Which Structural Constraints Are Learnable? A Regime Map for a Minecraft Voxel Generator

Alex Chengyu Li*
Independent Researcher
Tokyo, Japan
cl7076@nyu.edu

Abstract

Neural procedural content generation (PCG) systems produce 3D game content, but practitioners lack guidance on which structural constraints their generator will enforce. We condition a VQ-VAE + autoregressive transformer with classifier-free guidance on six discretized structural tokens, generate new Minecraft buildings (32^3 voxel grids, 513 remapped block tokens), and measure 14 output properties. The properties separate into three regimes: **Controllable** (9 properties, >100% relative shift; 7 confident), **Approachable** (4, 20–100%), and **Unresponsive** (1, <20%). The product of effective signal and training CV correlates with relative responsiveness for emergent (not directly conditioned) properties (Spearman ρ=0.879, p=0.002, n=10); the small sample limits predictive generalization. Varying the guidance scale helps diagnose *representation ceilings* (requiring architectural changes) versus *frequency floors* (potentially addressable with more data or stronger guidance), giving practitioners a diagnostic before expensive retraining.

CCS Concepts

• **Applied computing** → **Computer games**; • **Computing methodologies** → *Machine learning*.

Keywords

constraint learnability, controllability, procedural content generation, discrete generative models, VQ-VAE, autoregressive transformer, Minecraft

ACM Reference Format:
Alex Chengyu Li. 2026. Which Structural Constraints Are Learnable? A Regime Map for a Minecraft Voxel Generator. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3815598.3815669

1 Introduction

A level designer wants towers to be tall and buildings to be symmetric, but which of these requests will the generator actually honor? Neural PCG systems now produce game levels [19] and 3D structures [7], yet control over their outputs is uneven. Some structural constraints respond to conditioning while others do not, even within the same model. Practitioners have no standard way to predict which constraints will fail before running experiments.

We ask: *given a neural PCG pipeline and a set of structural constraints, which constraints are learnable, and can we predict this from the training data alone?* Concretely, the model outputs new voxel buildings; learnability is measured by how much a conditioned sample distribution moves a structural property relative to unconditioned generation.

For example, height is a direct condition and reliably produces taller outputs. Hollowness is not an input token, but it is computable from each generated building; across our tested conditions, it changes little. The usual fix is to add training data or tune hyperparameters, which assumes the failure is data-limited. But some constraints are bounded by what the decoder can represent, not by data scarcity. Whether the bottleneck is in the data or in the representation determines the right response: more data, or a different architecture.

We study this in Minecraft building generation (32^3 voxel grids, 513 remapped block tokens) using a VQ-VAE [22] + autoregressive transformer with classifier-free guidance (CFG), conditioned on six structural features (§3). The architecture uses the same learned token mechanism for all conditions, with no hard-coded structural constraint loss or decoder rule.

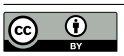
We make three contributions:

(1) A **pipeline-specific three-regime learnability map** across 14 structural properties with bootstrap 95% confidence intervals, classifying constraints as Controllable, Approachable, or Unresponsive (§4).
(2) A **composite correlation** between two pre-experiment features (effective signal and training CV) and relative responsiveness, with ρ= + 0.879 (permutation p=0.002, n=10) for emergent properties, proposed as a hypothesis for future validation (§5).
(3) A **dual bottleneck analysis** via CFG sensitivity, diagnosing representation ceilings versus frequency floors (§6).

2 Related Work

Discrete 3D generation. VoxelCNN [4] introduced autoregressive voxel generation on Minecraft structures from the 3D-Craft dataset. SAR3D [3] scales VQ-VAE + autoregressive generation to general 3D objects. Scaffold Diffusion [12] applies discrete diffusion to sparse multi-category Minecraft voxel structures. Minecraft PCG also includes the GDMC settlement generation competition [16], open-ended evolution for buildings [1], and interactive latent-variable evolution for structures [15]. These works demonstrate generation, search, and interaction in Minecraft; our focus is whether a fixed learned generator can reliably respond to structural constraints.

*Code, saved experimental results, and trained checkpoints are archived at DOI: 10.5281/zenodo.20821894.



FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/26/08
https://doi.org/10.1145/3815598.3815669

Controllable generation. MarioGPT [19] demonstrated text-conditioned level generation. DreamCraft [7] studies text-guided generation of functional Minecraft environments; projected diffusion [5] enforces hard constraints at inference. GAN-based generators have been used for conditional puzzle levels [10] and Doom levels [9]. PCGRL [14] frames level generation as reinforcement learning over design objectives. These works study controllable or objective-driven generation. We hold the mechanism fixed and ask which constraints respond.

