INFINIDREAM: OPEN-ENDED PHOTOREALISTIC 3D WORLDS THROUGH PROCEDURAL GENERATION

Anonymous authors
Paper under double-blind review

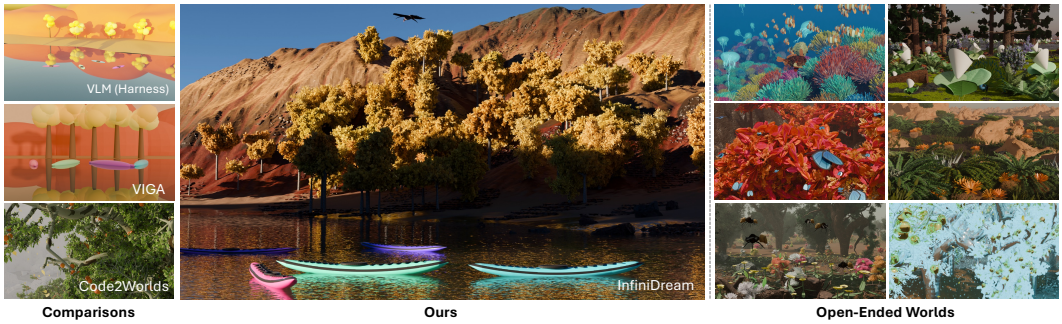

Figure 1: **Beyond Predefined Factories: Rich, Editable Procedural Worlds.** InfiniDream extends existing procedural generators to create novel assets and visually rich 3D scenes. We compare its results with existing VLM-based 3D scene generation methods.

ABSTRACT

Procedural generation plays a key role in synthetic data creation and game development, offering greater controllability, editability, and diversity than static mesh representations. However, VLM-based code generation often produces relatively simple 3D assets, while sophisticated photorealistic procedural systems require extensive manual development. Existing VLM-guided procedural methods remain largely constrained by predefined factories and APIs, limiting their ability to create novel asset types. We introduce **InfiniDream**, a VLM-driven framework for open-ended procedural 3D world generation. First, we propose *Symbolically Constrained Asset Generation*, which leverages existing procedural algorithms to create novel, photorealistic assets while preserving editability and seed-based diversity. Second, we develop an end-to-end text-to-world pipeline that combines semantic scene planning with *Deterministic Frustum-Guided Grid Placement* to construct large-scale 3D worlds with precise object placement, procedural population, and animation. Extensive evaluations on 100 object and 20 scene generation tasks demonstrate improvements across four perceptual metrics. Diversity analysis, ablation studies, and human preference evaluations further validate the effectiveness of our framework.

1 INTRODUCTION

Procedural generation creates 3D assets and scenes through executable programs, offering editability, diversity, and scalability. These advantages have made it widely applicable to synthetic data generation, game development, and virtual world creation (Raistrick et al., 2023). However, conventional procedural generation methods rely heavily on handcrafted rules and predefined asset generators, limiting the variety of generated content and their ability to create novel assets and scenes.

Recent advances in the code generation and tool-use capabilities of large language models (LLMs) and vision-language models (VLMs) have enabled the generation and editing of 3D assets and scenes through executable Blender code (Hu et al., 2024; Lu et al., 2025; Rondelli et al., 2026; Yin et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

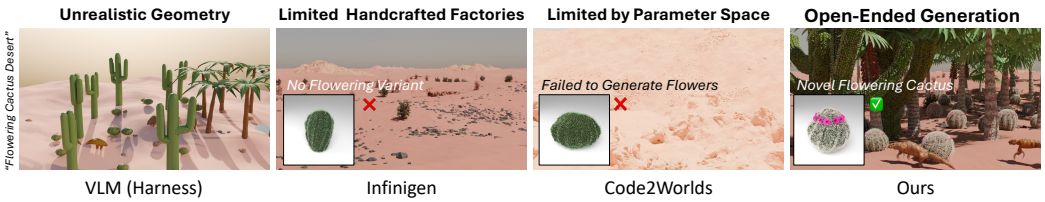

Figure 2: **Motivation: Beyond predefined procedural factories.** Given a flowering cactus desert prompt, VLM-based code generation produces simplified geometry, predefined factories offer limited variations, and parameter-based generation struggles to introduce new features. InfiniDream extends existing procedural generators to create novel assets within rich, photorealistic 3D worlds.

