Generate Diverse Skills with Large Language Models

Kaijie Xu
Mcgill University
Montreal, Canada
kaijie.xu2@mail.mcgill.ca

Clark Verbrugge
McGill University
Montreal, Canada
clump@cs.mcgill.ca

Abstract

Roguelike and auto-battler games derive much of their replayability from randomized progression, rewards, and encounter sequencing, yet this randomness rarely extends to genuinely diverse game mechanics and content such as skills or combat units. Existing procedural systems for such content usually depend on fixed rule spaces, hand-authored templates, or proxy spell engines, which limit their ability to produce outputs that are semantically meaningful, valuable in play, and sufficiently expressive. We present Neural Spore Genesis, a browser-based roguelike auto-battler in which players buy elemental cells, arrange them on a 5×5 grid, and compile that spatial layout into a combat skill for auto-battle. To support this task, we build a high-expressivity structured skill framework covering a broad range of 2D auto-battler combat behaviors, and evaluate local open-weight language models against deterministic, template-based, random, and calibrated sampler baselines. Across 6 generator conditions and a 105-configuration corpus evaluated with 11 metrics, local LLMs achieve near-perfect element coherence while maintaining high intra-config diversity, a combination the baselines do not match. This work offers a practical evaluation environment for intent-aware procedural mechanic generation and supports future research on semantically grounded content systems for high randomness games.

Keywords

Procedural Content Generation, Game Mechanics, Large Language Models, Skill Generation, Evaluation Framework

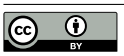
ACM Reference Format:
Kaijie Xu and Clark Verbrugge. 2026. Generate Diverse Skills with Large Language Models. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3815598.3815673

1 Introduction

Roguelike and auto-battler games remain compelling because each run recombines uncertainty, build adaptation, and emergent interactions into a fresh play experience. In practice, however, the skills, units, and synergies that define these games are still largely hand-authored, since unconstrained randomness and conventional PCG pipelines often produce content that is mechanically noisy, strategically unreadable, or disconnected from what the player was trying to build [7, 18]. *Noita*[1] shows how a rich spell engine can support an enormous combinatorial space of effects by letting players assemble spells from reusable components, but its pleasure comes from deliberate manual construction that resembles programming, requires substantial system knowledge, and remains bounded by a fixed authored ruleset. Recent work [25] studies the expressive range of such spell engines directly, yet these systems still depend on fixed generators that explore a predefined possibility space instead of responding to player-provided spatial semantic information inside a highly stochastic game loop. This motivates our central question: within a bounded executable skill schema, can a high-randomness game generate meaningfully diverse skills while preserving the observable spatial and elemental semantics of player-expressed intent?

To investigate this question, we built Neural Spore Genesis, a browser-based roguelike auto-battler in which players buy elemental cells, arrange them on a 5×5 grid, and fight using the skill compiled from that layout. Although instantiated in a simplified 2D/2.5D real-time combat setting, we use this game as a stylized testbed for a broader class of modular skill-engine games (e.g., roguelikes and auto-battlers) that offer players access to a large set of diverse, reusable skills and require the system to realize semantic information under strong runtime randomness while preserving long-term replayability. Within this environment, we designed a high-expressivity skill system and runtime protocol, and used local open-weight language models as semantic compilers that map spatial cell arrangements to executable skill data. We then compare these generators against deterministic, template-based, random, and calibrated sampler baselines over a 105-configuration grid corpus using 11 evaluation metrics. The experiments show that local open-weight LLMs achieve near-perfect element coherence and high intra-config diversity simultaneously, a quality profile that deterministic, template-based, and calibrated sampler generators cannot reproduce. Our goal is to provide a replicable framework for studying how procedural skill generators can map spatial/semantic information to executable mechanics in games with strong runtime randomness. Our contributions are:

- **Experimental game framework:** Neural Spore Genesis, a playable browser-based roguelike auto-battler in which a semantic compiler pipeline converts player-constructed elemental cell arrangements into executable combat skills at runtime, providing a controlled environment for intent-aware procedural generation research.
- **Skill generation and evaluation framework:** A structured skill schema with 11 action types and 11 modifier types, a 105-configuration grid corpus spanning 7 spatial



FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/26/08
https://doi.org/10.1145/3815598.3815673

[1] https://noitagame.com

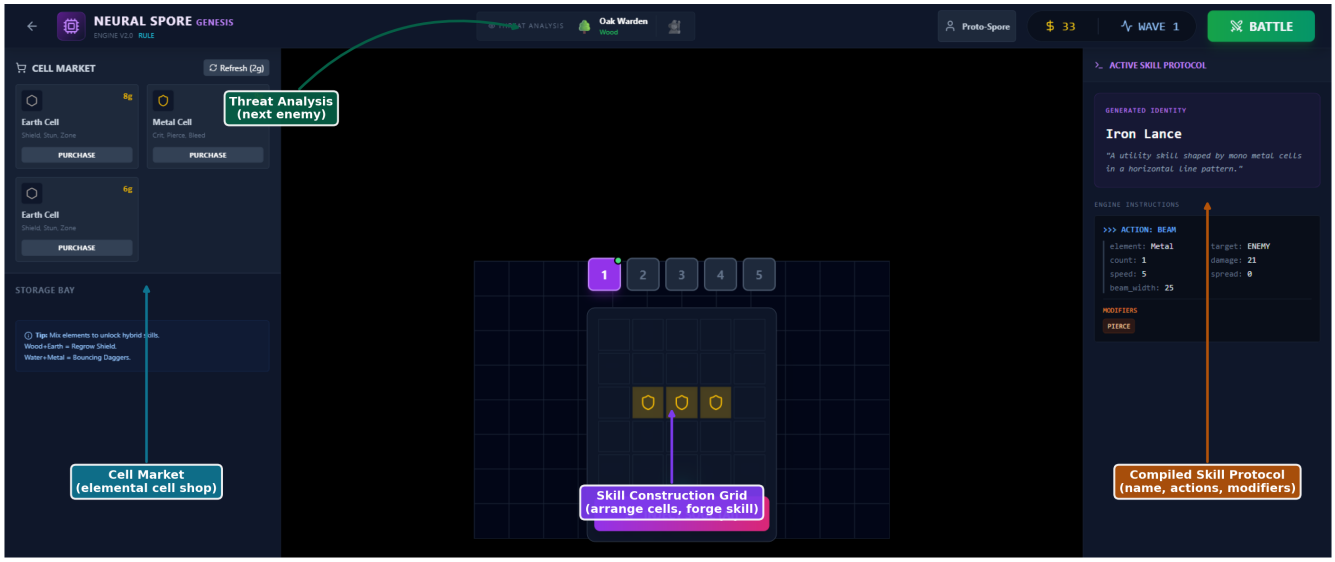


Figure 1: In-game interface and play loop snapshot. The left panel is the cell market and inventory, the center is the 5×5 skill-construction grid and combat arena, and the right panel shows the compiled skill protocol used by the runtime executor.

shapes and 5 elemental types, and 11 evaluation metrics covering structural validity, semantic coherence, and creative diversity.

- **Comparative evaluation:** Systematic comparison of 6 generators spanning different conditions, with a temperature sensitivity ablation study, indicating that instruction-tuned open-weight LLMs break the coherence-diversity tradeoff that limits all baseline approaches.

