Moonshine:Game Content Generators into Steerable Generative Models

s1312002 EINHEUSER Kiran

June 10th 2026

Paper Information

Authors:

- Yuhe Nie - New York University
- Michael Middleton - New York University
- Tim Merino - New York University
- Nidhushan Kanagaraja - New York University
- Ashutosh Kumar - New York University
- Zhan Zhuang - City University of Hong Kong
- Julian Togelius - New York University

Venue:

- AAAI Conference on Artificial Intelligence (AAAI 2025)

Published:

- April 11th 2025

Motivation

Problem:

- Procedural Content Generation (PCG) is widely used in games.
- Traditional generators can create content, but are difficult to control.
- PCGML (Machine Learning based PCG) offers flexibility, but requires large labeled datasets.
- Human labeling is expensive and time consuming.

Goal:

Create a controllable text-to-map generator without requiring manual annotation.

Key Idea: Moonshine

The authors propose a pipeline called **Moonshine**.

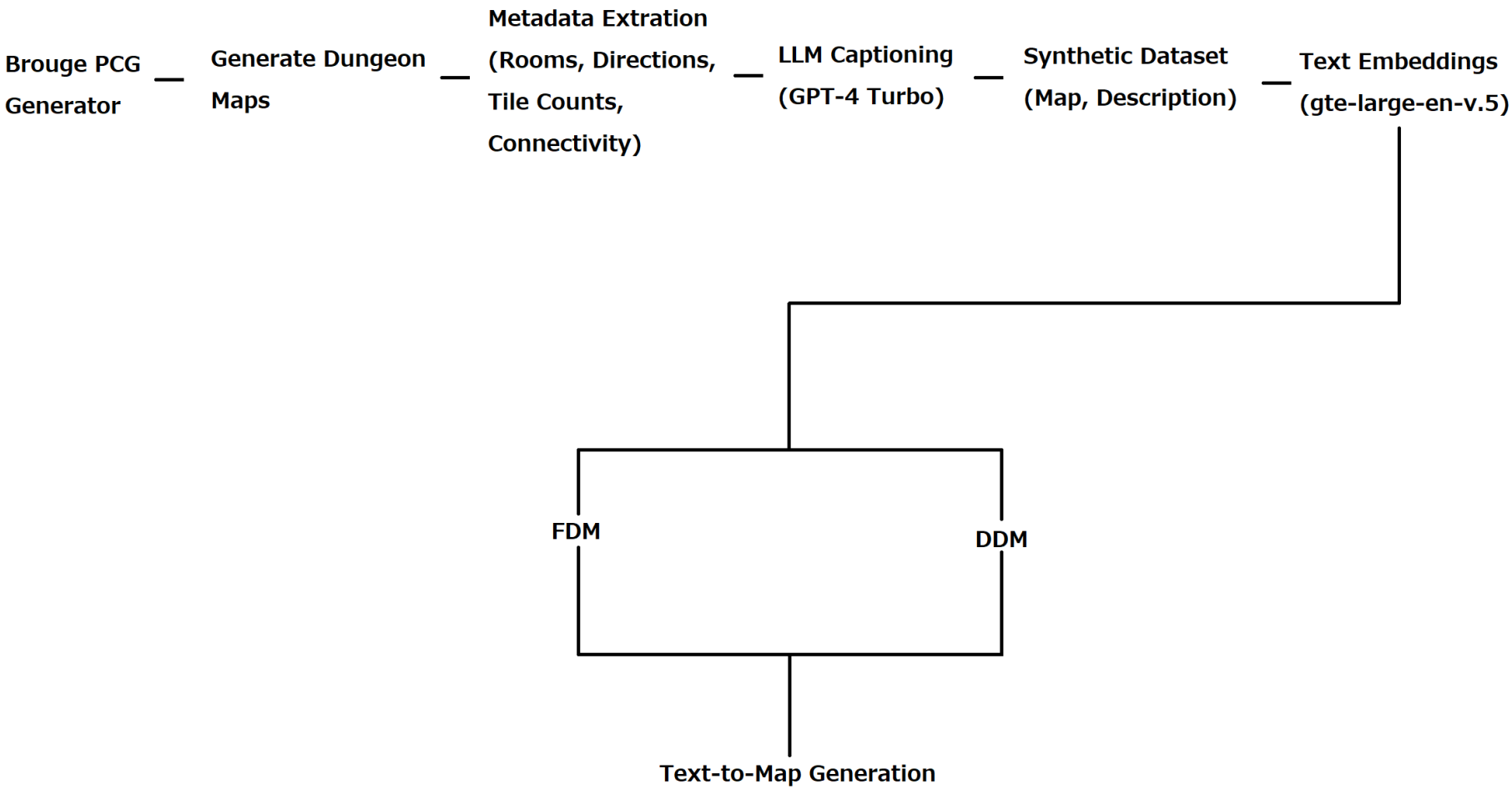
1. Generate maps using an existing PCG algorithm.
2. Analyze each map automatically.
3. Use an LLM to generate textual descriptions.
4. Train a text-conditioned generative model.

