers that approximates the properties of random numbers (Barker et al. 2007). A PRNG algorithm consists of three parts: the seed, which is the initial value, the formula, which converts the seed to output, and the distribution, which is the variance of the results (Matsumoto et al. 2006).

PRNG was first used as a data compression method because the generated sequence, while appearing random, can be reproduced if the same seed and algorithm are used. Combined with other methods, PRNG-based techniques can be used to generate buildings, textures, and items (Hendrikx et al. 2013). One of the most famous forms of PRNGs is noise functions. Perlin noise and other noise functions are commonly used for texture generation. Noise-generated textures can be mapped easily on complex objects, unlike raster 2D images. The implementation of noise is relatively simple and is present in many software shaders and hardware graphics cards, such as NVIDIA's (Hendrikx et al. 2013). One of the main drawbacks of this method is that images generated randomly pixel-by-pixel have no meaningful structure. Many procedural techniques address this issue by finding the balance between iterative generation and random generation.

(b) **Generative Grammars:** Generative grammars, stemming from Noam Chomsky's study of languages in

the 1960s (Hendrikx et al. 2013), are sequences of words (or "sentences") with rules regarding how and when to replace some words with other words. They can be used to create correct objects from elements encoded as letters/words. L-systems, split grammars, wall grammars, and shape grammars, which have been used for plant generation, linear map dungeon or story generation, and quest generation, are all part of generative grammars. The downside of using this method is that it is linear, so it cannot be used generate a non-linear story. The possible solution is using generative graphs, which we discuss in the following section.

(c) **Generative Graphs:** A graph is a set of vertices connected by edges. Generative graphs are used as a solution for the linearity problem of generative grammars. One well-known type of generative grammar is graph grammars. A graph grammar is a set of rules that modify a graph. It is a framework introduced decades ago (Pfaltz 1972; Ehrig, Pfender, and Schneider 1973).

One of the challenges of using graph grammars was the need for an expert to design the rules, but now, some researchers (Merrell 2023) have worked on creating the grammar automatically. The content generated using generative graphs is almost similar to that generated by generative grammars, only they do not have to be linear. One of the most famous products of graph grammars is SpeedTree (Ehrig, Pfender, and Schneider 1973), a well-known set of tools to generate trees in the entertainment industry (Columbia Metropolitan Magazine 2022).

(d) **Fractals:** The term "fractal" was coined and popularized by Benoit B. Mandelbrot (Mandelbrot 1985). It describes a broad set of shapes characterized by non-integer dimension or an interesting mismatch of dimension (Hausdorff dimension strictly exceeding topological dimension), and detail at all scales or self-similarity. Fractals are frequently used in procedural content generation because self-similarity seems to mimic natural processes such as erosion and plant growth. The subdivision method also maps well onto level of detail implementations, allowing an 'infinite' amount of detail to be included by recursively subdividing the detail shown as the viewpoint moves closer to the fractal object (Cristea and Liarokapis 2015).

Large Language Models

In the most recent year, there has been an explosion of research on the applications of LLMs, and PCG is no exception. We found 17 papers during these years using LLMs as part of their pipeline (with 3 other papers using combined methods that integrated LLMs). Referring back to Figure 1, we see that in 2023, the use of LLMs drastically increased compared to previous years (13 papers and 2 other papers with combined methods integrating LLMs). This coincides with the release of ChatGPT and its underlying model, GPT-3. These models are used to create narratives, NPC chatter, and even mechanics. A famous example is Dungeon 2, a text adventure game (Schrum, Volz, and Risi 2020). In this game, players can type in any command and the system can respond to it reasonably well, creating the first never-ending text adventure. The system is built on OpenAI's GPT-2 language model (Volz et al. 2018), which was further fine-tuned on a number of text adventure stories. 1001 Nights is also another project that uses GPT as one of the main mechanics of its gameplay (Sun et al. 2023). Language models are also used in role-playing board games as an assistant for the game masters (Zhu et al. 2023). LLMs can also be used to create game levels. SCENECRAFT (Kumaran et al. 2023b) is a framework that transforms high-level natural language instructions from authors into dynamic game scenes that include NPC interactions, dialogue, emotions, and gestures. The research by Todd et al. (2023), Nasir and Togelius (2023), and Sudhakaran et al. (2024) also focus on using LLMs to create levels for games. For example, the latter uses MarioGPT, a fine-tuned GPT-2 model designed to generate Super Mario Bros levels based on textual prompts.