2 Related Work

Procedural Content Generation has long studied how algorithmic systems can expand a game's content space through grammars, search, simulation, and learned policies [7, 11, 18]. Expressive Range Analysis established a standard way to characterize what a generator can produce [20], while later work argued that PCG can also serve as a game design instrument for creating new forms of play [19]. Grid-structured generation has been explored through methods such as PCGRL [9], and template-driven content authoring remains common in production-adjacent settings [23]. These works are important foundations, but most focus on levels, quests, or other assets whose content space is specified in advance.

Game mechanics and spell systems expose a related but distinct problem: how to generate effects that remain executable, legible, and strategically meaningful. Commercial examples such as *Noita* show that a well-designed spell engine can support a large combinatorial space of interactions, while recent work has begun to study that possibility space directly through large-scale simulation and analysis [25]. Even so, these approaches still operate inside fixed effect vocabularies and predefined composition logics. They reveal the expressive capacity of an engine, but they do not address how a generator should interpret a player's spatial construction as an intent-bearing signal during play.

LLM-based game generation has expanded rapidly in the last two years [6, 13, 28]. Prior studies examined tile-based level generation and prompt generalization [14, 22, 26, 27], text-guided platformer design [21], practical PCG pipelines using frontier language models [15], and rule or game-specification generation from language [5, 8, 30]. Work on game-playing and authoring support has also shown that language models can reason over structured game states and assist creative workflows [1]. However, most existing studies use text as the primary control channel or evaluate spatial constraint satisfaction at the level-design level, leaving the generation of executable combat mechanics from player-built spatial input largely unexplored.

Mixed-initiative and player-centered PCG research asks how generators can respond to human goals instead of producing content in isolation [3, 4, 12]. Surveys of mixed-initiative creation emphasize collaboration, interpretability, and designer control as central concerns [10]. Adjacent work on player modeling and adaptive content generation infers preferences from gameplay traces, online feedback, or observed behavior [2, 16, 29]. These systems improve personalization, but they usually respond after the fact, once the system has already observed what the player did over time.

Our work extends these studies by treating a player-constructed spatial layout as an immediate semantic input for procedural mechanic generation. We study executable skill generation instead of level generation, evaluate semantic grounding alongside diversity and validity, and compare language-model-based semantic compilation against other baseline generators inside a playable game environment. This framing moves procedural generation closer to a setting where the system must interpret what the player is trying to build and produce mechanically meaningful content at runtime.

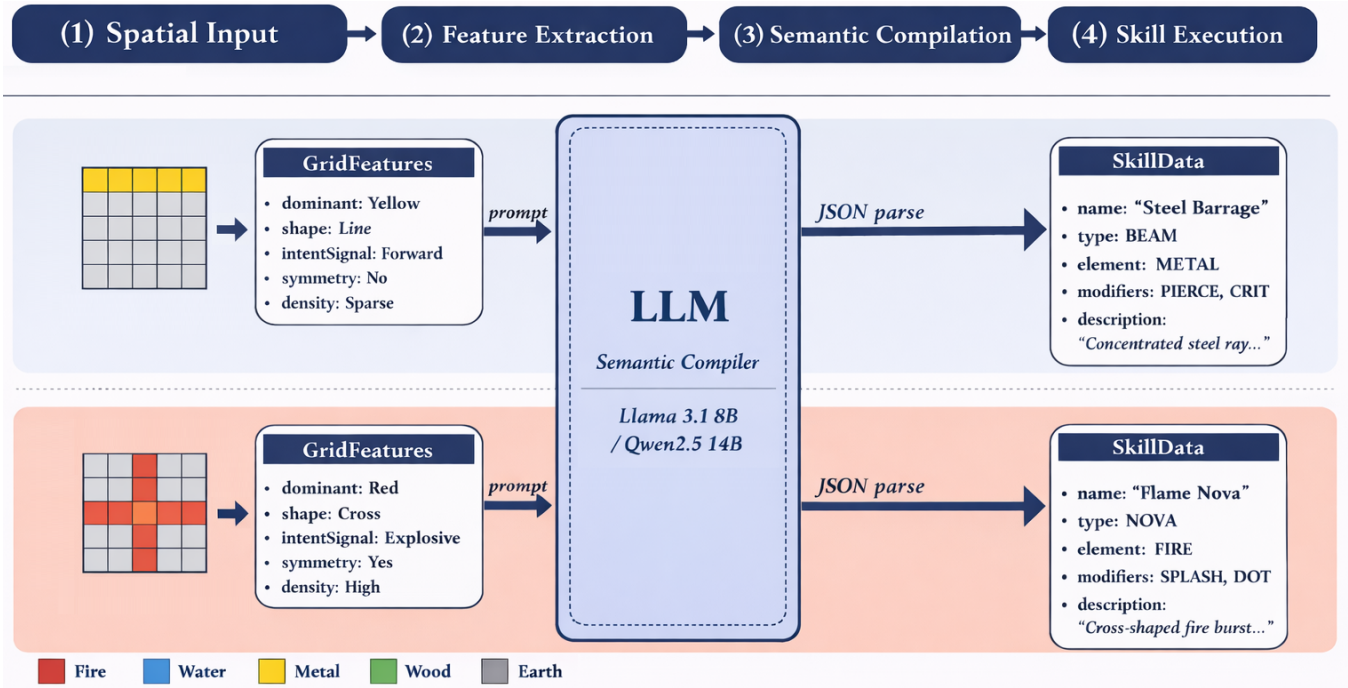


Figure 2: The overall pipeline. Players arrange elemental cells on a 5×5 grid; the game extracts GridFeatures including spatial map, symmetry score, and intent signal; the LLM compiles these features into a JSON-encoded SkillData object with 11 action types, 11 modifiers, and narrative description. Field values shown in the diagram are schematic illustrations of the pipeline structure; they do not represent exact runtime values or adhere to the enumerated field vocabulary defined in Section 3.

3 System Design

This section introduces Neural Spore Genesis through the gameplay design and creative inspirations that motivate the semantic-compiler approach, then formalizes the skill generation task, and concludes with the runtime skill schema.

3.1 Gameplay Design and Motivation

At a high level, *Neural Spore Genesis* is a browser-based roguelike auto-battler in which players buy elemental cells, arrange them across five independent 5×5 skill grids, and fight using the skills compiled from those layouts. Rather than selecting from a fixed authored skill catalog, players provide a spatial construction signal: elemental identity and grid structure together specify the observable pattern from which the compiler generates executable combat behavior. In this paper, we use "intent" operationally to refer to this grid-derived construction signal. In this sense, the game serves as a simplified 2D/2.5D real-time combat testbed for modular skill-engine systems in which player-authored configurations must be interpreted coherently under strong runtime randomness.

This design is motivated by several precedents in the genre. *Noita* shows that structured player construction can generate rich emergent skill diversity, but its spell assembly is manual and bounded by a fixed authored vocabulary. *Slay the Spire*[2] and *Teamfight Tactics*[3] show that combinatorial build expression creates deep player agency, but their synergies are selected from pre-authored assets rather than generated from spatial intent. We extend these ideas by treating cell arrangement itself as an explicit construction signal: instead of retrieving a predefined entry, the generator maps the grid's spatial and elemental features into a mechanically coherent skill.