Result:

$$\text{Traditional PCG Generator} \rightarrow \text{Synthetic Dataset} \rightarrow \text{Text-to-Map Generator}$$

This can be viewed as **knowledge distillation from an algorithm into a neural network**.

Moonshine Framework



Dataset Creation

Source game:

- Open-source roguelike **Brogue**

Map representation:

- 32×32 grids
- 14 terrain types

Dataset size:

- 49,000 training maps
- 14,000 testing maps
- 7,000 validation maps

Examples of terrain:

- Water, Lava, Sand, Grass, Fungus, Stone

Tile	Desc.	Tile	Desc.	Tile	Desc.
	Ashes		Bog		Bridge
	Crystal		Fire		Fungus
	Grass		Ground		Ice
	Lava		None		Sand
	Stone		Water		

Table 1: The details of dungeon tileset with descriptions.

Synthetic Description Generation

For each generated map:

1. Detect rooms and pathways.
2. Compute metadata:
   - room positions
   - tile counts
   - connectivity
3. Feed metadata into an LLM.
4. Generate multiple natural-language descriptions.

Example:

> *"A central aquatic zone surrounded by land, dotted with fungus and occasional crystals."*

Text-to-Game-Map (T2M)

Inspired by Text-to-Image generation.

Input:

- Natural language description

Output:

- Dungeon map

Difference from images:

- Images use continuous pixel values.
- Maps use discrete tile categories.

The task is therefore:

$$\text{Text} \rightarrow \text{Grid of Terrain Tiles}$$

Model 1: Five-Dollar Model (FDM)

Characteristics:

- Feed-forward neural network.
- Uses text embeddings as input.
- Directly predicts the entire map.

Advantages:

- Simple.
- Fast.
- Good semantic accuracy.

Disadvantages:

- Low diversity.
- Same prompt often produces similar maps.

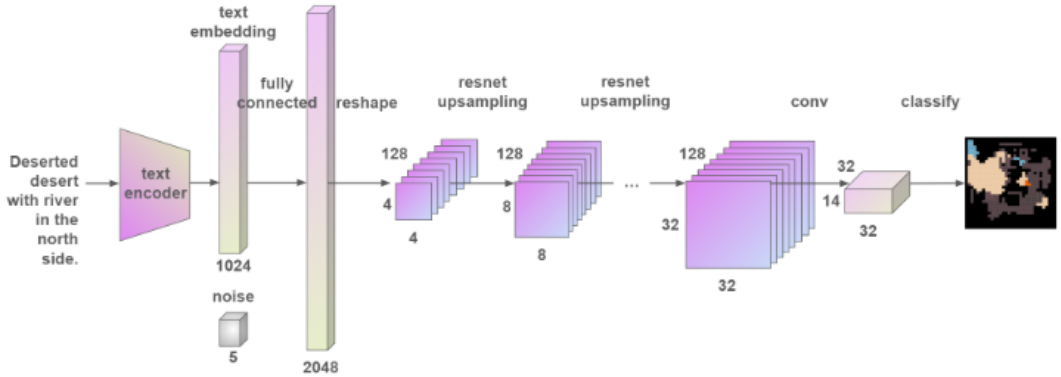


Figure 2: Five-Dollar Model architecture.

Model 2: Discrete Diffusion Model (DDM)

Based on diffusion models.

Generation process:

1. Start with random noise.
2. Iteratively denoise.
3. Condition each step on text embeddings.

Advantages:

- Higher diversity.
- Generates varied maps from the same prompt.

Disadvantages:

- More expensive to train.
- Slightly lower precision in some cases.