While current LLMs use transformers as their underlying model, with the publication of GPT-3, a new form of service has emerged. "Model as a service" (MaaS) involves deploying a model on a cloud-based infrastructure and offering its functionalities through APIs or web interfaces. These commercialized models, such as GPT-3, are often no longer open for researchers to explore and study their underlying architecture, creating a black box that is only known to select few. Our review shows that research published on using GPT-3 and its successors is no longer about training a new model, but about best ways to use an existing pre-trained model that is restricted by a private entity in many ways. While open source LLMs do exist, in our review, the vast majority of LLM usage was still with GPT-based LLMs.

Combined Methods

So far, we have discussed four different methods used for generating content in games. However, these methods can be combined. Many researchers have worked on using evolutionary algorithms with machine learning methods. One of the most popular examples of combining methods is the Latent Variable approach (Bontrager et al. 2018), which combines unsupervised learning in the form of a GAN or VAE with evolutionary computation to search for content in the learned space of a GAN/VAE. In the context of games, this approach has been employed to generate 2D game levels like Super Mario Bros and Zelda levels (Schrum, Volz, and Risi 2020; Volz et al. 2018). RL is also used in combination with many different algorithms. For instance, Kumaran et al. (2023a) creates game levels through a natural language interface and evaluates the levels using RL. Instead of relying solely on one method, combining different methods can be very effective in generating new content. RL models and search-based algorithms are effective tools to repair generated content (especially levels and structures) created by other algorithms (like GANs and LSTMs). More recently, combined methods integrating LLMs have also begun to emerge (Kumaran et al. 2023a; Volden, Grbic, and Burelli 2023). While PCG research tends to focus on using existing methods in innovative ways, these combined methods form a rather emerging approach in solving complex problems.

