Grounding Natural Language Prompts in Expressive Super Mario Level Generation

Anonymous ACL submission

Abstract

Natural-language-controllable procedural content generation depends critically on how linguistic concepts are grounded in structured representations. We show that widely used benchmarks rely on coarse semantic encodings that collapse distinct concepts, obscure grounding failures, and systematically inflate apparent instruction-following performance. Focusing on Super Mario level generation, we introduce MARIOPCG, a higher-fidelity dataset with expanded semantic coverage, and evaluate multiple decoder-only language models under controlled conditions. Increasing representational granularity exposes severe controllability failures in limited-capacity models that remain invisible under coarser benchmarks, while larger models exhibit stable behavior only when the representation supports meaningful grounding. These findings establish dataset semantic granularity as a necessary condition for valid evaluation of grounded language control and suggest that prior conclusions drawn from semantically collapsed benchmarks reflect representational artifacts rather than model capability. We will publicly release the dataset, prompts, and evaluation code to support reproducibility and further research.

1 Introduction

Procedural content generation (PCG) supports personalization, replayability, and reduced authoring cost by adapting game content to targeted player experiences. A common framing treats generation as the problem of producing content that satisfies experience-level objectives, such as difficulty, engagement, or frustration, often inferred from player behavior or preferences (Yannakakis and Togelius, 2011; Shaker et al., 2012; Yu and Trawick, 2011). Under this perspective, generators are evaluated by their ability to satisfy intended experiential constraints rather than by realism or novelty alone.

To enable controllability, prior PCG systems expose control through tunable parameters, optimization objectives, mixed-initiative tools, or latent control variables learned by generative models (Togelius et al., 2011; Liapis et al., 2013; Yannakakis et al., 2014). While effective, these interfaces are often indirect, require domain expertise, or rely on representations that are not easily interpretable by human designers.

Natural language offers a more expressive and accessible control interface. Recent advances in instruction-following language models enable free-form textual descriptions to be mapped to structured outputs, making language a practical mechanism for controllable generation. In games, this approach has been applied to level generation and world synthesis, including text-conditioned generation of functional puzzle and platformer levels (Todd et al., 2023; Sudhakaran et al., 2023; Schrum et al., 2025).

Using language as a control signal introduces a grounding problem. Natural-language prompts specify abstract concepts and relations, while generators must produce concrete structured representations under hard functional constraints. Grounding quality therefore depends not only on model capacity, but also on the semantic fidelity of the underlying representation and dataset. This issue is especially pronounced in tile-based platformer domains such as Super Mario Bros, where commonly used datasets collapse semantically distinct entities and omit gameplay-relevant tiles. As a result, many concepts that are natural to express in language cannot be explicitly represented or evaluated, conflating representational limitations with model limitations.

In this work, we empirically demonstrate that dataset semantics and representation design fundamentally constrain observable instruction-following behavior. Rather than proposing a new generation architecture, we show how representational granularity shapes the space of expressible instructions and the failure modes that emerge dur-

ing expressive Super Mario level generation.

Our main contributions and findings are summarized as follows:

- We establish **dataset semantic granularity** as a structural constraint on natural-language instruction following in procedural content generation, demonstrating that representational choices fundamentally bound prompt expressivity and the grounding failure modes that can be observed during evaluation.
- We introduce **MarioPCG**, a higher-fidelity Super Mario Bros level dataset with expanded semantic coverage, enabling controlled study of grounded language conditioning while remaining compatible with standard PCGML pipelines and evaluation tooling.
- Through controlled experiments across multiple decoder-only language models under matched training and inference conditions, we show that increasing representational expressivity exposes controllability failures in limited-capacity models that remain hidden under coarser benchmarks, while larger models exhibit more stable behavior with expressive supervision.

2 Task Formulation and Problem Definition

Procedural content generation via machine learning (PCGML) learns generators from existing game artifacts and supports intent-driven design workflows such as mixed-initiative and co-creative generation (Summerville et al., 2018). A persistent constraint in PCGML is limited domain data, as game-specific corpora are typically orders of magnitude smaller than standard language or vision datasets (Awiszus et al., 2020). This limitation is compounded by the heterogeneity of game representations, which are often specific to individual games or franchises and constrain dataset scale and semantic expressivity (Summerville et al., 2018). Throughout this work, we use expressivity to denote the number, granularity, and compositionality of semantic concepts that can be referenced in prompts and evaluated through grounded generation.

We focus on tile-based 2D platformer levels represented as grids of discrete symbols corresponding to tiles and objects. Such grids are commonly linearized into sequences to enable sequence modeling while preserving local spatial dependencies (Summerville and Mateas, 2016). This representation underpins a wide range of PCGML approaches and is used in public datasets such as the Video Game Level Corpus (VGLC) (Summerville et al., 2016).

2.1 Task Definition

We formulate natural-language-controllable level generation as conditional sequence modeling with grounded linguistic inputs. Each training example consists of a natural-language prompt x, describing desired properties of a level, and a corresponding level y, represented as a serialized sequence obtained by linearizing a tile grid into discrete tokens for tiles and objects (Summerville and Mateas, 2016). Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, we fine-tune a decoder-only language model using standard causal language modeling loss, where the prompt x_i provides conditioning context and the loss is applied to the level tokens y_i.

This formulation treats controllable PCG as an instruction-following problem under hard structural and functional constraints. The model must generate sequences that are not only distributionally plausible, but also structurally coherent and aligned with the semantics expressed in the natural-language prompt.