Constraint satisfaction and expressive range. Other work handles constraints *explicitly*: WaveFunctionCollapse [13] via constraint propagation, Sturgeon [6] via learned and designer-provided constraints, and answer-set programming [17] via logical specifications. These methods can enforce constraints directly, but only for constraints expressible to their solver or formalism. Our regime map characterizes where learned (implicit) control suffices and where explicit methods remain necessary. Smith and Whitehead's expressive range analysis [18] characterizes procedural level generators. The regime map extends this idea to *conditional* generation by measuring responsiveness to conditioning signals and diagnosing the bottleneck (data vs. representation) when responsiveness is low. Quality-diversity methods [8] illuminate feature spaces via search, whereas the regime map identifies which dimensions a learned generator covers *without* search. The PCGML survey [20] sets the broader context for learned generators; our map asks a narrower per-constraint responsiveness question. Search-based PCG [21] treats content generation as optimization over explicit objectives.

3 Method

Existing buildings train the VQ-VAE and transformer and provide the training-build measurements used by the predictor. Regime labels are computed from generated outputs: sampled latent-code sequences are decoded into 32^3 voxel buildings, 14 structural properties are measured, and conditional means are compared with unconditioned generation.

3.1 Data

The dataset contains 10,310 Minecraft builds from three sources: 3D-Craft [4] (annotated human constructions), Minecraft Schematics Dataset [2] (community-sourced schematics), and additional community builds. We crop, pad, and remap each build to a 32^3 voxel grid using a project-specific 513-token voxel vocabulary: air plus the 512 most frequent non-air source tokens, with the mapping released alongside the code. The builds include houses, towers, and larger complexes, but the training set is naturally imbalanced: symmetric structures make up only 3.9% and enclosed structures only 2.4%.

3.2 Architecture

To instantiate the diagnostic rather than propose a new generator architecture, we use a fixed VQ-VAE [22] (3D conv, 2048-entry codebook, 256d codes, ~40M params) that maps each 32^3 grid to an 8^3 array of discrete latent codes over 100K training steps; all 2048 codebook entries are active across the training set. An autoregressive transformer (512d, 12 layers, 8 heads, ~40M params) models and samples these 512 latent codes in raster-scan order with 3D sinusoidal positional encodings. The completed code grid is decoded back into a 32^3 voxel building by the VQ-VAE decoder. The transformer is trained for 80K steps (batch size 32, learning rate 2×10^{-4}, cosine decay). Raster-scan order means each token can attend only to spatially preceding positions; spatially adjacent but sequentially later tokens are invisible, which limits the model's ability to coordinate global properties such as symmetry.

3.3 Structural Conditioning

Six structural conditioning tokens are discretized from training-build measurements: height (y-span buckets ≤8, 16, 24, >24), size (non-air block-count buckets ≤100, 500, 2000, >2000), footprint (XZ bounding-box area ≤100, 400, >400), symmetry (X-axis mirror IoU >0.7), enclosure (flood-filled enclosed-air ratio >0.05), and complexity (surface-area ratio <1.5, ≤3, >3). Only these six tokens are passed to the generator; the evaluated properties below remain continuous, counts, or ratios measured after generation. Classifier-free guidance (CFG) [11] drops all conditioning with 10% probability during training; at inference, unconditional and conditional logits ℓ_u, ℓ_c are interpolated as $\hat{\ell} = \ell_u + s(\ell_c - \ell_u)$ with guidance scale s (default 2).

3.4 Evaluation Protocol

Training-data statistics are computed before generation: direct-condition frequencies use all 10,310 filtered builds; CVs and proxy correlations use a fixed 200-build measured training sample. Regime labels are computed from generated outputs: for seven configurations (unconditioned plus six probes; unspecified slots use the learned "any" mask), 40 samples are generated (5 seeds × 8 per seed) and 14 structural properties are measured per output. **Controllability** is a relative responsiveness score, $\text{ctrl}(p) = \max_{c\in C} |\mu_{p,c} - \mu_{p,\emptyset}|/\max(|\mu_{p,\emptyset}|, \epsilon)$, where $\mu_{p,c}$ is the mean of property p under condition c and $\mu_{p,\emptyset}$ is the unconditional mean (ϵ=0.001 prevents division by zero). A score above 100% means the conditional mean shifted by more than the unconditioned mean; it does not imply hard constraint satisfaction on every sample. Bootstrap 95% CIs use 10,000 replicates resampled from the 40 samples per condition. Regime thresholds are fixed throughout: >100% → Controllable, 20–100% → Approachable, <20% → Unresponsive.