A run alternates between three phases (Figure 1). In the *buy phase*, the market presents four randomly drawn elemental cells at variable costs; the player spends gold to purchase cells and may manually pay to re-roll the market to reveal additional options. In the *construction phase*, the player allocates purchased cells across five independent skill slots, each backed by its own 5×5 grid, enabling up to five distinct skills to be built and compiled simultaneously. A cyclic type-advantage system (Fire beats Wood, Wood beats Earth, Earth beats Water, Water beats Metal, Metal beats Fire) applies during combat, where each element deals 20% bonus damage against its disadvantaged counterpart. Once all arrangements are confirmed, the semantic compiler converts each populated grid independently into a SkillData JSON. In the *battle phase*, the engine executes the compiled skills against procedurally scaled enemy waves, and drop rewards feed the next buy phase.

The central generation challenge this design creates is a coherence-diversity tension. Semantic coherence requires the compiled skill to respect the elemental identity of the grid: a Fire-dominant cross should produce a fire-based area effect, not a water projectile. Creative diversity requires that repeated compilations of the same

[2]https://www.megacrit.com/
[3]https://teamfighttactics.leagueoflegends.com/

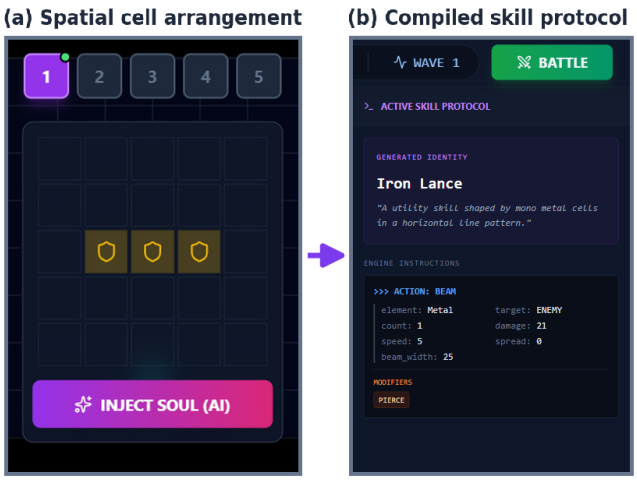


Figure 3: UI of the compilation process from spatial input to executable skill: (a) a fixed mono-Metal horizontal-line layout on the 5×5 grid, and (b) the generated protocol materialized in the skill panel as named text plus action/modifier fields.

grid across different runs produce meaningfully distinct skills, sustaining replayability. These objectives compete: a deterministic rule-based system achieves perfect coherence at the cost of zero run-to-run diversity; a uniform random sampler produces high variety but severs any connection between grid composition and skill semantics. The evaluation framework in Section 4.2 is designed to measure both dimensions simultaneously and to ask whether any generation strategy can break this tradeoff.

Listing 1: GridFeatures object

```
{
  "dominant": "Fire",
  "shape": "horizontal_line",
  "totalValue": 40,
  "symmetryScore": 0.72,      // bilateral symmetry
      [0,1]
  "densityRatio": 0.44,       // filled cells / 25
  "quadrantDistribution": [0.8, 0.8, 0.0, 0.0],
  "elementMixPattern": "binary", // mono/binary/ternary
      /diverse
  "elements": {"Fire": 32, "Water": 8},
  "grid": [...]                  // 5x5 element matrix
}
```

3.2 Problem Formulation

We define the skill generation task as follows. A grid configuration G is a 5×5 matrix where each cell contains an element $e \in \{\text{Fire, Water, Earth, Wood, Metal}\}$ or is empty. A feature extractor $\phi : G \to \mathbf{f}$ maps a grid to a GridFeatures vector $\mathbf{f}$ (Listing 1) capturing spatial and elemental properties. A skill generator $\mathcal{G} : \mathbf{f} \to \sigma$ produces a SkillData object σ conforming to the skill schema. The quality of $\mathcal{G}$ is characterized by a metric profile $\mathcal{M}(\mathcal{G}) = \{m_1, \ldots, m_{11}\}$ measuring validity, coherence, and diversity. The full $G \to \mathbf{f} \to \sigma$ pipeline is illustrated in Figure 2.

Table 1: Skill Schema: Action and Modifier types.

Action Types (11)	Modifier Types (11)
PROJECTILE, ZONE, HEAL	LIFESTEAL, STUN, SLOW
SHIELD, BUFF	KNOCKBACK, DOT
DASH, SUMMON, BEAM	PIERCE, CHAIN, SPLASH
NOVA, TRAP, ECHO	HOMING, CRIT, BLEED

The intentSignal field in GridFeatures is derived heuristically: high density with aggressive elements (Fire, Metal) yields "aggressive"; Earth or Wood dominant, or any Wood presence, yields "defensive"; sparse layouts (<25% density) yield "utility"; otherwise "balanced." Figure 3 illustrates the concrete transition from player input to runtime schema output. The left panel fixes the exact mono-Metal horizontal-line arrangement used as input. The right panel shows the corresponding SkillData instance after compilation, including the generated identity, action type, numeric fields, and modifier tags that are consumed by the runtime executor.

3.3 Skill Schema and Game System

The game is implemented as a browser-based React/TypeScript application in which players must defeat 20 escalating waves of enemies. Because the runtime executor reads the compiled SkillData object directly, the schema simultaneously defines the generation target and the execution contract. The skill schema enumerates 11 action types and 11 modifier types (Table 1), covering a bounded but representative subset of game mechanics derived from similar games. Actions span a range of combat patterns: BEAM delivers sustained line damage, NOVA produces a radial burst, DASH grants repositioning with temporary invincibility, SUMMON deploys orbiting drones, TRAP places a delayed-trigger zone, and ECHO replicates the primary action at 50% power. Modifiers augment any action: PIERCE passes projectiles through enemies, CHAIN bounces hits to nearby targets, SPLASH adds area effect on impact, HOMING steers toward the nearest enemy, CRIT applies a probability damage multiplier, and BLEED stacks a damage-over-time effect. Under the cyclic advantage system, each element deals 20% bonus damage against its disadvantaged counterpart.

Shape detection maps the spatial configuration to 7 categories organized in three groups: **Lines:** horizontal, vertical, diagonal; **Blocks:** dense_block, l_shape, t_shape; **Clusters:** corner_cluster. Each shape maps to a set of expected primary action types.

4 Experimental Setup

This section describes the generators, evaluation metrics, and corpus scale used in our experiments. Whereas the full Neural Spore Genesis game is designed for real-time player interaction (dynamic markets, live grid construction, battle feedback), the evaluation framework decouples the skill generation pipeline from live gameplay by operating on a predefined grid corpus. This controlled setup, an experimental variant and deliberate simplification of the full game mechanic, allows systematic measurement across the complete shape-element design space without confounding player behavior, session length, or purchase decisions.

4.1 Baselines

We evaluate 6 generators spanning the range from fully deterministic to LLM based:

Rule-Based (RB): Deterministic baseline mapping the full 11-action and 11-modifier schema to spatial features, using intentSignal, symmetryScore, and the 7-shape action map.