2026; He et al., 2026; Wang et al., 2026c). However, while existing approaches can construct scenes by composing assets or directly synthesizing geometry through code, generating large-scale 3D scenes with consistently rich geometric details, complex materials, and photorealistic appearance remains challenging.

Infinigen (Raistrick et al., 2023) is built on over 40,000 lines of code, including carefully handcrafted procedural asset factories, to generate photorealistic, editable 3D scenes. However, its generative capabilities remain constrained by predefined asset categories and procedural rules.

Recent studies have explored using LLMs and VLMs to control sophisticated procedural generation systems through function orchestration, parameter adjustment, and the composition of existing generators (Sun et al., 2023; Zhou et al., 2025; Zhang et al., 2026). While these approaches improve the controllability of procedural scene generation, methods that primarily operate within predefined functions and parameter spaces remain limited in their ability to create novel asset types beyond the capabilities of existing generators.

Another line of research explores large-scale 3D world generation using retrieved or generatively synthesized mesh assets (Guo et al., 2026; Wang et al., 2026a). Although these methods support instance-level editing, their mesh-based assets lack the parameterized editability and seed-based diversity of procedural generators. Producing diverse object variations often requires direct mesh manipulation or regeneration.

Figure 2 highlights three limitations: VLM Harness produces less photorealistic scenes, Infinigen is restricted to predefined factories, and Code2Worlds (Zhang et al., 2026), a recent VLM-guided Infinigen framework, is limited to function calls and parameter adjustment. These limitations motivate open-ended procedural generation beyond existing factory capabilities.

To address these challenges, we introduce **InfiniDream**, a VLM-driven framework for open-ended procedural 3D world generation, designed to achieve realism, open-endedness, editability, diversity, and scalability. By employing VLM reasoning to compose and extend sophisticated procedural programs, InfiniDream generates novel assets beyond predefined factories while leveraging existing procedural algorithms. Our framework enables the creation of photorealistic, editable, and diverse large-scale 3D worlds from natural language descriptions.

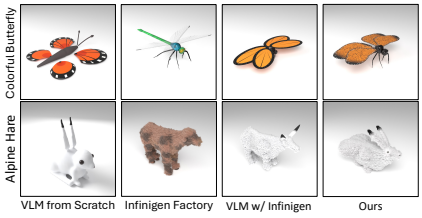

Figure 3: **Comparison of procedural asset generation strategies.** From left to right: VLM-generated Blender code from scratch, predefined Infinigen factories, VLM Agent Harness with full Infinigen access, and our symbolically constrained editing.

At the asset level, Fig. 3 further motivates our Symbolically Constrained Editing (SCE) strategy, which combines the flexibility of VLM-based code generation with the rich procedural structures of existing factories.

Our main contributions are summarized as follows:

1. **Open-Ended Procedural Asset Generation.** We propose Symbolically Constrained Editing (SCE) to generate novel, photorealistic procedural assets with rich geometric details beyond predefined factories, while preserving editability and seed-based diversity.

2. **Scalable Procedural World Generation.** We introduce an end-to-end text-to-world framework combining semantic scene planning and deterministic frustum-guided grid placement to generate large-scale, editable 3D worlds.
3. **Comprehensive Experimental Validation.** We conduct extensive object- and scene-level evaluations, diversity analysis, ablation studies, and human preference studies to validate the effectiveness of InfiniDream.