2.2 Natural-Language Controllable PCG as a Grounding Problem

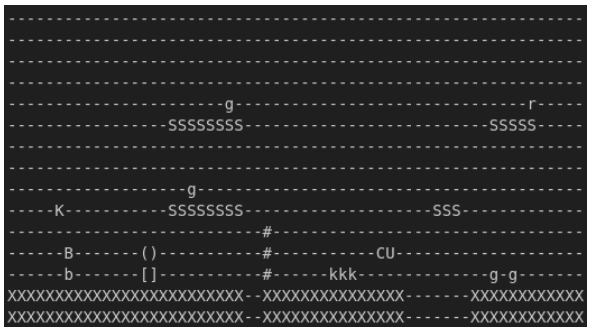
A central challenge in controllable PCG is specifying user intent in a form that is both expressive and operational. Prior work commonly relies on non-linguistic control channels, such as latent vectors optimized to satisfy target properties or structured constraints exposed through interactive editing. For example, Volz et al. evolve latent vectors in the space of a generative adversarial network to steer level generation toward desired characteristics, using continuous control signals that are not directly interpretable by human designers and typically require iterative search (Volz et al., 2018). Mixed-initiative tools similarly expose control through explicit constraints and editor interactions rather than free-form language.

In contrast, text-conditioned generation treats control as a grounding problem, where natural-language descriptions must be mapped to structured, non-linguistic outputs under domain-specific constraints. Prior work such as MarioGPT demonstrates that linguistic prompts can be used to condition level generation by mapping textual de-

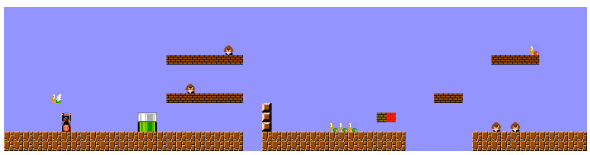
scriptions directly to serialized level representations (Sudhakaran et al., 2023). Related work further shows that large language models trained for NLP tasks can be adapted to generate structured game content when appropriately conditioned (Todd et al., 2023). This framing aligns controllable PCG with established problems in instruction following and semantic grounding, providing a principled basis for analyzing how representational choices shape observable instruction-following behavior.

Semantic Granularity. Within this framing, we define *semantic granularity* as the resolution at which distinct functional entities and structural relations are explicitly represented and supervised in the dataset. Coarser representations collapse semantically distinct elements into shared symbols, limiting the set of concepts that can be expressed in natural-language prompts and obscuring grounding failures during evaluation. Finer-grained representations increase expressivity by preserving meaningful distinctions, but simultaneously impose stricter grounding requirements on instruction-following models.

3 MarioPCG Dataset



(a) String Representation



(b) Actual Level

Figure 1: String representation and corresponding level layout in the MarioPCG dataset.

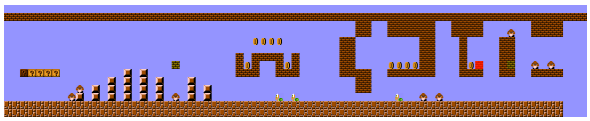
We introduce MARIOPCG, an open-source dataset for Super Mario Bros level generation that increases semantic granularity in tile-based representations while remaining compatible with standard PCGML pipelines. MarioPCG builds on the widely used symbolic grid encoding for Super Mario Bros, refining it to preserve fine-grained distinctions among enemies, objects, and structural elements.

MarioPCG addresses two limitations of existing datasets: limited dataset scale and reduced semantic coverage. The dataset contains a larger number of levels than commonly used benchmarks and employs an expanded tile vocabulary that more accurately reflects gameplay-relevant entities present in the original games.

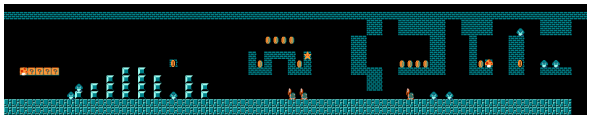
The dataset is constructed from levels recreated in Super Mario Maker 2, which provides a unified authoring environment capable of reproducing levels from multiple Super Mario Bros titles using a consistent internal representation. MarioPCG consists primarily of faithful recreations of original levels, complemented by a small subset of original user-created levels. Level selection prioritizes adherence to established Super Mario level design patterns, as characterized by Dahlskog et al. (Dahlskog and Togelius, 2012) and Thompson (Thompson, 2015). The released dataset contains exclusively symbolic, ASCII-based tile grids encoding abstract level structure and does not include original visual, audio, or executable game assets. Source levels were obtained from a publicly released Super Mario Maker 2 API dataset by The Great Rambler (TheGreatRambler, 2022).

Because the source data originates from Super Mario Maker 2, preprocessing is required to convert levels into the standard tile-based representation used in Super Mario Bros PCG research. This includes minor tile placement adjustments to better reflect original game mechanics. The resulting dataset supports a wide range of enemies, items, and blocks compatible with the Mario AI Framework. In total, MarioPCG includes 26 distinct tile types, compared to 13 in the VGLC, enabling a broader range of concepts to be expressed and controlled through natural-language prompts.

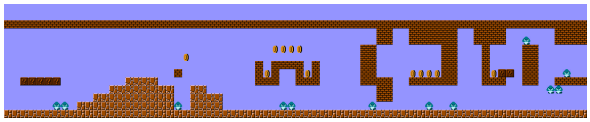
The final dataset contains 112 levels, including 77 faithful recreations from Super Mario Bros, Super Mario Bros: The Lost Levels, Super Mario Land, and New Super Mario Bros, as well as 35 original user-created levels. While additional recreations exist in Super Mario Maker 2, many could not be included due to mechanics or enemies not supported by the Mario AI Framework (Khalifa, 2019). By providing higher-fidelity level representations with expanded semantic coverage, MarioPCG enables controlled study of grounded natural-



(a) MarioPCG Representation



(b) Original Game Level



(c) VGLC Representation

Figure 2: Comparison of level 1–2 in Super Mario Bros across MarioPCG, the original game, and the VGLC representation.

language control in procedural content generation while remaining independent of any specific modeling approach.