Random Valid (RV): Schema-valid random generator selecting action types, elements, and modifiers uniformly from valid sets, with numeric fields sampled uniformly within schema bounds. This provides a statistical performance floor.

Template-Based (TB): A library of 20 hand-authored skill templates covering common shape-element combinations, instantiated with per-configuration power scaling. This represents expert design knowledge as a reference ceiling for description quality.

Calibrated Sampler (CS): A lightweight probabilistic generator that first determines the grid's dominant element, then samples an action type, modifier set, and numeric parameters from fixed hand-specified distributions. These distributions are derived from the same design priors used by the skill schema: element affinities define likely modifiers, shape categories define likely primary actions, and numeric fields are sampled within the schema ranges used by the runtime executor. The calibration was completed before evaluation and was limited to making the sampler produce executable and varied outputs; it was not optimized against the final metric scores or tuned separately for any evaluated generator. CS therefore serves as an inference-free stochastic baseline that uses the same vocabulary and execution constraints as the other methods, but lacks language-model semantic compilation.

Llama 3.1 8B (Llama): Open-weight model via Ollama on an RTX 4080 16 GB at approximately 15 tok/s. Temperature 0.8, context 4096 tokens.

Qwen2.5 14B (Qwen): Open-weight model via Ollama at approximately 8 tok/s. Temperature 0.8, context 4096 tokens.

4.2 Metrics

The evaluation covers three dimensions: structural validity, semantic coherence, and creative diversity. Semantic metrics use the all-MiniLM-L6-v2 sentence-transformer model [17, 24]; ICD is aggregated per configuration across N repeated samples.

Structural validity metrics assess whether the generator produces a well-formed, in-range skill object. Intuitively, a well-formed skill is one that has the required structure and can be read by the game executor, while an in-range skill uses numeric values that stay within playable schema-defined bounds. **Schema Validity (SV)** is the fraction of required schema checks passed: non-null output, required fields present, valid action and element type values. **Numeric Range Compliance (NRC)** is the fraction of numeric action fields within schema-specified bounds, penalizing out-of-range values proportionally.

Semantic coherence metrics measure whether the compiled skill reflects the spatial and elemental intent of the grid. **Element Coherence (EC)** checks whether the primary action's element matches the dominant grid element (1.0), appears only in a secondary action (0.5), or is absent (0.0). **Shape-Action Coherence (SAC)** checks whether the primary action type belongs to the expected set $\mathcal{A}(s)$ for shape s: $SAC(\sigma, s) = \mathbb{1}[\sigma.\text{actions}[0].\text{type} \in \mathcal{A}(s)]$. **Semantic Coherence (SC)** is the cosine similarity between the skill's name and description and the element's reference theme corpus via sentence embeddings. **Description Alignment (DA)** measures the cosine similarity between the skill's natural-language description field and a compact mechanics text composed of the skill's action types, element, and modifier types, capturing whether the narrative description reflects the actual generated mechanics.

Creative diversity metrics characterize the range of outputs a generator produces. **Intra-Config Diversity (ICD)** is the mean pairwise cosine distance between skill embeddings across N samples of the same configuration: $ICD = \frac{2}{N(N-1)} \sum_{i<j} (1 - \cos(\mathbf{e}_i, \mathbf{e}_j))$. **Action Diversity (AD)** is the Shannon entropy of the action type distribution normalized to $[0, 1]$ over 11 types. **Modifier Utilization (MU)** is the fraction of the 11 modifier types used at least once across all samples. **Skill Complexity Score (SCS)** is a normalized composite of action count, modifier count, and special flags. **Cross-Config Consistency (CCC)** measures how uniformly valid a generator is across shape categories: $CCC = 1 - \overline{\sigma_s(\text{SV})}$, where validity scores are grouped by shape type and σ_s is the within-shape standard deviation; a generator that reliably succeeds or fails for each shape scores high, while one whose validity varies within shape groups scores low.

4.3 Experiment Parameters

The grid corpus enumerates 7 spatial shapes × 5 elemental types × 3 composition variants (mono, binary, ternary), yielding **105 configurations** in total. All main-result generators are evaluated over the full corpus with identical sampling settings to maintain fully matched comparison conditions across methods. Each generator uses 5 samples per configuration, yielding 105 configurations × 5 samples = 525 skills per condition. The temperature ablation study uses Llama 3.1 8B at $T \in \{0.3, 0.6, 0.9, 1.2\}$ over 50 configurations × 2 samples per temperature.

5 Results

We present quantitative results across all generators and metrics, followed by qualitative case analysis.

5.1 Quantitative Results

Structural Validity: All schema-constrained baselines except CS achieve SV = 1.000 by construction. Qwen2.5 14B also achieves perfect SV = 1.000; Llama 3.1 8B reaches SV = 0.984. Both substantially exceed CS (SV = 0.918), confirming that instruction-tuned open-weight models have stronger JSON schema adherence than a calibrated stochastic sampler. NRC follows the same ordering: Llama 0.932, Qwen 0.954, CS 0.918. The SV gap reflects a qualitative architectural difference: instruction-tuned models reliably produce complete structured outputs under the prompt, whereas CS samples fields independently from calibrated distributions and does not enforce all cross-field compatibility checks used by the runtime validator.

Element Coherence: Rule-based generators achieve perfect EC = 1.000 through hard-coded element binding. Both Llama and Qwen achieve EC = 0.994, which is near-perfect and substantially exceeds

Table 2: Generator comparison across all 11 metrics (mean values). Left block: structural validity and semantic coherence; right block: creative diversity and coverage. Kruskal-Wallis tests confirm significant differences for EC, SAC, and SC ($p < 10^{-5}$).

Generator	N	Validity & Coherence						Diversity & Coverage				
		SV	NRC	EC	SAC	SC	DA	ICD	AD	MU	SCS	CCC
Rule-Based	525	1.000	1.000	1.000	1.000	0.349	0.269	0.000	0.816	0.909	0.267	1.000
Random Valid	525	1.000	1.000	0.301	0.228	0.155	0.103	0.018	0.998	1.000	0.314	1.000
Template-Based	525	1.000	1.000	0.371	0.229	0.290	0.503	0.000	0.616	0.818	0.294	1.000
Calib. Sampler	525	0.918	0.918	0.824	0.665	0.293	0.301	0.371	0.869	0.455	0.160	0.728
Llama 3.1 8B	525	0.984	0.932	0.994	0.422	0.353	0.422	0.429	0.799	1.000	0.401	0.875
Qwen2.5 14B	525	1.000	0.954	0.994	0.554	0.336	0.430	0.433	0.778	1.000	0.352	1.000

SV=Schema Validity; NRC=Numeric Range Compliance; EC=Element Coherence; SAC=Shape-Action Coherence; SC=Semantic Coherence; DA=Description Alignment; ICD=Intra-Config Diversity; AD=Action Diversity; MU=Modifier Utilization; SCS=Skill Complexity; CCC=Cross-Config Consistency.