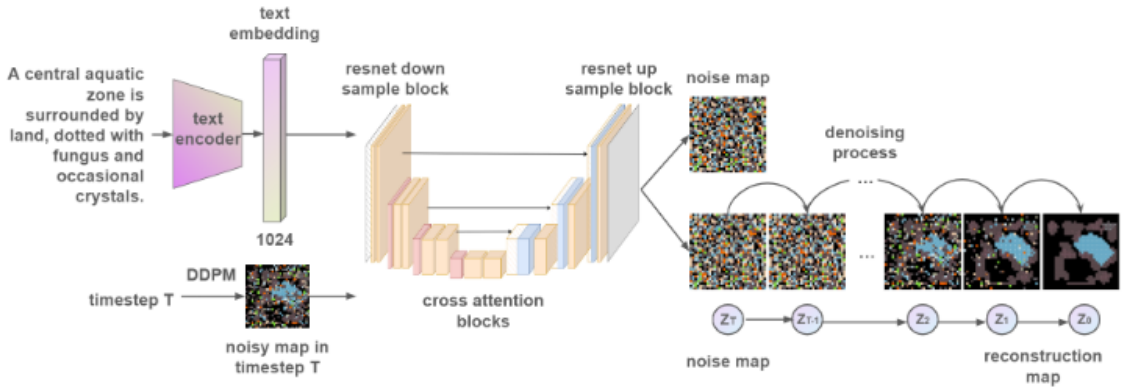


Figure 3: Discrete Diffusion Model architecture.

Results

Description quality evaluation:

- BLEU
- ROUGE-L
- METEOR
- SPICE

Key finding:

Observation

Long LLM-generated descriptions consistently outperformed short descriptions.

Long descriptions better captured:

- semantics
- map structure
- human writing style

Results

S.No	Metric	Des. Long	Des. Short
0	Bleu_1	54.71	**58.37**
1	Bleu_2	**26.57**	26.56
2	Bleu_3	**12.18**	11.68
3	Bleu_4	**05.62**	04.86
4	METEOR	**19.50**	12.64
5	ROUGE_L	**33.27**	26.16
6	SPICE	**11.31**	03.89

Table 2: Comparison within generated descriptions.

S.No	Metric	Des. Long	Des. Short
0	Bleu_1	**70.84**	50.58
1	Bleu_2	**43.47**	21.13
2	Bleu_3	**23.56**	07.88
3	Bleu_4	**11.49**	02.38
4	METEOR	**16.56**	09.38
5	ROUGE_L	**42.92**	23.25
6	SPICE	**03.43**	00.77

Table 3: Comparison between generated descriptions and human references.

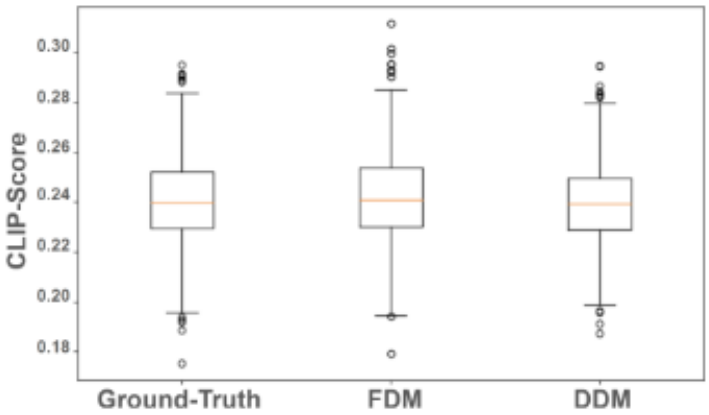


Figure 4: CLIP score of FDM (left) and DDM (right) v.s. the Brogue ground truth (X-axis).

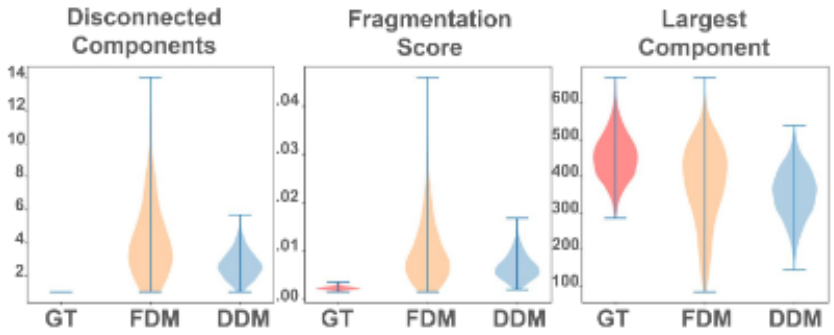


Figure 5: Connectivity analysis of models based on three metrics: the number of disconnected components, fragmentation score, and largest component size. DDM has fewer disconnected components, lower fragmentation, and more stable component sizes compared to FDM.