4 Experiments

4.1 Setup

Levels are linearized into token sequences using column-wise traversal. Training examples are constructed by extracting fixed-length slices of 800 characters using a sliding window that advances by one column at a time without crossing level boundaries, yielding 17,100 training examples for MarioPCG. Each slice is paired with an automatically generated natural-language prompt following the MarioGPT procedure, where prompts describe the presence and quantity of tiles, enemies, and structures. Prompts and level slices are formatted as user–assistant pairs, with prompts provided as input and serialized levels as output.

To isolate the effect of dataset composition, we construct a parallel training set from the original MarioGPT dataset derived from VGLC resulting in 7,250 training examples. The DistilGPT2 VGLC baseline is reproduced using the official MarioGPT implementation and training recipe.

For evaluation, we generate 500 samples per model and dataset using a fixed sampling temperature of 2.0, and evaluate all models under identical sampling conditions.

4.2 Evaluation Metrics

We evaluate generated levels along four complementary axes that capture different aspects of grounded instruction following: functional validity, prompt adherence, structural integrity, and memorization.

Playability. We measure playability using the Mario AI Framework by simulating each generated level with Robin Baumgarten's A* agent. A level is considered playable if the agent completes it in a single attempt under default settings. We interpret solvability as a conservative functional-validity signal rather than a definitive measure of human playability, due to rare agent failure cases.

Prompt Adherence. To measure how well generated levels satisfy the natural-language prompt, we adopt the caption adherence score (c-score) introduced by Schrum et al. (Schrum et al., 2025), with a modification to account for under-specified prompts. In practice, prompts typically reference only a subset of available concepts. We therefore compute adherence only over concepts explicitly mentioned in the prompt, rather than treating unmentioned concepts as absent.

For example, given the prompt "many goombas and coins", a valid and playable level necessarily contains ground and supporting structures that are not mentioned. Under the original c-score definition, such levels would be penalized for containing unspecified concepts, while degenerate levels consisting primarily of empty space populated only by goombas and coins could receive higher adherence scores despite being unplayable. This failure mode becomes more pronounced as semantic vocabularies grow more granular. Our modification avoids this issue by scoring adherence only over explicitly referenced concepts.

Given a prompt p and an automatically generated caption c for the produced level, we compute:

$$\text{c-score}(p, c) = \frac{\sum_{t \in T(p)} \text{match}(\text{phrase}(t, p), \text{phrase}(t, c))}{|T(p)|} \tag{1}$$

$$\text{match}(p_t, c_t) = \begin{cases} 1.0 & \text{if } p_t = c_t \\ 1.0 - \dfrac{|\text{qu}(p_t) - \text{qu}(c_t)|}{|Q| - 1} & \text{if co}(p_t) \wedge \text{co}(c_t) \\ 0.1 & \text{if } p_t \neq \emptyset \wedge c_t \neq \emptyset \\ -1.0 & \text{otherwise} \end{cases} \tag{2}$$

where $T(p) \subseteq T$ denotes the set of concepts referenced in the prompt. The function $\text{phrase}(t, s)$ extracts the quantity phrase for concept t from text s, and $\text{match}(\cdot, \cdot)$ assigns exact or graded partial credit based on ordinal distance over an ordered quantity scale (e.g., $\{no, few, some, many\}$). This

formulation isolates adherence to stated constraints while allowing models to complete unspecified structure necessary for coherent, playable levels.

Structural Integrity. We additionally evaluate domain-specific integrity constraints for multi-tile structures. Pipes and cannons require consistent top and body arrangements, and generated outputs sometimes violate these constraints. We report the percentage of generated levels containing at least one malformed pipe and at least one malformed cannon.

Memorization.

To measure direct copying from the training set, we adopt an n-gram-based plagiarism metric over level columns. For a given n, we extract all ordered column n-grams from the training levels and store them as a reference set T_n. For each generated level, we extract the corresponding ordered n-grams L_n and compare each $l \in L_n$ against T_n. An n-gram is counted as matched if it appears exactly in T_n, or if it matches a training n-gram with normalized Hamming similarity of at least 95%, allowing for small tile-level variations that preserve overall structure.

We compute memorization as the fraction of ordered n-grams in L_n that match the training set. A generated level is considered copied if more than 50% of its ordered n-grams are matched. Following Dahlskog and Togelius' definition of level "beats" as short recurring challenge segments separated by lower-intensity regions (Dahlskog and Togelius, 2012), we use a small n-gram size (n=2) together with a majority-overlap threshold, so that isolated beat reuse is permitted while extended structural copying results in more than 50% n-gram overlap.

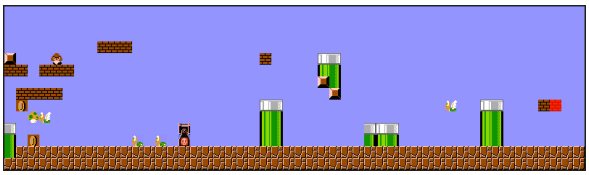
5 Results

We evaluate controllable level generation across multiple decoder-only language models, including Qwen 2.5, Qwen 3, Gemma 3, and Llama 3.1, all fine-tuned using standard causal language modeling loss applied to assistant tokens only. Models are trained on user–assistant formatted data without architectural modifications, external prompt encoders, or cross-attention mechanisms. This stands in contrast to MarioGPT's DistilGPT2-based architecture, which injects prompt information through cross-attention layers and relies on a frozen encoder for prompt representations.