CS (EC = 0.824). CS's lower score reflects its shallow element-to-action lookup: the mapping can be overridden when CS independently samples a secondary-action element that conflicts with the dominant grid element. In contrast, instruction-tuned LLMs have internalized elemental semantics from large pretraining corpora that include fantasy game text and elemental lore, allowing them to reliably bind output vocabulary to the dominant element. Random Valid (0.301) and Template-Based (0.371) confirm that generators lacking grid-conditioned element selection cannot achieve meaningful EC regardless of other properties.

Shape-Action Coherence: Rule-Based achieves perfect SAC = 1.000 by construction, since its action selection is directly keyed to the 7-shape action map. CS reaches SAC = 0.665. Both local LLMs score below CS: Llama 0.422, Qwen 0.554. This pattern is counterintuitive but explicable: CS samples from element-conditioned action distributions whose mode action types (e.g., BEAM for Metal, NOVA for Fire) empirically overlap with the shape-expected action sets in the corpus, without enforcing any shape-action constraint explicitly, while LLMs occasionally produce geometrically unconventional but narratively plausible alternatives. For example, a diagonal line suggests swift traversal, producing DASH+BEAM rather than the mapped BEAM alone. This behavior represents richer interpretive flexibility rather than error, at the cost of rule compliance. Qwen's higher SAC (0.554 vs. 0.422 for Llama) suggests that larger parameter counts improve internalization of geometric-to-action correspondences.

Semantic Coherence and Description Alignment: Llama (SC = 0.353) and Rule-Based (SC = 0.349) achieve the highest SC scores and are statistically indistinguishable (Bonferroni-corrected $p > 0.05$); Rule-Based attains this through direct element-vocabulary mappings while Llama reaches equivalent thematic alignment via pretraining on fantasy-genre text. For DA, Template-Based achieves the highest score (0.503) due to richly hand-crafted elemental metaphors. Both local LLMs substantially exceed CS on DA: Qwen 0.430, Llama 0.422, versus CS 0.301. The absolute gap between Template-Based DA (0.503) and local LLMs (Qwen 0.430, Llama 0.422) is approximately 7–8 percentage points, indicating that real open-weight LLMs generate descriptions with substantially greater thematic precision than CS (0.301) while still falling short of hand-authored template precision.

Diversity and Coverage: Random Valid achieves AD = 0.998 and MU = 1.000 by uniform sampling. Both local LLMs achieve full modifier utilization (MU = 1.000, using all 11 modifier types across 525 skills) and high action diversity (Llama AD = 0.799, Qwen AD = 0.778). The most striking contrast with CS is on modifier utilization: CS MU = 0.455 reaches less than half the modifier vocabulary, while both LLMs reach MU = 1.000. CS uses the same executable modifier vocabulary as the other generators, but its hand-specified modifier prior concentrates probability mass on a narrower set of basic modifiers. In contrast, instruction-tuned LLMs treat all 11 modifiers as semantically available options and distribute usage accordingly. CS achieves AD = 0.869 by distributing action selection across the full set of shape-expected types, yet this breadth does not extend to modifiers. Both local LLMs also generate more complex skill objects (Llama SCS = 0.401, Qwen SCS = 0.352 vs. CS SCS = 0.160), producing multi-action, multi-modifier designs where CS typically emits single-action output. Llama and Qwen cluster in ICD = 0.429–0.433, consistently above CS ICD = 0.371, indicating that instruction-tuned LLMs produce more diverse outputs across identical configurations.

The Coherence-Diversity Space: A central question is whether generator quality follows a coherence-diversity tradeoff. Table 2 shows that Rule-Based occupies the high-coherence, zero-diversity quadrant (EC = 1.000, ICD = 0.000), while random and template generators occupy low-coherence positions regardless of diversity (EC $\leq$ 0.371). The two local LLMs simultaneously achieve near-perfect EC and high ICD, occupying a high-coherence, high-diversity region that no baseline generator reaches.

This result challenges the assumption that coherence and diversity must be traded off against each other: the rule-based generator and the random generator each maximize one objective at the cost of the other, while instruction-tuned LLMs break this tradeoff by leveraging deep contextual understanding of both elemental semantics and generative variety. The practical implication for high-randomness games is that a deployed semantic compiler can provide elementally consistent and run-to-run distinct executable skills. Whether this translates into perceived player agency or satisfaction requires human evaluation.

Shape-by-Generator Analysis: Figure 4 shows SAC broken down by shape and generator for the seven evaluated shapes. Horizontal and diagonal lines achieve high SAC for Rule-Based and Qwen, where the BEAM and PROJECTILE mapping is semantically unambiguous. Vertical lines score substantially lower for local LLMs (Llama 0.07, Qwen 0.09), suggesting the models' spatial prior