4 Regime Map

The 14 measured properties span local, semi-local, and global structure: height (Y-span), `n_blocks` (non-air count), `bbox_volume` (bounding-box volume), `vertical_aspect` (height / max horizontal span), `surface_ratio` (blocks with an air neighbor / blocks), `floor_count` (Y-layers filling >10% of footprint), `elongation` (max/min XZ span), `hollowness` (air fraction inside the bounding box), `layer_consist.` (mean IoU of adjacent Y-layers), `fp_convexity` (filled XZ footprint / XZ box), X- and Z-axis mirror IoU, `enclosed_ratio` (flood-filled enclosed air / air), and 6-connected component count. Of the six conditioning features (§3.3), four have a direct property counterpart (`height`, `n_blocks` via size, `symmetry_iou` via symmetry, `enclosed_ratio` via enclosure); footprint and complexity affect multiple properties but lack a one-to-one mapping. The remaining 10 properties are emergent, responding only through correlation with conditioned

Table 1: Constraint learnability regime map (CFG scale s=2, 5 seeds × 8 samples per condition). Dir.: has a conditioning token. Eff.: effective signal (target-class frequency for direct; max proxy correlation for emergent). CV: training coefficient of variation. Ctrl%: relative responsiveness (max conditional mean shift) with bootstrap 95% CI.

Property	Dir.	Eff.	CV	Ctrl% [95% CI]	Regime
Controllable (>100% shift)					
height	✓	.068	0.60	266 [197, 360]	C*
n_blocks	✓	.117	1.03	632 [394, 1050]	C*
symmetry_iou	✓	.039	1.33	237 [122, 492]	C*
enclosed_ratio†	✓	.024	5.37	564 [132, 1141]	C
floor_count		.765	0.76	182 [116, 269]	C*
bbox_volume		.750	1.03	422 [251, 725]	C*
connected_comp.‡		.090	1.44	128 [58, 233]	C
vertical_aspect		.784	0.61	333 [245, 446]	C*
z_symmetry_iou		.469	0.93	324 [161, 753]	C*
Approachable (20–100% shift)					
elongation		.179	1.76	64 [29, 105]	A
surface_ratio		.400	0.08	32 [27, 37]	A*
layer_consist.		.320	0.39	44 [29, 62]	A*
fp_convexity		.437	0.37	51 [34, 71]	A*
Unresponsive (<20% shift)					
hollowness		.311	0.23	14 [8, 19.9]	U*

*Regime assignment is *confident*: 95% CI lies entirely within one regime (enclosed_ratio excluded; see †).
†enclosed_ratio: percentage inflated by near-zero baseline (μ_0=0.00003); absolute shift is only ∼0.006. Per-seed controllability ranges from 73% to 1119% (SD ≈ 462%); classification unreliable.
‡CI spans the C/A boundary; classified by point estimate (see text).

features. Table 1 separates quantities known before generation from outcomes measured after generation: **Eff.** and **CV** come from training-build measurements, while **Ctrl%** and the regime label come from the generated samples.

Observations. All four directly conditioned properties are Controllable (n=4, insufficient to generalize). Five emergent properties also reach Controllable through proxy correlation alone; `connected_comp.` is borderline (CI spans the C/A boundary). The largest cross-axis transfer is `z_symmetry_iou` (+324%): X-axis symmetry conditioning increases Z-axis symmetry by over 3×, likely driven by correlated training data (symmetric builds tend to be symmetric on both axes), though partial preservation of symmetry in the latent space cannot be ruled out.

The single Unresponsive property (`hollowness`, 14% [8%, 19.9%]) has moderate proxy correlation (0.31) but very low training CV (0.23): the measured training sample shows little variation along this axis for the model to learn a responsive representation.

Robustness. Confidence markers distinguish assignments whose bootstrap CI lies within one regime from borderline cases. Varying boundaries from (10%, 80%) to (30%, 120%) changes at most one assignment. Under CI-conservative classification, 11 properties are confidently assigned (7 Controllable, 3 Approachable, 1 Unresponsive); 3 remain borderline: `enclosed_ratio` (CI above 100% but unreliable due to near-zero baseline), `connected_comp.` (CI [58%, 233%] spans C/A), and `elongation` (CI [29%, 105%] spans A/C).