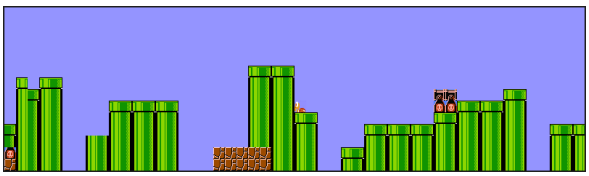
5.1 Effect of Prompt and Dataset Granularity

Expanding the prompt space reveals severe controllability failures in the MarioGPT base model that are not observable under coarser prompt schemas. As shown in Figure 3, generation collapses toward overuse of specific structural elements, most notably pipes and, to a lesser extent, cannons, at the expense of structural diversity and coherence. We refer to this failure mode as *Structural Overcompensation*, where a small subset of structures dominates generation regardless of the prompt's relative balance. This degradation becomes increasingly pronounced as the prompt space becomes more granular and natural. In contrast, larger and more recent decoder-only models accommodate the expanded semantic vocabulary without similar collapse, motivating their use for controllable PCG under richer natural-language prompts.

Figure 3 further shows that increasing the number of promptable classes causes severe degradation in the MarioGPT DistilGPT2 base model. Notably, a substantial adherence drop is observed even when holding the prompt schema fixed to the original four classes and changing only the dataset, indicating that controllability failures cannot be attributed solely to the prompt interface. This behavior reflects *Prompt Signal Dilution*, where explicit linguistic constraints are overwhelmed by dataset-level regularities under increased semantic expressivity.



(a) MarioGPT (DistilGPT2) fine-tuned on MarioPCG with four prompt classes.



(b) MarioGPT (DistilGPT2) fine-tuned on MarioPCG with eleven prompt classes.

Figure 3: Generation degradation of the MarioGPT base model as the number of promptable classes increases. Prompt: `many enemies, some pipes, some blocks, high elevation`.

Table 1 summarizes results across datasets and model capacities. To isolate the effect of dataset

Model	Adherence	Playable	Broken Pipes	Broken Cannons	Memorization
VGLC (Summerville et al., 2016)					
DistilGPT2 (MarioGPT)	0.559	0.872	0.255	0.006	0.824
Qwen 2.5 (7B)	0.488	0.684	0.180	0.003	0.194
Qwen 2.5 (14B)	0.537	0.718	0.188	0.004	0.288
Qwen 3 (8B)	0.548	0.720	0.179	0.002	0.192
Qwen 3 (14B)	0.496	0.738	0.171	0.002	0.314
Gemma 3 (12B)	0.361	0.612	0.073	0.012	0.183
Llama 3.1 (8B)	0.511	0.662	0.146	0.002	0.178
MarioPCG (Ours)					
Qwen 2.5 (7B)	0.690	0.780	0.018	0.004	0.196
Qwen 2.5 (14B)	0.704	0.784	0.006	0.004	0.272
Qwen 3 (8B)	0.698	0.750	0.014	0.000	0.190
Qwen 3 (14B)	0.691	0.733	0.004	0.000	0.226
Gemma 3 (12B)	0.449	0.612	0.088	0.060	0.234
Llama 3.1 (8B)	0.677	0.733	0.016	0.021	0.264

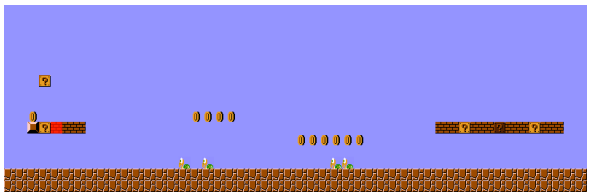
Table 1: Main results across datasets and model capacities. All results use the same 4-class prompt interface at inference time. Models trained on MarioPCG receive 11-class semantic supervision during training.

semantic granularity from the prompt interface, all results use the same four-class prompt schema at inference time. For the VGLC dataset, models are trained and evaluated using this schema. For MarioPCG, models are trained with the full eleven-class semantic schema but evaluated using prompts that mention only four randomly selected classes.

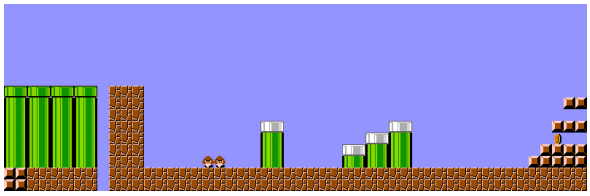
On the VGLC dataset, the MarioGPT baseline exhibits both high playability and high memorization. Because many VGLC levels are fully playable by construction, reproducing large portions of the training data can yield high solvability under automated agents. As a result, playability alone can overestimate generalization performance in settings where memorization is prevalent, since functional validity may arise from copying rather than grounded instruction following. Modern decoder-only models generally exhibit lower copying on VGLC but do not outperform MarioGPT in prompt adherence under the same coarse prompt schema. We additionally observed that Gemma 3 trained on VGLC often collapses toward generating overly simple and repetitive level structures that are largely invariant to the prompt, producing outputs that are technically playable and satisfy coarse prompt constraints but are qualitatively poor and weakly responsive to prompt variation.

When trained on MarioPCG but evaluated using the four-class prompt interface, the MarioGPT base model exhibits essentially unchanged prompt adherence and memorization. We hypothesize that this behavior arises from implicit attribute interference. Prior work shows that unannotated attributes in datasets can provide strong learning signals that degrade control of explicitly specified attributes, a phenomenon referred to as attribute transfer (Ma et al., 2023), which manifests here where unprompted structural and compositional attributes dominate generation in limited-capacity models. In this setting, the four-class prompt constrains only coarse tile categories, while many structural and compositional properties of level slices remain unspecified. Because MarioPCG covers a broader range of realizations under the same prompt labels, limited-capacity models may prioritize modeling dataset-level regularities over preserving the prompt signal, weakening controllability. The absence of this degradation in larger decoder-only models supports this interpretation.