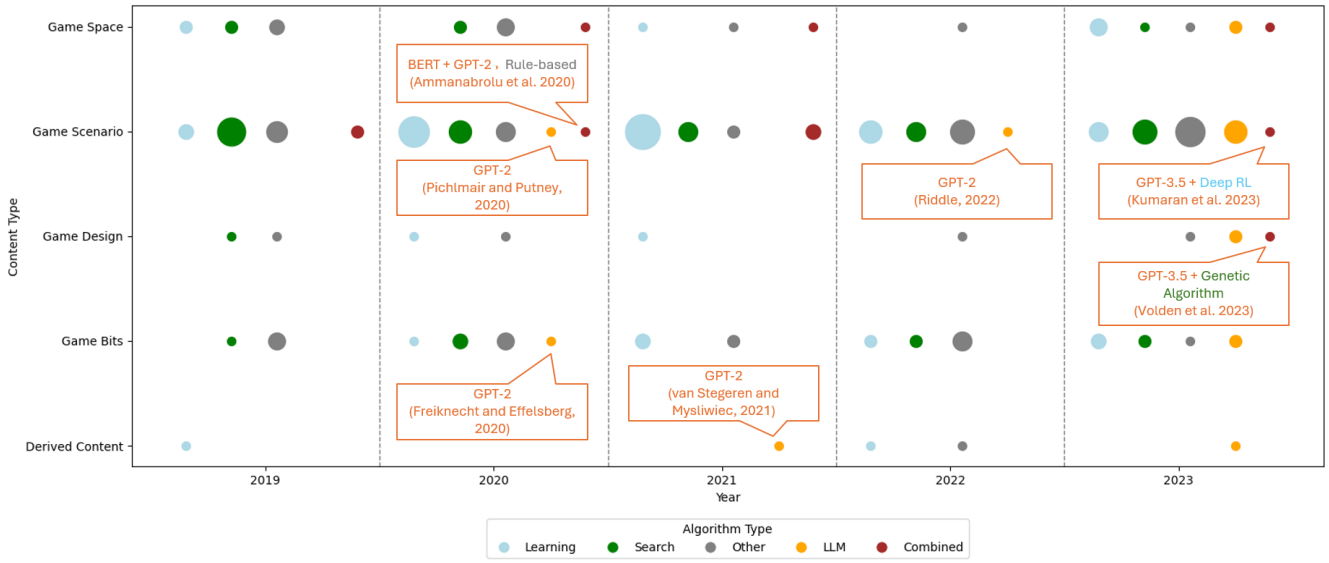

Figure 2: A timeline breaking down research published 2019-2023 in select game-related conferences on the topic of PCG, sorted by targeted content type. A few note-worthy LLM-based works are pointed out.

Analysis

In this section, we discuss the information obtained from reviewing selected papers. First, we talk about the trends in the papers. Then, we discuss the gaps in current works, and finally, we outline future research directions based on current trends.

Trends

Figure 2 shows an overview of 207 published papers using different methods for the period of 2019-2023, broken down by content type. Each row represents one type of content, such as the aforementioned Game Space. The dots represent papers in that particular year - the larger the dot, the more papers we found. The dots are color-coded by its algorithm type. We also gathered information about the 629 authors who worked on these papers which is shown in Figure 3 based on the number of co-authored papers.

By looking at Figure 2 which is extracted using collected articles, we notice that level generation is by far the most researched topic in PCG. There are 102 papers, almost half of the selected papers, focusing on level generation. Except for a few works such as Baek et al. (2022), most of these papers focus on 2D level generation. This seems reasonable as creating a 3D level is a far more complex task than creating a 2D level. We also found that 2D platformers are a popular genre in PCG, with *Super Mario Bros.* being the most used environment, featured in 18 papers.

From Figure 2, it is evident that machine learning-based algorithms have gained significant interest, and the trend of using them in PCG continues to grow. Liapis et al. (2020) mentions the consistent inclusion of vision papers and the increased use of machine learning in games, either on its own or in conjunction with PCG. Our data shows that the trend has not slowed down in recent years. The emergence of ChatGPT (OpenAI 2024) and the explosion of LLMs like BERT and GPT in recent years have also affected content generation in games. Many of the papers using LLMs were published in the last year. It is interesting to note that we found only five articles published before 2023 that used any type of LLMs, and all five used GPT-2 (which is an open-source model), with one also using BERT (another open-source model) (Riddle 2022; van Stegeren and Myśliwiec 2021; Ammanabrolu et al. 2020; Pichlmair and Putney 2020; Freiknecht and Effelsberg 2020). 2023 was the year when many LLM-based research emerged, but also when people moved on from open-source models to closed-source, mostly GPT-based services. Among the works, two caught our attention. These are combined methods with LLMs. Kumaran et al. (2023a) combined GPT-3.5 with the use of deep reinforcement learning for level generation and selection, while Volden et al. (2023) put together LLMs with a genetic algorithm to generate rules for difficulty adaptation in a serious game. We believe that these combined methods will show versatility in many situations, and this is an open research field with a lot of potential.

Gaps in Research

As mentioned, most of the research in level generation focuses on 2D level generation. 3D games are very popular in the games industry, and level generation for 3D games is a topic that needs further research and exploration.

Based on our findings, despite having innovative ideas in the selected papers, many do not follow up on those ideas, with some rare exceptions such as 1001 Nights (Sun et al. 2023). This gap between academic work and the games industry requires more discussion. One possible solution could

Figure 3: A word cloud of all authors in the published papers at the selected conferences during the 5 years. The size is proportionate to the number of co-authored papers.

be creating a playable demo of a game with generated content (if feasible). This approach could increase the chances of finding investors for the research. Of course, we cannot ignore the ethical considerations when using tools such as LLMs for generating game content (Melhart et al. 2023).

Possible Research Directions

Using the data we gathered, we believe that combined algorithms, especially deep learning algorithms and LLMs, will be a popular topic in the future. Instead of trying to generate whole items or levels, researchers could focus on repairing incorrect or improper content, reducing the effort needed to create a finished product.