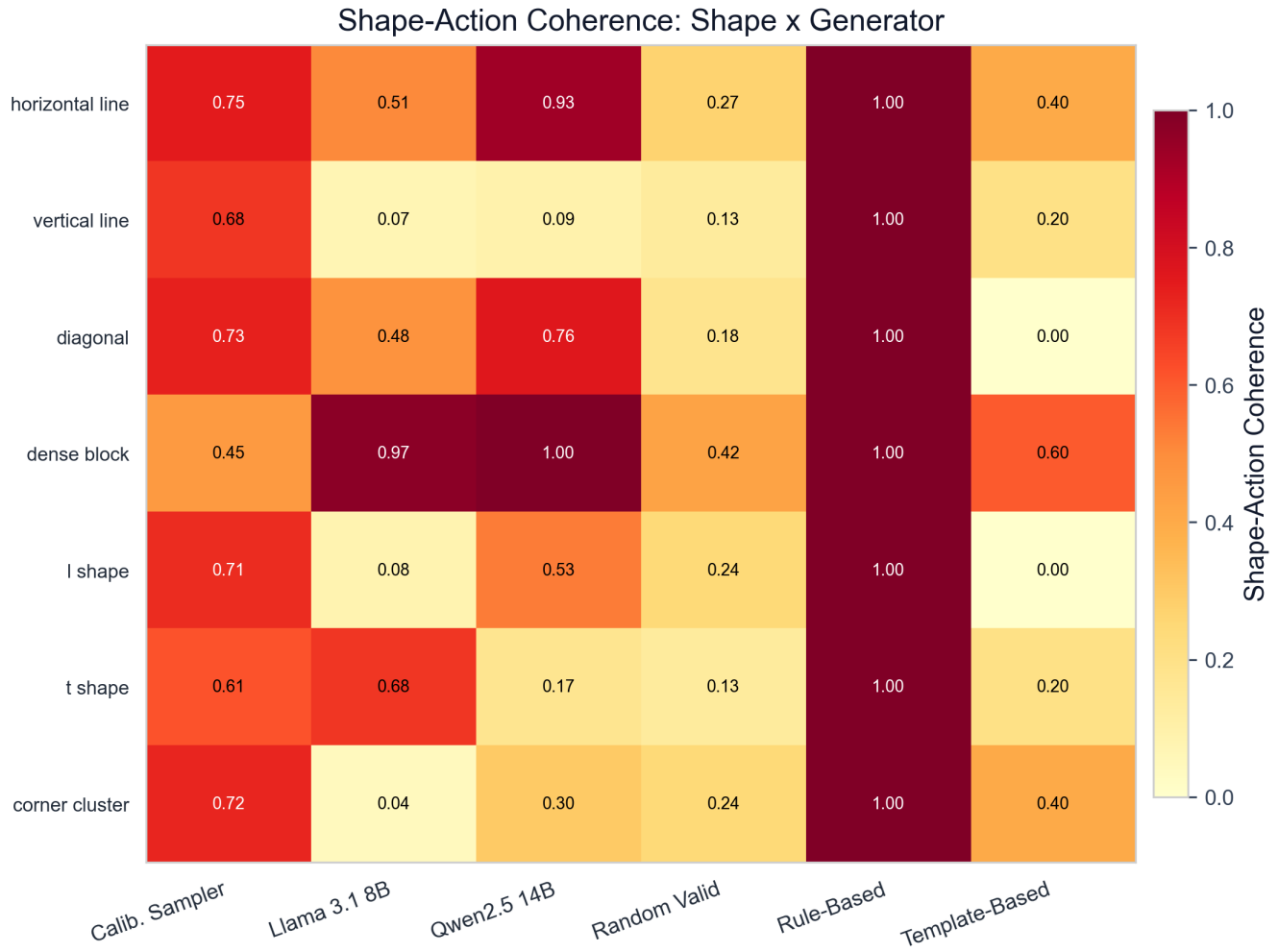


Figure 4: Shape-Action Coherence by shape type and generator across the seven evaluated shapes. Horizontal and diagonal lines achieve high SAC for Rule-Based and Qwen; vertical lines score substantially lower for both LLMs (Llama 0.07, Qwen 0.09), suggesting the models' spatial prior does not reliably distinguish line orientation. Dense-block configurations achieve near-perfect SAC for both Llama (0.97) and Qwen (1.00). Calib. Sampler achieves moderate SAC across all seven shapes (0.45–0.75), scoring highest on horizontal and diagonal lines and lowest on dense-block, reflecting that its element-conditioned action distributions partially align with shape-expected action types.

does not reliably distinguish line orientation. Dense-block configurations achieve near-perfect SAC for both Llama (0.97) and Qwen (1.00), indicating strong recognition of filled-area patterns.

Temperature Ablation: Table 3 and Figure 5 show metric trajectories across $T \in \{0.3, 0.6, 0.9, 1.2\}$. The main experiment uses $T = 0.8$, which falls between the tested brackets $T = 0.6$ and $T = 0.9$; because the ablation uses a smaller corpus, its absolute values reflect trends rather than main-result magnitudes. ICD increases substantially from 0.297 at $T = 0.3$ to 0.472 at $T = 1.2$, confirming that temperature is the primary control for creative diversity in this task. EC remains at 1.000 for $T \leq 0.9$ and decreases only slightly to 0.995 at $T = 1.2$, indicating that elemental grounding is robust to temperature variation. SAC shows a mild positive trend (0.380 to 0.460), contrary to the hypothesis that higher temperatures degrade shape-action conformance; at higher temperatures the model explores a wider range of action types, some of which happen to be

Table 3: Temperature ablation results for Llama 3.1 8B (50 configs × 2 samples per temperature).

T	SV	EC	SAC	SC	ICD	DA
0.3	1.000	1.000	0.380	0.364	0.297	0.439
0.6	0.980	1.000	0.380	0.367	0.427	0.417
0.9	0.990	1.000	0.400	0.363	0.467	0.434
1.2	0.980	0.995	0.460	0.367	0.472	0.422

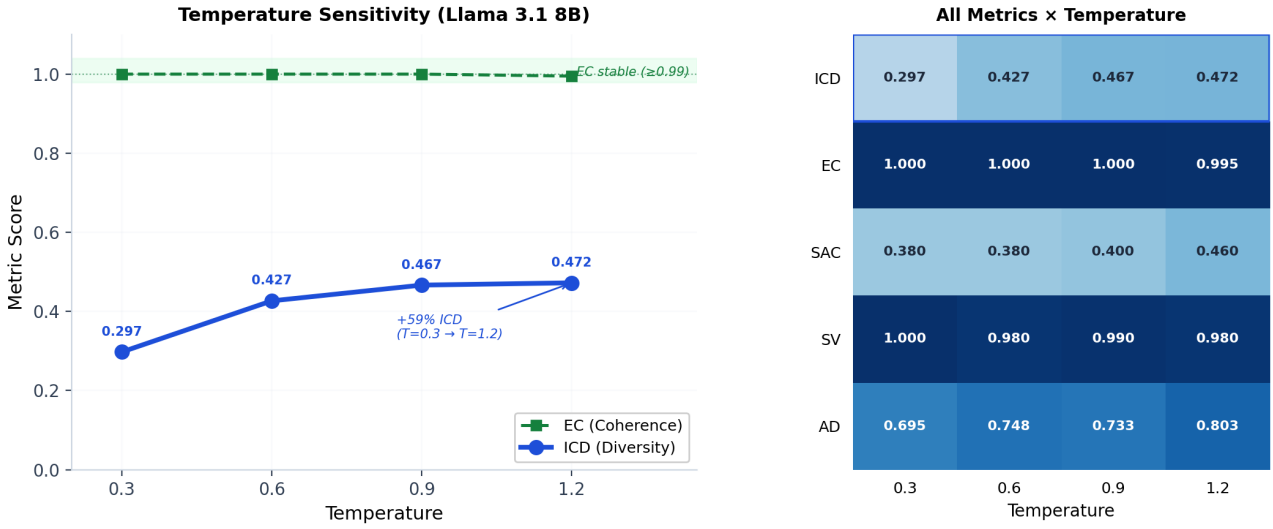


Figure 5: Effect of temperature on Llama 3.1 8B quality metrics. ICD increases monotonically with temperature; EC remains near-perfect across all conditions; SAC shows a mild positive trend.

shape-appropriate alternatives. SV dips modestly to 0.980 at $T = 0.6$ and $T = 1.2$ (recovering slightly to 0.990 at $T = 0.9$), remaining well above the CS baseline (0.918) and suggesting that structural validity is largely robust to temperature variation. For practitioners who prioritize diversity, $T = 1.2$ achieves the highest ICD with only a 0.5% EC reduction. For practitioners who prioritize exact elemental grounding, $T = 0.3$ provides perfect EC at lower diversity cost.

Model Size Scaling: The main results (Table 2) directly compare Llama 3.1 8B and Qwen2.5 14B under matched 105-configuration corpus coverage and identical 5-sample-per-configuration evaluation. Qwen achieves higher SV (1.000 vs. 0.984), higher SAC (0.554 vs. 0.422), higher NRC (0.954 vs. 0.932), and higher CCC (1.000 vs. 0.875). Both models reach near-identical EC (0.994), ICD (approximately 0.43), and MU (1.000), suggesting that element grounding and modifier coverage are stable across the 8B-to-14B range tested; whether larger scales yield further gains remains an open question. The throughput difference is approximately 1.9× (15 tok/s vs. 8 tok/s), placing the SAC improvement from Llama to Qwen at a throughput cost of 90% per skill generation call.