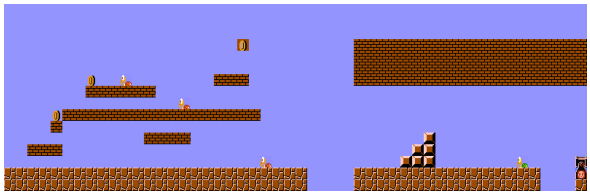
Figure 4 presents qualitative examples of levels generated under distinct natural-language prompts, illustrating how entity types and quantity specifications are grounded in the generated structure. Taken together, these effects highlight recurring patterns of grounding failure as semantic granularity increases, which can help interpret aggregate controllability metrics.



(a) `Make a level with many coins, some powerups and some koopas`



(b) `I want a level that has many pipes and some goombas`



(c) `Generate a level with many breakable blocks, no pipes, some koopas, no goombas, a few special enemies and a few coin blocks`

Figure 4: Examples of levels generated under distinct natural-language prompts.

Model	Adherence	Playable	Broken Pipes	Broken Cannons	Memorization
Qwen 2.5 (7B)	0.725	0.707	0.017	0.002	0.116
Qwen 2.5 (14B)	0.753	0.712	0.016	0.002	0.087
Qwen 3 (8B)	0.723	0.596	0.014	0.000	0.108
Qwen 3 (14B)	0.724	0.716	0.022	0.000	0.090
Gemma 3 (12B)	0.659	0.587	0.044	0.002	0.091
Llama 3.1 (8B)	0.672	0.701	0.018	0.002	0.170

Table 2: Results on MarioPCG using exact numerical quantity specification in prompts.

5.2 Effect of Quantity Specification in Prompts

We compare semantic quantity descriptors, such as *some* or *many*, with exact numerical quantity specifications in natural-language prompts. Aside from the form of quantity expression, all other factors, including dataset, models, and training procedure, are held constant.

Numerical prompts specify exact target counts during generation but are evaluated using the same ordinal quantity bins as semantic prompts. After generation, realized quantities are mapped to bins and compared against the bins implied by the prompt, ensuring both formulations are scored under the same adherence metric.

Exact numerical quantity specification yields modest improvements in prompt adherence for the strongest models, particularly Qwen 2.5 and Qwen 3, while Gemma 3 and Llama 3.1 exhibit slight degradation. Playability generally decreases under numerical prompting, indicating a trade-off between precise quantity control and structural validity. These results suggest that enforcing exact numerical constraints can interfere with maintaining global level coherence, especially in smaller or less robust models.

This behavior aligns with prior work on numeracy in language models. Earlier studies show that standard language models treat numerals as discrete symbols without explicit numerical structure, leading to poor handling of exact values (Spithourakis and Riedel, 2018). More recent work demonstrates that large language models encode numerical magnitude in hidden representations in a linearly decodable manner, indicating that numeric value information is implicitly present despite token-level limitations (Zhu et al., 2025). These results further illustrate that increased constraint precision does not guarantee better grounding under insufficient representational support.

6 Segment-Chained Generation

Decoder-only models trained on fixed-width level slices do not naturally generate arbitrary-length levels in a single forward pass. We therefore introduce *segment-chained generation*, an inference-time continuation procedure that composes multiple fixed-width generations into a longer level while leaving the underlying model and training objective unchanged.

Fixed-size scene generation is common in PCGML, and longer levels are often constructed by composing multiple scenes or segments (Volz et al., 2018; Green et al., 2020; Summerville and Mateas, 2016). Similar distinctions between scene-level generation and full-level construction appear in recent text-conditioned diffusion approaches (Schrum et al., 2025). Segment-chained generation extends slice-trained decoder-only generators to arbitrary length while preserving local grounding behavior. We do not evaluate long-horizon grounding in this work.

6.1 Overlap-Based Continuation

Models are trained to generate fixed-width segments of 50 columns and 16 tiles in height. At inference time, we repeatedly decode segments to reach a target level length. To encourage continu-

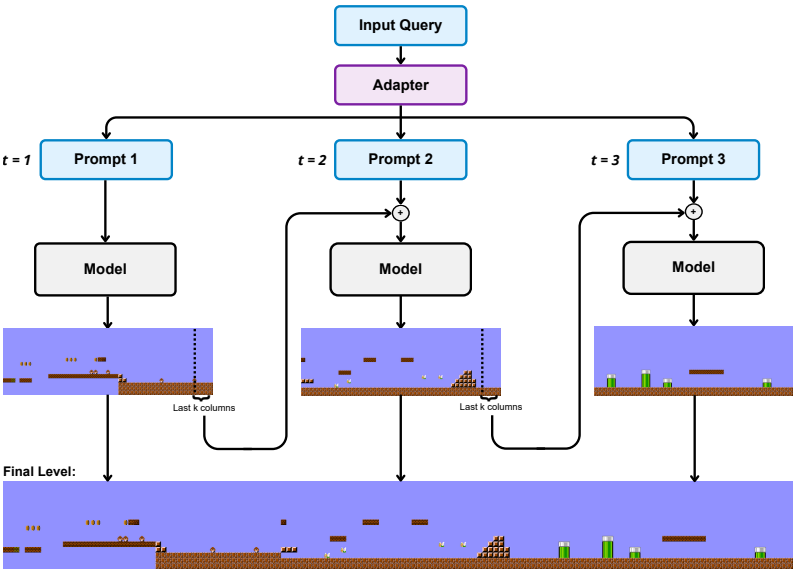


Figure 5: Temporal prompt decomposition for segment-chained generation. An input query is decomposed into ordered prompts. Each segment is generated conditioned on the current prompt and overlapping context, and only the non-overlapping suffix is appended to the growing level.

ity, we carry forward an overlap of o columns by inserting the final o columns of segment t as initial assistant-side context for segment $t{+}1$. A new segment of width W is then decoded, and only its non-overlapping suffix is appended to the growing level. This rolling-window procedure stays within the model's trained segment format and requires no architectural modifications.