LLMs are still fresh, and there is much unknown potential in them that can be used in PCG. They can serve as AI assistants for game designers, story and quest generators, and even level generators that produce levels as a sequence of words or sentences. Therefore, more research on the applications of LLMs could prove beneficial, especially open-source alternatives to closed-source commercial MaaS.

Conclusion

In this survey, we discussed different categories of PCG algorithms and the content that can be generated for games. We categorized PCG algorithms into five sections: search-based methods, machine learning-based methods, other methods, LLMs (MaaS), and combined methods. We also selected PCG related papers published between 2019-2023 and analyzed them to identify trends, gaps in current work, and possible future research directions.

The increasing use of LLMs for generative tasks is a recent development that was unheard of until about five years ago. Combining deep learning methods, such as GANs and RL, with other algorithms in content generation is another interesting trend that has been gaining more attraction recently. Both of these trends are built on advancements in deep learning, which have made machine learning methods effective for completely new classes of problems.

Although a variety of generated content types (e.g., levels, sound, maps, textures) have been investigated, each subcategory can be further explored to gain more insights. In-depth research on the best-suited algorithms for generating each type of content would also be beneficial. Additionally, a study dedicated to evaluation algorithms used to test generated content could be a valuable addition to this field.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant. We thank members of the Serious Games Research Group and the anonymous reviewers for their feedback.

References

Abraham, F.; and Stephenson, M. 2023. Utilizing generative adversarial networks for stable structure generation in angry birds. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 2–12.

A.I.Design. 1980. Rogue.

Amato, A. 2017. Procedural content generation in the game industry. *Game Dynamics: Best Practices in Procedural and Dynamic Game Content Generation*, 15–25.

Aminian, Parsa. 2023. Procedural Art Generation and Dynamic Content Creation in Games. https://www.artstation.com/blogs/pixunegroup/2l1O/procedural-

art-generation-and-dynamic-content-creation-in-games. Accessed: 2024-07-03.

Ammanabrolu, P.; Cheung, W.; Tu, D.; Broniec, W.; and Riedl, M. 2020. Bringing stories alive: Generating interactive fiction worlds. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 3–9.

Baek, I.-C.; Ha, T.-G.; Park, T.-H.; and Kim, K.-J. 2022. Toward Cooperative Level Generation in Multiplayer Games: A User Study in Overcooked! In *2022 IEEE Conference on Games (CoG)*, 276–283. IEEE.

Ballabio, M.; and Loiacono, D. 2019. Heuristics for placing the spawn points in multiplayer first person shooters. In *2019 IEEE Conference on Games (CoG)*, 1–8. IEEE.

Barker, E. B.; Barker, W. C.; Burr, W. E.; Polk, W. T.; and Smid, M. E. 2007. Sp 800-57. recommendation for key management, part 1: General (revised).

Barriga, N. A. 2019. A short introduction to procedural content generation algorithms for videogames. *International Journal on Artificial Intelligence Tools*, 28(02): 1930001.

Bentley, P. J.; Kumar, S.; et al. 1999. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *GECCO*, volume 99, 35–43.

Bhaumik, D.; Togelius, J.; Yannakakis, G. N.; and Khalifa, A. 2023. Lode enhancer: Level co-creation through scaling. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–8.

Bontrager, P.; Roy, A.; Togelius, J.; Memon, N.; and Ross, A. 2018. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 1–9. IEEE.

Botea, A.; and Bulitko, V. 2023. Generating and Solving Champion-Level Romanian Crosswords Puzzles. In *2023 IEEE Conference on Games (CoG)*, 1–4. IEEE.

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

Browne, C. B. 2008. *Automatic generation and evaluation of recombination games*. Ph.D. thesis, Queensland University of Technology.

Chen, E.; Sydora, C.; Burega, B.; Mahajan, A.; Abdullah, A.; Gallivan, M.; and Guzdial, M. 2020. Image-to-level: Generation and repair. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 189–195.

Ching, W.; Zhang, S.; and Ng, M. 2007. On multidimensional Markov chain models. *Pacific Journal of Optimization*, 3(2): 235–243.