5.2 Qualitative Results

Figure 6 shows the coherence-diversity scatter (left) alongside supplemental bar charts for MU, SAC, DA, and SCS (right). The most visible difference between CS and real LLMs is on MU (0.455 vs. 1.000): CS systematically underuses modifiers, producing skills with only STUN, SLOW, KNOCKBACK, LIFESTEAL, or DOT while real LLMs distribute across all 11 modifier types. CS also substantially underperforms on skill complexity (SCS: 0.160 vs. 0.35–0.40) and intra-configuration diversity (ICD: 0.371 vs. 0.43); real LLM descriptions additionally show a clear gap on DA (0.301 vs. 0.42–0.43), incorporating multi-element metaphors and thematic vocabulary far more precisely than CS's distribution-sampled output.

To control for input variance, Figure 7 fixes the same mono-Metal horizontal-line configuration and compares four generation methods. Rule-Based produces a compact deterministic protocol ("Iron Lance") centered on a BEAM with PIERCE. Template generation ("Steel Rail Shot") favors a single PROJECTILE profile with hand-authored naming and concise action fields. Llama ("Metal Blast") emits a denser BEAM schema with additional optional fields and CRIT emphasis. Qwen ("Metallic Barrage") shifts to a high-count PROJECTILE profile with CRIT+PIERCE, retaining the same elemental grounding while changing attack expression.

The battle frames in Figure 7 show corresponding runtime differences from the same cell arrangement: deterministic line damage in Rule-Based, sparse high-impact shots in Template, broader effect-heavy behavior in Llama, and fast projectile-oriented pressure in Qwen. This controlled visual comparison matches the quantitative result that local LLMs preserve coherence while exploring a wider executable design space than deterministic baselines.

6 Conclusion

We presented Neural Spore Genesis, a semantic compiler framework that maps spatial-elemental grid arrangements into executable SkillData objects, evaluated across 6 generators and 11 metrics on a 105-configuration corpus. Within this bounded schema, the results show that local open-weight LLMs can convert grid-derived construction signals into executable skills while simultaneously achieving near-perfect element coherence, full modifier utilization, high skill complexity, and high intra-config diversity. The temperature ablation further shows that diversity is the primary dimension controlled by temperature, while elemental grounding and structural validity remain robust across the range.

This work has four limitations. First, all coherence metrics are automated: embedding similarity and rule conformance scores capture

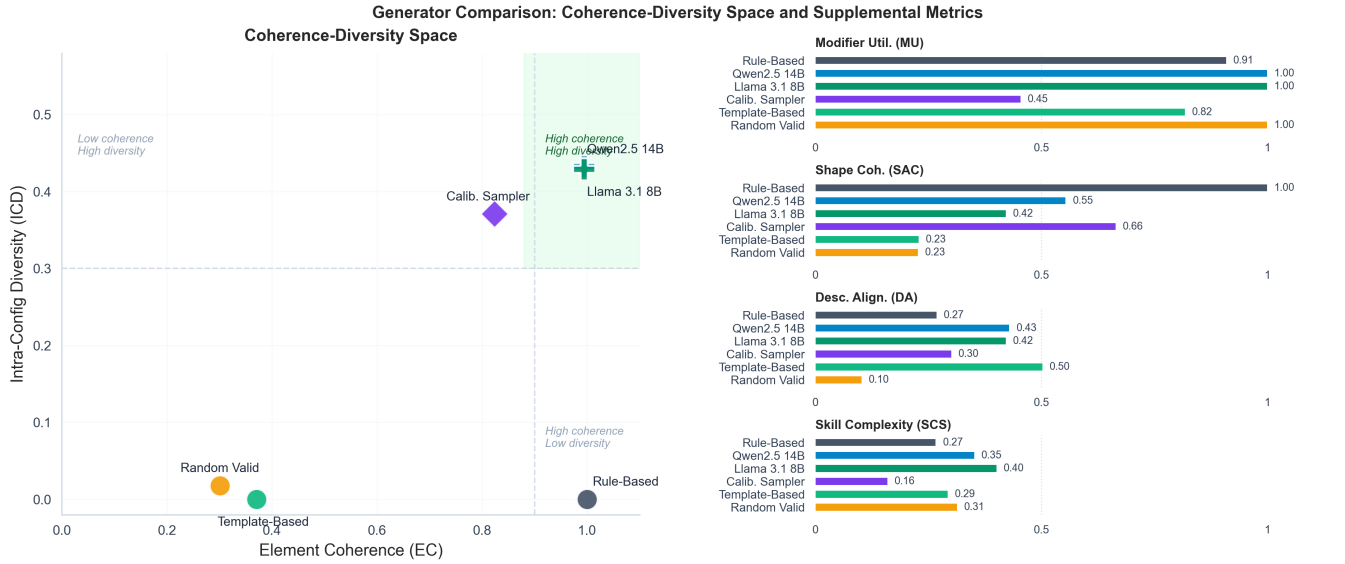


Figure 6: Left: generator positions in the coherence-diversity space (EC × ICD). Only instruction-tuned open-weight models occupy the high-coherence, high-diversity quadrant (top right), while deterministic generators achieve high EC at zero diversity and random/template generators sacrifice both. Right: four supplemental bar charts showing that both LLMs reach full modifier utilization and outperform CS on description alignment and skill complexity, while Rule-Based leads on SAC.

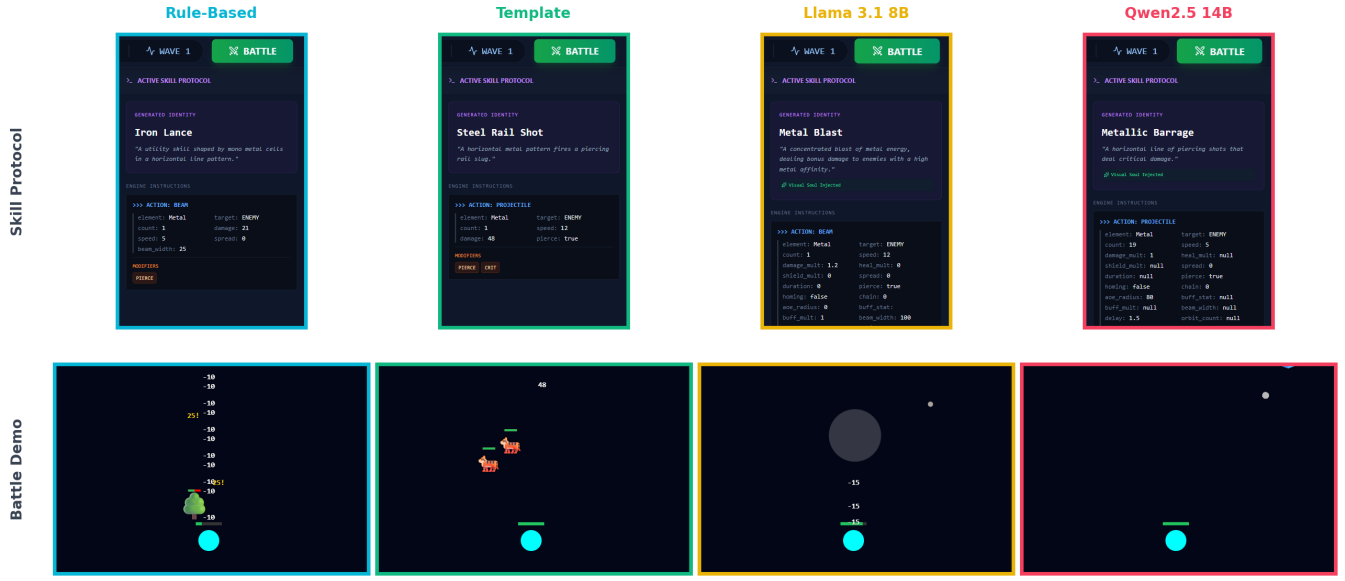


Figure 7: Controlled qualitative case study with identical input across four generators. Input is fixed to a mono-Metal horizontal line. Each method shows the generated skill description panel (top) and a short in-battle runtime frame (bottom).