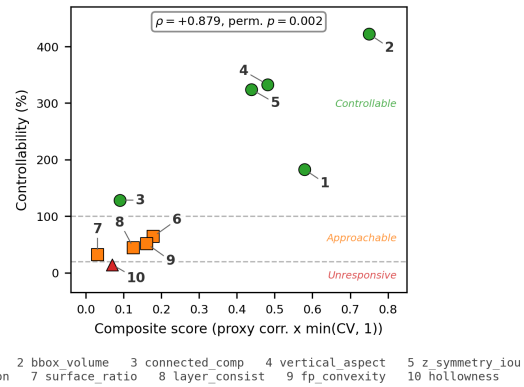


Figure 1: Composite score (proxy corr. × min(CV, 1)) vs. controllability (n=10 emergent properties). Neither proxy correlation (ρ=0.624, permutation p=0.060) nor CV (ρ=0.636, p=0.053) alone reaches significance; their product does (ρ=0.879, p=0.002).

5 Predictive Framework

For emergent properties, we test whether relative responsiveness *correlates* with features computable from training data alone, as a hypothesis for future validation on other architectures:

- **Effective signal**: For directly conditioned properties, the target-class frequency in training data. For emergent properties, the maximum absolute Pearson correlation with any conditioned property (proxy correlation).
- **Training CV**: The coefficient of variation of the property across the measured training sample, measuring how much variation the model has opportunity to learn.

5.1 Direct Conditioning

All four directly conditioned properties are Controllable despite a ∼5× range in training frequency (2.4%–11.7%), consistent with CFG amplification compensating for low frequency. However, n=4 with zero negative cases is too small to generalize (rule-of-three 95% upper bound on failure rate: 3/n=75%). `enclosed_ratio` (564%) reflects a near-zero baseline (μ=0.00003; absolute shift ≈0.006; see Table 1 footnote).

5.2 Emergent Property Predictor

For the 10 emergent properties, relative responsiveness is better aligned with the combination of proxy correlation and training CV than with either feature alone. A composite score, signal × min(CV, 1), captures this association (Figure 1; Spearman ρ; controllability values permuted among properties, 10,000 replicates): Each point in Figure 1 is one emergent property from Table 1; the x-axis is computed before generation from training-build measurements, while the y-axis is measured on generated samples.

We compare three simple formulations (CV alone, proxy correlation alone, and their product); only the composite reaches significance under permutation testing (p=0.002; Bonferroni-adjusted p=0.005 across three tests). Post hoc alternatives are comparable: signal × CV (ρ=0.915) and signal × log(1+CV) (ρ=0.879); we report

Table 2: CFG sensitivity (3 seeds × 8 samples, 4 conditions). Controllability (%) at three guidance scales. Bold: only observed regime transition. §Approachable in Table 1.

Property	s=0	s=2	s=4	Pattern
height	132	224	240	C→C
n_blocks	243	569	603	C→C
symmetry_iou	149	190	242	C→C
enclosed_ratio	158	542	328	C→C
floor_count	**88**	**167**	**197**	**A→C**
elongation	37	40	30	A→A
connected_comp.	57	94	75	A→A
layer_consist.	24	31	38	A→A
fp_convexity	26	45	52	A→A
hollowness	6	12	19	U→U
surface_ratio§	13	15	14	U→U

Note: Fewer seeds (3 vs. 5) and conditions (4 vs. 7) than Table 1; values at s=2 are not directly comparable, and three Controllable emergent properties are omitted. `surface_ratio` appears Unresponsive here (15%) but Approachable (32%) because its best-shift condition is absent.

min(CV, 1) as the most conservative variant. At n=10, formulations cannot be distinguished reliably, and the 95% CI on ρ=0.879 ([0.55, 0.97]) remains compatible with either a moderate or strong relationship. The fitted interaction is multiplicative: weak proxy signal and low training CV give different failure modes. Hollowness (proxy correlation 0.31, CV 0.23) has moderate proxy correlation but too little training CV; connected components (proxy correlation 0.09, CV 1.44) has high training CV but almost no correlation pathway.

A discrete decision tree achieves 90% training accuracy but only 60% three-class LOO accuracy (6/10 correct; majority-class baseline 50%), so the continuous correlation is the primary result.