Columbia Metropolitan Magazine. 2022. speedtree-takes-hollywood. https://shaggydev.com/2022/03/16/generative-grammars. Accessed: 2024-07-03.

Coutinho, F.; and Chaimowicz, L. 2022. On the challenges of generating pixel art character sprites using GANs. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 87–94.

Cristea, A.; and Liarokapis, F. 2015. Fractal nature-generating realistic terrains for games. In *2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, 1–8. IEEE.

Davoodi, O.; Ashtiani, M.; and Rajabi, M. 2022. An approach for the evaluation and correction of manually designed video game levels using deep neural networks. *The Computer Journal*, 65(3): 495–515.

De Carli, D. M.; Bevilacqua, F.; Pozzer, C. T.; and d'Ornellas, M. C. 2011. A survey of procedural content generation techniques suitable to game development. In *2011 Brazilian symposium on games and digital entertainment*, 26–35. IEEE.

Drageset, O.; Winands, M. H.; Gaina, R. D.; and Perez-Liebana, D. 2019. Optimising level generators for general video game AI. In *2019 IEEE conference on games (CoG)*, 1–8. IEEE.

Duro, J. A.; and de Oliveira, J. V. 2008. Particle swarm optimization applied to the chess game. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 3702–3709. IEEE.

Efros, A. A.; and Leung, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, 1033–1038. IEEE.

Ehrig, H.; Pfender, M.; and Schneider, H. J. 1973. Graph-grammars: An algebraic approach. In *14th Annual symposium on switching and automata theory (swat 1973)*, 167–180. IEEE.

Fine, S.; Singer, Y.; and Tishby, N. 1998. The hierarchical hidden Markov model: Analysis and applications. *Machine learning*, 32: 41–62.

Forbes Magazine. 2023. Council post: The Gaming Industry: A behemoth with unprecedented global reach. https://www.forbes.com/sites/forbesagencycouncil/2023/11/17/the-gaming-industry-a-behemoth-with-unprecedented-global-reach. Accessed: 2024-07-03.

Freiknecht, J.; and Effelsberg, W. 2020. Procedural generation of interactive stories using language models. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–8.

Gerhold, M.; and Tijben, K. 2023. Computer Aided Content Generation–A Gloomhaven Case Study. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–10.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; and Wierstra, D. 2015. Draw: A recurrent neural network for image generation. In *International conference on machine learning*, 1462–1471. PMLR.

Gumin, M. 2021. Wavefunctioncollapse: Bitmap and tilemap generation from a single example with the help of ideas from Quantum Mechanics. https://github.com/mxgmn/WaveFunctionCollapse. Accessed: 2024-07-03.

Hendrikx, M.; Meijer, S.; Van Der Velden, J.; and Iosup, A. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1): 1–22.

Huang, L.; and Sun, X. 2023. Create ice cream: Real-time creative element synthesis framework based on gpt3. 0. In *2023 IEEE Conference on Games (CoG)*, 1–4. IEEE.

Juego Studio. 2023. how long does it take to develop video game. https://www.juegostudio.com/blog/how-long-does-it-take-to-develop-video-game. Accessed: 2024-07-03.

Kang, S.; Ok, Y.; Kim, H.; and Hahn, T. 2020. Image-to-image translation method for game-character face generation. In *2020 IEEE Conference on Games (CoG)*, 628–631. IEEE.

Kennedy, J.; and Eberhart, R. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, 1942–1948. ieee.

Kernighan, B. W.; and Pike, R. 2006. *The practice of programming*. Addison-Wesley.

Khan, S.; Naseer, M.; Hayat, M.; Zamir, S. W.; Khan, F. S.; and Shah, M. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s): 1–41.

Kreitzer, M.; Ashlock, D.; and Pereira, R. 2019. Automatic generation of diverse cavern maps with morphing cellular automata. In *2019 IEEE Conference on Games (CoG)*, 1–8. IEEE.

Kumaran, V.; Carpenter, D.; Rowe, J.; Mott, B.; and Lester, J. 2023a. End-to-end procedural level generation in educational games with natural language instruction. In *2023 IEEE Conference on Games (CoG)*, 1–8. IEEE.