6.2 Temporal Prompt Decomposition

Segment-chained generation also supports temporally varying intent. Given a prompt describing ordered phases, we apply a prompt adapter that decomposes the input into a sequence of prompts. Segments are generated sequentially, each conditioned on the current sub-prompt and the overlapping context from the previous segment. The prompt is updated when advancing to the next phase. Figure 5 illustrates this procedure.

Conceptually, this approach aligns with long-form generation methods in NLP that decompose generation into ordered steps or phases (Yang et al., 2022, 2023). Unlike revision-based reprompting pipelines, segment-chained generation uses a fixed-window continuation loop with scheduled prompt changes, chosen to match the fixed-segment training regime while enabling temporal control.

7 Conclusion

This work establishes dataset semantic granularity as a decisive factor in revealing grounding and controllability in natural-language-conditioned procedural content generation. By varying representation design while holding model architectures and training fixed, we show that coarse benchmarks systematically overestimate instruction-following ability by collapsing distinct semantic concepts and obscuring failure modes. Through the introduction of MARIOPCG, a higher-fidelity Super Mario level dataset with expanded semantic coverage, we demonstrate that increased representational expressivity exposes grounding failures that remain hidden under existing evaluation protocols. Limited-capacity models degrade sharply as the prompt space becomes richer and more natural, while larger decoder-only models exhibit stable behavior only when the representation supports meaningful grounding. These results indicate that prior conclusions about controllable generation are often shaped by representational constraints in benchmark design rather than model capability alone. More broadly, this work reframes controllable procedural generation as a grounding and representation problem, providing a principled foundation for future benchmarks and evaluations of instruction-following behavior.

8 Limitations

This study is conducted under a deliberately constrained experimental setting. The evaluation is limited to tile-based Super Mario Bros level generation using symbolic grid representations, which

may not directly generalize to other procedural content generation domains such as three-dimensional environments, continuous control spaces, or graph-structured representations.

Natural-language prompts are automatically derived from level content rather than authored by humans. While this enables controlled manipulation of semantic granularity, it restricts linguistic diversity and evaluates grounding under structured, synthetic prompt distributions rather than open-ended human instructions.

Grounding quality is measured using automated proxy metrics, including prompt adherence, playability, structural validity, and memorization indicators. These metrics capture complementary aspects of grounded generation but do not fully reflect human judgments of controllability or design intent.

Finally, models generate fixed-width level segments, with longer levels produced via inference-time composition. This design facilitates local grounding analysis but does not address native long-horizon generation or global structural dependency modeling.

Although grounded in PCG, the observed failure modes arise from representational collapse and may therefore extend beyond this domain.

9 Ethical Considerations

This work uses level layout data derived from recreations in Super Mario Maker 2. The released MarioPCG dataset contains only symbolic, ASCII-based representations of level layouts and does not include original visual, audio, or executable game assets. Levels are represented as abstract character grids commonly used in prior Super Mario Bros. procedural content generation research.

The symbolic grid representations are a long-established abstraction in procedural content generation research for Super Mario Bros. and related domains. They encode functional tile categories and structural relationships rather than expressive audiovisual elements, and are used as analytical representations for modeling and evaluation rather than as substitutes for the original game content.

The source data was obtained from a publicly released Super Mario Maker 2 API dataset by The Great Rambler under a CC BY-NC-SA 4.0 license (TheGreatRambler, 2022). MarioPCG does not include player identifiers, metadata, or other user-related information and is distributed solely for non-commercial academic research.

Acknowledgments

Anonymous Acknowledgments

References

Maren Awiszus, Frederik Schubert, and Bodo Rosenhahn. 2020. Toad-gan: Coherent style level generation from a single example. In *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Yi-Chun Chen and Arnav Jhala. 2025. Gametilenet: a semantic dataset for low-resolution game art in procedural content generation. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE '25. AAAI Press.

Steve Dahlskog and Julian Togelius. 2012. Patterns and procedural content generation: revisiting mario in world 1 level 1. In *Proceedings of the First Workshop on Design Patterns in Games*, DPG '12, page 1–8. ACM.

Michael Cerny Green, Luvneesh Mugrai, Ahmed Khalifa, and Julian Togelius. 2020. Mario level generation from mechanics using scene stitching. In *Proceedings of the IEEE Conference on Games (CoG)*. arXiv.

Sergey Karakovskiy and Julian Togelius. 2012. The mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67.

Ahmed Khalifa. 2019. Mario-ai-framework. `https://github.com/amidos2006/Mario-AI-Framework`. GitHub repository.

Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. 2013. Sentient sketchbook: Computer-aided game level authoring. In *International Conference on Foundations of Digital Games*.

Congda Ma, Tianyu Zhao, Makoto Shing, Kei Sawada, and Manabu Okumura. 2023. Focused prefix tuning for controllable text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1116–1127, Toronto, Canada. Association for Computational Linguistics.

Jacob Schrum, Olivia Kilday, Emilio Salas, Bess Hagan, and Reid Williams. 2025. Text-to-level diffusion models with various text encoders for super mario bros. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Noor Shaker, Julian Togelius, and Mark J. Nelson. 2011. The 2010 mario AI championship: Level generation track. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(4):332–347.

Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. 2012. Towards player-driven procedural content generation. In *Proceedings of the 9th Conference on Computing Frontiers*, CF '12, page 237–240, New York, NY, USA. Association for Computing Machinery.

Kristin Siu, Matthew Guzdial, and Mark Riedl. 2017. Evaluating singleplayer and multiplayer in human computation games. pages 1–10.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, Melbourne, Australia. Association for Computational Linguistics.