6 Dual Bottleneck Analysis

The map shows which properties respond, but not why: low responsiveness can reflect a weak conditioning pathway (**frequency floor**) or a decoder/latent-space limit (**representation ceiling**).

These are probed by varying the CFG scale ($s \in \{0, 2, 4\}$, where s=0 disables CFG amplification while retaining the condition prefix) at inference without retraining. A frequency-limited property should strengthen as s increases; a representation-limited one should change little.

Table 2 shows three patterns. *Weak amplification:* `hollowness` responds monotonically (6%, 12%, 19% at s=0, 2, 4) but stays below 20%; whether stronger guidance would cross the threshold remains untested. `surface_ratio` shows negligible response (13%, 15%, 14%). *Frequency floor:* `floor_count` crosses from A to C (88%, 167%, 197%). `enclosed_ratio` responds non-monotonically (158%, 542%, 328%), possibly reflecting known guidance tradeoffs [11]. *Stable Approachable:* the remaining properties stay within 20–100% across all guidance scales.

The `floor_count` transition suggests that some frequency-floor constraints may respond to stronger guidance, though only one such case was observed. Representation-ceiling constraints require changes to the decoder or latent space. Figure 2 illustrates the amplification effect.

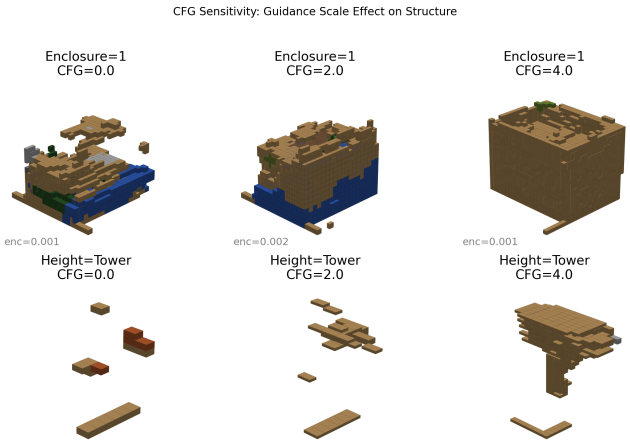


Figure 2: CFG sensitivity at scales s=0, 2, 4. Enclosure illustrates guidance-sensitive structure changes. Height at s=0 produces fragmented outputs; at s=4, conditioning is stronger but block-type diversity decreases.

7 Discussion and Conclusion

Implications for practitioners. In this pipeline, standard CFG conditioning sufficed for **Controllable** properties. **Approachable** properties may indicate a frequency floor and may benefit from data augmentation, stronger CFG, or targeted fine-tuning. **Unresponsive** properties may require changes to the decoder or latent space, or practitioners can layer explicit constraint methods [5, 13, 21] on top of the learned generator.

Limitations. (1) Small, heterogeneous data and a single pipeline. The model is trained on 10,310 filtered 32^3 builds (2,028 3D-Craft, 173 text2mc, 8,109 rom1504) selected from 13,854 processed candidates; this is small for neural 3D generation and rare conditions, so outputs often capture coarse silhouettes rather than human-authored fidelity. (2) n=10 emergent properties is small. The composite (ρ=0.879, CI ≈[0.55, 0.97]) is a hypothesis, not a validated predictor. (3) Zero negative cases for direct conditioning (n=4). (4) Percentage shifts mislead for near-zero baselines (`enclosed_ratio`: 564% but only 0.006 absolute). Regime assignments near boundaries depend on which conditions are tested (`surface_ratio`: Approachable with 7 conditions, Unresponsive with 4). (5) The 40 per-condition samples are clustered within 5 seeds (8 per seed); the bootstrap resamples individual samples, which may understate variance if within-seed correlation is high. (6) The metrics evaluate structural responsiveness, not perceptual quality; visual quality requires separate evaluation.

Conclusion. In this pipeline, constraint learnability is associated with the product of effective signal and training CV (ρ=0.879, p=0.002), while CFG sensitivity distinguishes frequency floors from representation ceilings. The result is pipeline-specific; the diagnostic method should be tested on larger datasets, alternative architectures, and other content domains.