Kumaran, V.; Mott, B.; and Lester, J. 2019. Generating game levels for multiple distinct games with a common latent space. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 102–108.

Kumaran, V.; Rowe, J.; Mott, B.; and Lester, J. 2023b. Scenecraft: Automating interactive narrative scene generation in digital games with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 86–96.

Liapis, A. 2020. 10 Years of the PCG workshop: Past and Future Trends. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–10.

Liu, J.; Snodgrass, S.; Khalifa, A.; Risi, S.; Yannakakis, G. N.; and Togelius, J. 2021. Deep learning for procedural content generation. *Neural Computing and Applications*, 33(1): 19–37.

Machado, T.; Gopstein, D.; Wang, A.; Nov, O.; Nealen, A.; and Togelius, J. 2019. Evaluation of a recommender system for assisting novice game designers. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 167–173.

Makhmutov, M. 2019. Adaptive game soundtrack generation based on music transcription. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 216–218.

Mandelbrot, B. B. 1985. Self-affine fractals and fractal dimension. *Physica scripta*, 32(4): 257.

Massive software. 2024. about. https://www.massivesoftware.com/about.html. Accessed: 2024-07-03.

Matsumoto, M.; Saito, M.; Haramoto, H.; and Nishimura, T. 2006. Pseudorandom Number Generation: Impossibility and Compromise. *J. Univers. Comput. Sci.*, 12(6): 672–690.

Medsker, L. R.; Jain, L.; et al. 2001. Recurrent neural networks. *Design and Applications*, 5(64-67): 2.

Melhart, D.; Togelius, J.; Mikkelsen, B.; Holmgård, C.; and Yannakakis, G. N. 2023. The ethics of AI in games. *IEEE Transactions on Affective Computing*, 15(1): 79–92.

Merino, T.; Negri, R.; Rajesh, D.; Charity, M.; and Togelius, J. 2023. The five-dollar model: generating game maps and sprites from sentence embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 107–115.

Merrell, P. 2023. Example-based procedural modeling using graph grammars. *ACM Transactions on Graphics (TOG)*, 42(4): 1–16.

Mitsis, K.; Kalafatis, E.; Zarkogianni, K.; Mourkousis, G.; and Nikita, K. S. 2020. Procedural content generation based on a genetic algorithm in a serious game for obstructive sleep apnea. In *2020 IEEE Conference on Games (CoG)*, 694–697. IEEE.

Mittermueller, M.; Ye, Z.; and Hlavacs, H. 2022. EST-GAN: Enhancing style transfer gans with intermediate game render passes. In *2022 IEEE Conference on Games (CoG)*, 25–32. IEEE.

Mohaghegh, S.; Dehnavi, M. A. R.; Abdollahinejad, G.; and Hashemi, M. 2023. PCGPT: Procedural Content Generation via Transformers. *arXiv preprint arXiv:2310.02405*.

Nasir, M. U.; and Togelius, J. 2023. Practical PCG through large language models. In *2023 IEEE Conference on Games (CoG)*, 1–4. IEEE.

Ng, A.; et al. 2011. Sparse autoencoder. *CS294A Lecture notes*, 72(2011): 1–19.

OpenAI. 2024. ChatGPT: Large Language Model. Accessed: 2024-07-03.

Osborn, J. C.; Dickinson, M.; Anderson, B.; Summerville, A.; Denner, J.; Torres, D.; Wardrip-Fruin, N.; and Mateas, M. 2019. Is your game generator working? Evaluating Gemini, an intentional generator. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, 59–65.

Pachet, F. 2004. Beyond the cybernetic jam fantasy: The continuator. *IEEE Computer Graphics and Applications*, 24(1): 31–35.

Pfaltz, J. L. 1972. Web grammars and picture description. *Computer Graphics and Image Processing*, 1(2): 193–220.

Pichlmair, M.; and Putney, C. 2020. Procedural generation for divination and inspiration. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–7.

Prasetya, H. A.; and Maulidevi, N. U. 2016. Search-based Procedural Content Generation for Race Tracks in Video Games. *International Journal on Electrical Engineering & Informatics*, 8(4).