Map Generation Results

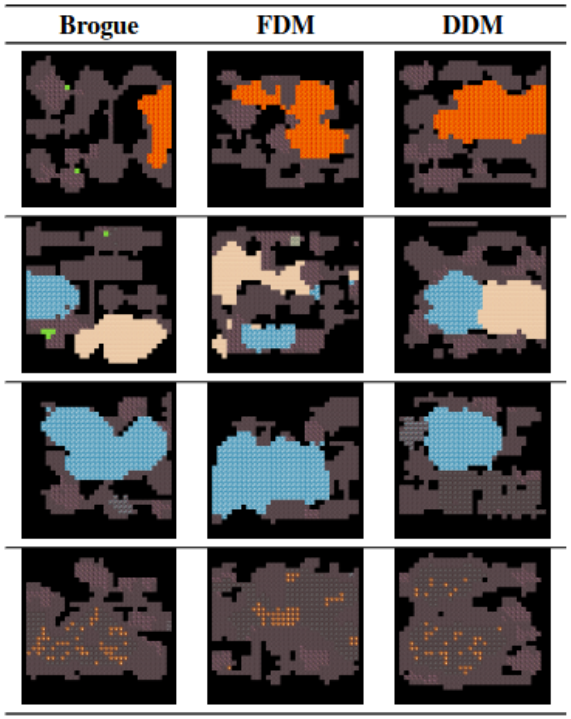


Table 4: Generated maps based on generated descriptions: (1) A central area dominated by a large expanse of lava surrounded by solid ground, with sparsely growing fungi and a few patches of grass; (2) A vast desert landscape merges with clear blue waters, spotted with fungus growths and small patches of greenery amidst the sandy terrain; (3) A central aquatic zone is surrounded by land, dotted with fungus and occasional crystals. Two smaller regions lie on the eastern edge, connected by pathways to the main area; (4) A charred landscape with scorched earth and rampant wildfires spread across. Patches of fungi cling to life amidst the destruction, with rare mineral-rich stones scattered sparsely.

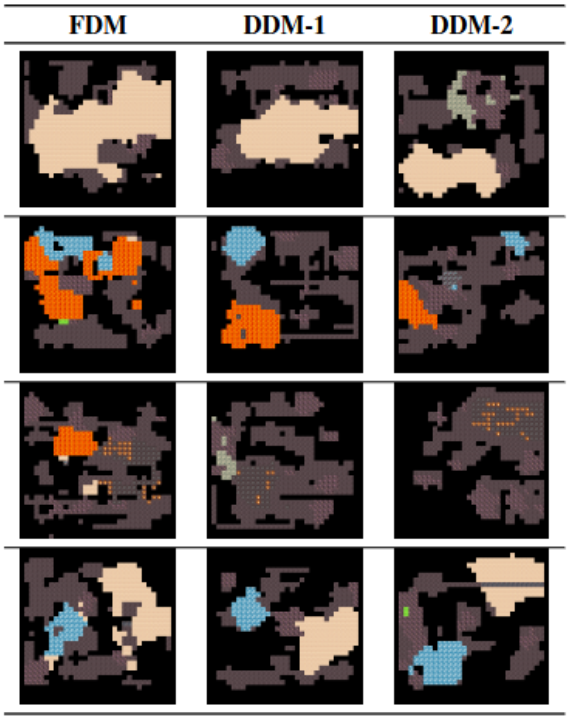


Table 5: Generated maps based on human prompts: (1) A vast sandy area; (2) Some lakes in the north, and a lot of magma and lava; (3) Burning out fire in the center; (4) The lake to the left. The desert to the right. Connected by bridges.

Map Generation Results

FDM

- Better prompt accuracy
- Less variation

DDM

- More diverse maps
- Better structural variety
- Fewer disconnected components

Conclusion:

Trade-off

FDM = Accuracy
DDM = Diversity

Discussion and Critique

Strengths:

- Eliminates costly manual labeling.
- Clever use of LLM-generated synthetic data.
- Bridges traditional PCG and modern generative AI.

Limitations:

- Relies on quality of LLM descriptions.
- Fine-grained control remains difficult.
- Only terrain generation is considered.

Future work:

- Enemies
- Items
- Stories and quests
- Interactive designer feedback

Takeaways

1. Moonshine distills a traditional PCG algorithm into a controllable AI model.
2. LLMs generate synthetic labels automatically.
3. Text-to-Game-Map (T2M) is introduced as a new task.
4. FDM offers better accuracy.
5. DDM offers better diversity.

Moonshine demonstrates a practical path toward controllable AI-assisted game content generation.

Questions?