References

[1] Matthew Barthet, Antonios Liapis, and Georgios N. Yannakakis. 2023. Open-Ended Evolution for Minecraft Building Generation. *IEEE Transactions on Games*

15, 4 (2023), 603–612. doi:10.1109/TG.2022.3189426

[2] Romain Beaumont. 2020. Minecraft Schematics Dataset. https://github.com/rom1504/minecraft-schematics-dataset.

[3] Yongwei Chen, Yushi Lan, Shangchen Zhou, Tengfei Wang, and Xingang Pan. 2025. SAR3D: Autoregressive 3D Object Generation and Understanding via Multi-Scale 3D VQVAE. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:2411.16856.

[4] Zhuoyuan Chen, Demi Guo, Tong Xiao, Saining Xie, Xinlei Chen, Haonan Yu, Jonathan Gray, Kavya Srinet, Haoqi Fan, Jerry Ma, Charles R. Qi, Shubham Tulsiani, Arthur Szlam, and C. Lawrence Zitnick. 2019. Order-Aware Generative Modeling Using the 3D-Craft Dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 1764–1773. doi:10.1109/ICCV.2019.00185

[5] Jacob K. Christopher, Stephen Baek, and Ferdinando Fioretto. 2024. Constrained Synthesis with Projected Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37. 89307–89333.

[6] Seth Cooper. 2022. Sturgeon: Tile-Based Procedural Level Generation via Learned and Designed Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, Vol. 18. 26–36. doi:10.1609/aiide.v18i1.21944

[7] Sam Earle, Filippos Kokkinos, Yuhe Nie, Julian Togelius, and Roberta Raileanu. 2024. DreamCraft: Text-Guided Generation of Functional 3D Environments in Minecraft. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG)*. ACM, 1–15. doi:10.1145/3649921.3649943

[8] Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. 2021. Illuminating Mario Scenes in the Latent Space of a Generative Adversarial Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 5922–5930. doi:10.1609/aaai.v35i7.16740

[9] Edoardo Giacomello, Pier Luca Lanzi, and Daniele Loiacono. 2018. DOOM Level Generation Using Generative Adversarial Networks. In *Proceedings of the IEEE Games, Entertainment, Media Conference (GEM)*. 316–323. doi:10.1109/GEM.2018.8516539

[10] Andreas Hald, Jens Stuckermann Hansen, Jeppe Theiss Kristensen, and Paolo Burelli. 2020. Procedural Content Generation of Puzzle Games using Conditional Generative Adversarial Networks. In *Proceedings of the 15th International Conference on the Foundations of Digital Games (FDG)*. ACM, Article 99, 4 pages. doi:10.1145/3402942.3409601

[11] Jonathan Ho and Tim Salimans. 2021. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[12] Justin Jung. 2025. Scaffold Diffusion: Sparse Multi-Category Voxel Structure Generation with Discrete Diffusion. arXiv preprint arXiv:2509.00062.

[13] Isaac Karth and Adam M. Smith. 2017. WaveFunctionCollapse is Constraint Solving in the Wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG)*. ACM, 1–10. doi:10.1145/3102071.3110566

[14] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. PCGRL: Procedural Content Generation via Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, Vol. 16. 95–101. doi:10.1609/aiide.v16i1.7416

[15] Timothy Merino, Michael Charity, and Julian Togelius. 2023. Interactive Latent Variable Evolution for the Generation of Minecraft Structures. In *Proceedings of the 18th International Conference on the Foundations of Digital Games (FDG)*. ACM, 1–8. doi:10.1145/3582437.3587208

[16] Christoph Salge, Michael Cerny Green, Rodrigo Canaan, and Julian Togelius. 2018. Generative Design in Minecraft (GDMC). In *Proceedings of the 13th International Conference on the Foundations of Digital Games (FDG)*. ACM, 1–10. doi:10.1145/3235765.3235814

[17] Adam M. Smith and Michael Mateas. 2011. Answer Set Programming for Procedural Content Generation: A Design Space Approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 187–200. doi:10.1109/TCIAIG.2011.2158545

[18] Gillian Smith and Jim Whitehead. 2010. Analyzing the Expressive Range of a Level Generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games (PCGames)*. ACM, 1–7. doi:10.1145/1814256.1814260

[19] Shyam Sudhakaran, Miguel González-Duque, Matthias Freiberger, Claire Glanois, Elias Najarro, and Sebastian Risi. 2023. MarioGPT: Open-Ended Text2Level Generation through Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[20] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270. doi:10.1109/TG.2018.2846639

[21] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186. doi:10.1109/TCIAIG.2011.2148116

[22] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.