Shyam Sudhakaran, Miguel Gonzalez-Duque, Claire Glanois, Matthias Freiberger, Elias Najarro, and Sebastian Risi. 2023. Mariogpt: Open-ended text2level generation through large language models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Adam Summerville and Michael Mateas. 2016. Super mario as a string: Platformer level generation via lstms. In *Proceedings of the First International Joint Conference of DiGRA and FDG*.

Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgard, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270.

Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Onta n'on Villar. 2016. The vglc: The video game level corpus. *Proceedings of the 7th Workshop on Procedural Content Generation*.

TheGreatRambler. 2022. Mario maker 2 datasets.

T. Thompson. 2015. The fine line between rehash and sequel: Design patterns of the super mario series. Design Patterns in Games Workshop.

Graham Todd, Sam Earle, Muhammad Umair Nasir, Michael Cerny Green, and Julian Togelius. 2023. Level generation through large language models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, FDG 2023, page 1–8. ACM.

Julian Togelius, Georgios Yannakakis, Kenneth Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3:172 – 186.

Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, page 221–228. ACM.

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. 2023. DOC: Improving long story coherence with detailed outline control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3378–3465, Toronto, Canada. Association for Computational Linguistics.

Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4393–4479, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Georgios Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on*, 2:147–161.

Georgios N. Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative cocreativity. In *International Conference on Foundations of Digital Games*.

Hong Yu and Tyler Trawick. 2011. Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 7(1):208–213.

Fangwei Zhu, Damai Dai, and Zhifang Sui. 2025. Language models encode the value of numbers linearly. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 693–709, Abu Dhabi, UAE. Association for Computational Linguistics.

A Related Work

A.1 Procedural Content Generation

Procedural Content Generation (PCG) studies algorithmic methods for producing game content such as levels, rules, and quests, often with the goal of improving replayability, personalization, or authoring efficiency. Early PCG systems were typically rule-based or search-based, while more recent work increasingly leverages machine learning to imitate designer distributions and capture implicit design regularities, an approach commonly referred to as PCG via Machine Learning (PCGML) (Summerville et al., 2018). A recurring challenge for PCGML is data scarcity and dataset

bias, since level representations and design conventions are highly game-specific. Public corpora such as the Video Game Level Corpus (VGLC) partially address this by providing standardized, tile-based level encodings across multiple games (Summerville et al., 2016).

Super Mario Bros (SMB) has become a canonical benchmark for PCG research due to its well-studied design principles, strong functional constraints (playability), and widely used evaluation tooling. The Mario AI Benchmark and its associated competitions helped standardize experimental protocols for SMB agents and level generation, enabling reproducible evaluation and comparison across generators (Karakovskiy and Togelius, 2012; Shaker et al., 2011). Building on this ecosystem, a large body of work has explored both learning-based and search-based generators for SMB, including sequential modeling with string representations (Summerville and Mateas, 2016), adversarial approaches that evolve levels in a learned latent space (Volz et al., 2018), one-shot or few-shot generative methods designed to reduce data requirements (Awiszus et al., 2020), and reinforcement learning formulations that frame level design as a sequential decision process (Khalifa et al., 2020).

A.2 Prompt Guided Level Generation

Recent work has explored natural language as a high-level control interface for level generators, motivated by the broader trend of using prompts to steer generative models. In this setting, the generator must map a free-form or templated textual request to a structured level representation while preserving both global coherence and local constraints. Todd et al. (Todd et al., 2023) show that large language models can generate functional levels (in Sokoban) and that generation quality scales strongly with dataset size, highlighting both the promise of language-based generators and the importance of training data construction. In the SMB domain, MarioGPT (Sudhakaran et al., 2023) demonstrates open-ended text-to-level generation by fine-tuning a GPT-style model on tile sequences paired with automatically derived prompts, enabling controllable generation through descriptive text. More recently, text-conditioned diffusion models have also been explored for tile-based SMB generation, emphasizing captioning strategies, text encoders, and designer-facing workflows for assembling longer levels from generated segments (Schrum et al., 2025).

A.3 PCG Datasets

PCGML methods depend strongly on the availability and structure of training data, yet game level datasets are often game-specific and encode strong representational assumptions. As noted by Summerville et al. (Summerville et al., 2018), even games within the same genre frequently differ in underlying data structures and semantic interpretations, limiting dataset reuse across titles and constraining dataset scale. Public datasets therefore play a central role in shaping what types of generative models and controls are feasible.

The Video Game Level Corpus (VGLC) (Summerville et al., 2016) is the most widely used dataset for level generation research. It provides standardized, parseable representations of levels from twelve games, including multiple Super Mario Bros. titles, using tile-based, graph-based, and vector encodings. The tile representation in particular has enabled a large body of PCGML work and effectively established SMB as a canonical benchmark. However, VGLC relies on simplified abstractions introduced during automated extraction, such as collapsing multiple enemy types into a single symbol and omitting certain structural details. As a result, several recent works rely on cleaned or modified versions of VGLC, treating its levels as noisy or incomplete data rather than ground truth. Text-conditioned diffusion approaches for Super Mario Bros. (Schrum et al., 2025) follow this strategy by correcting errors in the corpus and pairing level segments with structured textual descriptions, enabling generation and repair conditioned on semantic intent and improving alignment beyond direct imitation of tile encodings.

Beyond static level corpora, player activity has also been explored as a data source. The Gwario human computation game (Siu et al., 2017) embeds annotation tasks into modified Super Mario Bros. levels, illustrating how player behavior can provide structured signals during play. Although not designed for level generation, such playtrace-based data captures aspects of difficulty and engagement absent from static layouts, but is costly to collect and difficult to scale.