Prins, V. L.; Prins, J.; Preuss, M.; and Gómez-Maureira, M. A. 2023. Storyworld: Procedural quest generation rooted in variety & believability. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–4.

Riddle, A. 2022. A hybrid approach to co-creative story authoring using grammars and language models. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 282–284.

Risi, S.; and Togelius, J. 2020. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8): 428–436.

Russell, S. J.; and Norvig, P. 2016. *Artificial intelligence: a modern approach*. Pearson.

Sarkar, A.; Summerville, A.; Snodgrass, S.; Bentley, G.; and Osborn, J. 2020. Exploring level blending across platformers via paths and affordances. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 280–286.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61: 85–117.

Schrum, J.; Volz, V.; and Risi, S. 2020. Cppn2gan: Combining compositional pattern producing networks and gans for large-scale pattern generation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 139–147.

Shaker, N.; Togelius, J.; and Nelson, M. J. 2016. Procedural content generation in games. *Computational Synthesis and Creative Systems*.

Siler, C. 2022. Open-world narrative generation to answer players' questions. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 307–310.

Snodgrass, S.; and Ontanón, S. 2016. Learning to generate video game maps using markov models. *IEEE transactions on computational intelligence and AI in games*, 9(4): 410–422.

Snodgrass, S.; and Sarkar, A. 2020. Multi-domain level generation and blending with sketches via example-driven bsp and variational autoencoders. In *Proceedings of the 15th international conference on the foundations of digital games*, 1–11.

Song, A.; and Whitehead, J. 2019. TownSim: Agent-based city evolution for naturalistic road network generation. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 1–9.

Song, Y.; Kim, H.; Yoo, T.; Bae, B.-c.; and Cheong, Y.-G. 2020. An intelligent storytelling system for narrative conflict generation and resolution. In *2020 IEEE Conference on Games (CoG)*, 192–197. IEEE.

Sudhakaran, S.; González-Duque, M.; Freiberger, M.; Glanois, C.; Najarro, E.; and Risi, S. 2024. Mariogpt: Open-ended text2level generation through large language models. *Advances in Neural Information Processing Systems*, 36.

Summerville, A.; and Mateas, M. 2016. Super mario as a string: Platformer level generation via lstms. *arXiv preprint arXiv:1603.00930*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, 10(3): 257–270.

Sun, Y.; Li, Z.; Fang, K.; Lee, C. H.; and Asadipour, A. 2023. Language as reality: a co-creative storytelling game experience in 1001 nights using generative AI. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 425–434.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Świechowski, M.; Godlewski, K.; Sawicki, B.; and Mańdziuk, J. 2023. Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3): 2497–2562.

Todd, G.; Earle, S.; Nasir, M. U.; Green, M. C.; and Togelius, J. 2023. Level generation through large language models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–8.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2010. Search-based procedural content generation. In *Applications of Evolutionary Computation: EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I*, 141–150. Springer.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3): 172–186.

van Stegeren, J.; and Myśliwiec, J. 2021. Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 1–8.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Volden, T.; Grbic, D.; and Burelli, P. 2023. Procedurally generating rules to adapt difficulty for narrative puzzle games. In *2023 IEEE Conference on Games (CoG)*, 1–4. IEEE.

Volz, V.; Schrum, J.; Liu, J.; Lucas, S. M.; Smith, A.; and Risi, S. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the genetic and evolutionary computation conference*, 221–228.

Wootton, J. R. 2020a. Procedural generation using quantum computation. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–8.

Wootton, J. R. 2020b. A quantum procedure for map generation. In *2020 IEEE Conference on Games (CoG)*, 73–80. IEEE.

Yannakakis, G. N.; and Togelius, J. 2018. *Artificial intelligence and games*, volume 2. Springer.

Zhang, Y.; Zhang, G.; and Huang, X. 2022. A survey of procedural content generation for games. In *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, 186–190. IEEE.

Zhu, A.; Martin, L.; Head, A.; and Callison-Burch, C. 2023. CALYPSO: LLMs as Dungeon Master's Assistants. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 380–390.