aspects of semantic alignment but do not measure player perception of coherence, surprise, agency, or satisfaction. Second, the evaluation corpus intentionally uses a closed ontology of seven spatial shapes, five elements, and three composition variants; therefore, the results support claims within this bounded schema but do not establish generalization to more ambiguous or less discretized player constructions. Third, all LLM conditions evaluate open-weight models at 8B–14B parameter scale; frontier API models and larger open-weight variants may exhibit different quality profiles, and it remains unknown whether increased scale can close the observed SAC deficit relative to rule-based generators or further narrow the description-alignment gap with template-authored skills. Fourth,

all LLM conditions use a single prompt template without prompt-style ablations or few-shot examples, leaving prompt sensitivity and the remaining gap between local LLM and template description-alignment scores unaddressed.

Future work will address each limitation in sequence. A player study measuring perceived coherence, design satisfaction, and replayability will provide human-grounded evaluation to complement the automated metrics. Extending the evaluation to frontier API models and larger open-weight parameter counts will test whether model scale closes the shape-action coherence deficit and whether the coherence-diversity quality profile observed at 8B–14B generalizes to more capable models. Prompt engineering with few-shot template examples will target the remaining description-alignment gap and may bring local LLM output quality to or beyond template-level precision. Taken together, these directions extend the framework from a controlled generator comparison toward a deployed, player-facing design tool. For example, auto-battler games could expose race, class, synergy, and randomized build-state variables as structured inputs, allowing an LLM compiler to generate semantically coherent units or skills while leaving balance and validation to game-specific systems. More broadly, we argue that modular skill structures deserve greater attention as a foundation for scalable and meaningful skill PCG, with the potential to make high-variety combat content more practical for independent and small-studio game development. This evaluation infrastructure should support further research at the intersection of spatial semantic intent, procedural content generation, and LLM-based game design.

References

[1] Bahar Bateni and Jim Whitehead. 2024. Language-driven play: Large language models as game-playing agents in slay the spire. In *Proceedings of the 19th International Conference on the Foundations of Digital Games*. 1–10.

[2] Nancy N Blackburn, M Gardone, and Daniel S Brown. 2023. Player-centric procedural content generation: Enhancing runtime customization by integrating real-time player feedback. In *Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 10–16.

[3] Megan Charity, Ahmed Khalifa, and Julian Togelius. 2020. Baba is Y'all: Collaborative Mixed-Initiative Level Design. In *2020 IEEE Conference on Games (CoG)*. 542–549. doi:10.1109/CoG47356.2020.9231807

[4] Omar Delarosa, Hang Dong, Mindy Ruan, Ahmed Khalifa, and Julian Togelius. 2021. Mixed-initiative level design with rl brush. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer, 412–426.

[5] 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*. 1–15.

[6] Roberto Gallotta, Graham Todd, Marvin Zammit, Sam Earle, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2024. Large language models and games: A survey and roadmap. *IEEE Transactions on Games* (2024).

[7] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 1 (2013), 1–22.

[8] Chengpeng Hu, Yunlong Zhao, and Jialin Liu. 2024. Game generation via large language models. In *2024 IEEE Conference on Games (CoG)*. IEEE, 1–4.

[9] 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*, Vol. 16. 95–101.

[10] Gorm Lai, Frederic Fol Leymarie, and William Latham. 2022. On mixed-initiative content creation for video games. *IEEE Transactions on Games* 14, 4 (2022), 543–557.

[11] Antonios Liapis. 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.

[12] Antonios Liapis, Gillian Smith, and Noor Shaker. 2016. Mixed-initiative content creation. In *Procedural content generation in games*. Springer, 195–214.

[13] Mahdi Farrokhi Maleki and Richard Zhao. 2024. Procedural content generation in games: A survey with insights on emerging llm integration. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 20. 167–178.

[14] Arash Moradi Karkaj, Mark J Nelson, Ioannis Koutis, and Amy K Hoover. 2024. Prompt wrangling: On replication and generalization in large language models for pcg levels. In *Proceedings of the 19th International Conference on the Foundations of Digital Games*. 1–8.

[15] Muhammad U Nasir and Julian Togelius. 2023. Practical PCG Through Large Language Models. In *2023 IEEE Conference on Games (CoG)*. 1–4. doi:10.1109/CoG57401.2023.10333197

[16] Xiangyu Peng, Jessica Quaye, Sudha Rao, Weijia Xu, Portia Botchway, Chris Brockett, Nebojsa Jojic, Gabriel DesGarennes, Ken Lobb, Michael Xu, et al. 2024. Player-driven emergence in llm-driven game narrative. In *2024 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[17] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410

[18] Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games*. Springer.

[19] Gillian Smith, Elaine Gan, Alexei Othenin-Girard, and Jim Whitehead. 2011. PCG-based game design: enabling new play experiences through procedural content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. 1–4.

[20] 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*. 1–7.

[21] 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*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 54213–54227.

[22] 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*. 1–8.

[23] Susanna Värtinen, Perttu Hämäläinen, and Christian Guckelsberger. 2024. Generating Role-Playing Game Quests With GPT Language Models. *IEEE Transactions on Games* 16, 1 (2024), 127–139. doi:10.1109/TG.2022.3228480

[24] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 5776–5788. https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[25] Oliver Withington. 2025. Exploring the possibility space of 1 billion spells. In *Proceedings of the 20th International Conference on the Foundations of Digital Games*. 1–4.

[26] Kaijie Xu and Clark Verbrugge. 2025. Constraint Is All You Need: Optimization-Based 3D Level Generation with LLMs. In *Proceedings of the 20th International Conference on the Foundations of Digital Games (FDG '25)*. Association for Computing Machinery, New York, NY, USA, Article 66, 13 pages. doi:10.1145/3723498.3723840

[27] Kaijie Xu and Clark Verbrugge. 2025. A Database-Driven Framework for 3D Level Generation with LLMs. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 21, 1 (Nov. 2025), 367–376. doi:10.1609/aiide.v21i1.36840

[28] Daijin Yang, Erica Kleinman, and Casper Harteveld. 2025. GPT for Games: An Updated Scoping Review (2020-2024). *IEEE Transactions on Games* (2025), 1–16. doi:10.1109/TG.2025.3563780

[29] Georgios N Yannakakis and Julian Togelius. 2018. *Artificial intelligence and games*. Vol. 2. Springer.

[30] Andrew Zhu, Lara Martin, Andrew Head, and Chris Callison-Burch. 2023. CALYPSO: LLMs as Dungeon Master's Assistants. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 19. 380–390.