GameTileNet (Chen and Jhala, 2025) instead focuses on semantic annotation of low-resolution game art assets. By providing labeled tiles with affordance and connectivity information, it supports narrative-to-visual alignment and vision-language

modeling at the asset level. While it does not represent full levels, it complements level corpora by enriching tile-level semantics within PCGML datasets.

B Dataset Details

Tile Type	Symbol	Visualization
Empty	-	
Ground Block	X	
Hard Block	#	
Breakable Block	S	
Question Block w/ Powerup	?	
Coin	o	
Left pipe top	<	
Right pipe top	>	
Left pipe lower	[	
Right pipe lower	]	
Pipe Top Left w/ Plant Enemy	(	
Pipe Top Right w/ Plant Enemy	)	
Cannon Top	B	
Cannon Body	b	
Coin Brick Block	C	
Mushroom Brick Block	U	
Coin Question Block	!	
Invisible 1-Up Block	1	
Invisible Coin Block	2	
Goomba	g	
Winged Goomba	G	
Koopa	k	
Winged Koopa	K	
Red Koopa	r	
Winged Red Koopa	R	
Spiny	y	

Table 3: All Entities in Dataset

Table 3 demonstrates all unique tiles in the MarioPCG dataset

C Quantity Bins

Prompts are constructed as combinations of entity classes and quantity specifications, computed over fixed-width level segments of 50 columns.

Count-based entities may be specified using either ordinal keywords or exact numerical quantities:

- {*no*, *few*, *some*, *many*} or exact counts special enemies
- {*no*, *few*, *some*, *many*} or exact counts pipes
- {*few*, *some*, *many*} or exact counts ground blocks
- {*few*, *some*, *many*} or exact counts hard blocks
- {*few*, *some*, *many*} or exact counts breakable blocks
- {*few*, *some*, *many*} or exact counts coin blocks
- {*no*, *few*, *some*, *many*} or exact counts coins
- {*no*, *few*, *some*, *many*} or exact counts powerups
- {*no*, *few*, *some*, *many*} or exact counts goombas
- {*no*, *few*, *some*, *many*} or exact counts koopas

When ordinal keywords are used, entity counts are mapped to dataset-derived bins specific to each class, as defined in Table 4. The keyword *no* corresponds to zero occurrences. When numerical quantities are used, the specified count is interpreted directly during generation and mapped back to ordinal bins for evaluation.

Ordinal and numerical quantity specifications correspond to separate training conditions; models are trained and evaluated using one prompt formulation at a time and do not jointly support both within a single model.

Elevation is specified independently:

- {*low*, *high*} elevation

The *low* and *high* elevation labels are determined from the height of the highest unbreakable blocks in a segment of the level.

Example prompts:

- many goombas, some pipes, few coin blocks, high elevation
- 5 goombas, 2 pipes, low elevation

Table 4 summarizes the entity classes, their associated tile symbols, and the thresholds used to map counts to ordinal bins.

D Training Details

D.1 Model Initialization

All models are initialized from instruction-tuned, decoder-only checkpoints and fine-tuned using parameter-efficient adaptation. We use the Unsloth framework for model loading and training. The

Entity	Tile Symbols	Low	Medium	High
Special Enemy	B, y	1	2	3
Pipe	<>, ()	1	2	5
Ground Block	X	1	87	150
Hard Block	#	1	11	23
Coin Block	Q, !, 2, C	1	2	6
Breakable Block	S	1	19	38
Coin	o	1	5	10
Powerup	1, ?, U, L	1	2	3
Goomba	g, G	1	2	5
Koopa	r, R, k, K	1	3	6

Table 4: Entity-to-tile mappings and ordinal quantity bin thresholds used for prompt generation and evaluation. Each entity aggregates multiple tile symbols from Table 3. Quantity bins correspond to dataset-derived low, medium, and high count thresholds.

maximum sequence length is fixed to 2048 tokens for all experiments. All models are loaded in 4-bit precision using bitsandbytes quantization, with `dtype` left to the framework default.

The following checkpoints are used:

- `unsloth/Qwen3-14B-unsloth-bnb-4bit`
- `unsloth/Qwen3-8B-bnb-4bit`
- `unsloth/gemma-3-12b-it-bnb-4bit`
- `unsloth/Qwen2.5-7B-Instruct-bnb-4bit`
- `unsloth/Qwen2.5-14B-Instruct-bnb-4bit`
- `unsloth/Meta-Llama-3.1-8B-Instruct-bnb-4bit`

D.2 Fine-Tuning Method

All models are fine-tuned using Low-Rank Adaptation (LoRA), with base model weights frozen throughout training. LoRA adapters are applied to attention and MLP projection layers following standard practice for decoder-only transformers.

The LoRA configuration is held constant across all model families:

- LoRA rank: 128
- LoRA alpha: 128
- LoRA dropout: 0.05
- Bias: none

D.3 Data Formatting

Training data are formatted as chat-style user–assistant interactions. Prompts are provided as user messages and serialized level representations are provided as assistant responses. Model-specific chat templates are applied for Gemma 3, Llama 3.1, Qwen 2.5, and Qwen 3 to ensure tokenizer compatibility. Aside from required formatting differences, the training data and procedure are identical across models.

D.4 Optimization and Training

All models are trained for a single epoch using identical optimization hyperparameters. One epoch was selected based on internal validation, where additional epochs led to increased memorization without consistent improvements in prompt adherence. During training, 10% of the data is held out as a validation set. The validation split is used only for monitoring training behavior, selecting hyperparameters and selecting the number of epochs.

Training hyperparameters are as follows:

- Per-device batch size: 2
- Gradient accumulation steps: 4
- Effective batch size: 8
- Learning rate: 2×10^{-4}
- Warmup ratio: 0.05
- Optimizer: AdamW (8-bit)
- Weight decay: 0.01
- Learning rate scheduler: linear