2 Related Work

Procedural 3D Asset Generation. Procedural modeling encodes 3D content as executable rules, from L-systems (Lindenmayer, 1968; Prusinkiewicz & Lindenmayer, 1990) and shape grammars (Wonka et al., 2003; Müller et al., 2006) to learned procedural shape and CAD programs (Sharma et al., 2018; Tian et al., 2019; Jones et al., 2020; 2021; 2023; Ritchie et al., 2023; Wu et al., 2021; Khan et al., 2024). Advances in code generation (Chen et al., 2021) and tool use (Schick et al., 2023; Yao et al., 2023) now let language models synthesize editable 3D assets as Blender code (Du et al., 2024; Lu et al., 2025; Rondelli et al., 2026; Wang et al., 2026c; Yamada et al., 2025; Dai et al., 2025; Raistrick et al., 2026), other shape programs (Zhang et al., 2025a; Jones et al., 2026; Wang et al., 2026b; Lin et al., 2026), or CAD scripts for mechanical parts (Alrashedy et al., 2025; Wang et al., 2025; Rukhovich et al., 2025; Mallis et al., 2025). Multimodal reasoning, retrieval augmentation (Lewis et al., 2020), and iterative visual feedback (Alrashedy et al., 2025; Wang et al., 2025; 2026b; Lin et al., 2026) further improve reconstruction, editing, and visual fidelity (Huang et al., 2024; Yin et al., 2026; He et al., 2026; Wang et al., 2026c). Procedural material graphs are likewise generated or inferred from images (Shi et al., 2020; Guerrero et al., 2022; Hu et al., 2023; Li et al., 2024; 2025), whereas inverse procedural modeling fits predefined procedural generators to targets (Talton et al., 2011; Vanegas et al., 2012; Stava et al., 2014; Pearl et al., 2025; Zhao et al., 2025) and remains bounded by their categories. Still, reliably synthesizing complex procedural geometry with rich geometric details and photorealistic appearance remains challenging. Benchmarks for 3D code generation (Gao et al., 2026; Wang et al., 2026c; Du et al., 2024; Alrashedy et al., 2025; Gu et al., 2025; Yang et al., 2026; Liu et al., 2026a) assess executability, geometric fidelity, and the benefits of execution feedback (Chen et al., 2024) and agentic coding (Yang et al., 2024a), but procedural modeling benchmarks primarily draw categories from predefined procedural factories (Gao et al., 2026; Raistrick et al., 2023; 2024), leaving open-ended procedural asset generation beyond predefined factories underexplored.

3D World Generation. Procedural content generation (Smelik et al., 2014; Hendrikx et al., 2013; Shaker et al., 2016) has long produced noise-based terrains (Perlin, 1985; Musgrave et al., 1989) and cities (Parish & Müller, 2001), and now yields photorealistic natural and indoor scenes (Raistrick et al., 2023; 2024; 2026), simulation-ready assets (Joshi et al., 2026), and houses for embodied AI (Deitke et al., 2022). Language models have been leveraged for 3D scene planning, spatial reasoning, and executable scene construction (Hu et al., 2024; Sun et al., 2023; Zhou et al., 2025; Zhang et al., 2026; Feng et al., 2023; Yang et al., 2024b; Çelen et al., 2024; Fu et al., 2024; Sun et al., 2025a; Aguina-Kang et al., 2024; Zhang et al., 2025b; Tam et al., 2025; Gumin et al., 2025), and one line couples them with existing procedural generators through function orchestration, parameter adjustment, and composition of predefined procedural assets (Sun et al., 2023; Zhou et al., 2025; Qian et al., 2024; Zhang et al., 2026; 2024b; Liu et al., 2025; Tang et al., 2025; Liu et al., 2026b). More recent approaches (Hu et al., 2024; Yin et al., 2026; Zhang et al., 2026) extend such pipelines with hierarchical generation (Fu et al., 2024; Sun et al., 2025b; Wang et al., 2024), iterative visual feedback (Ling et al., 2026), and physics-aware dynamic simulation (Lv et al., 2024). However, methods built on predefined procedural generators remain confined to their asset categories and parameter spaces, while semantically consistent and geometrically precise object placement in complex scenes remains challenging. Another line of work builds large-scale worlds (Guo et al., 2026; Wang et al., 2026a) from procedural layouts, retrieved assets (Deitke et al., 2023; Yang et al., 2024b; Aguina-Kang et al., 2024; Deng et al., 2024), generative 3D models (Poole et al., 2023; Hong et al., 2024; Zhang et al., 2024a; Xiang et al., 2025b), and object-level composition (Wang et al., 2024; Huang et al., 2025; Meng et al., 2026), expanding generatable content and enabling instance-level manipulation. However, these approaches and neural scene generators (Höllein et al., 2023; Fridman et al., 2023; Yu et al., 2024; 2025; HunyuanWorld Team et al., 2025; Cohen-Bar et al., 2023; Gao et al., 2024; Zhou et al., 2024; Engstler et al., 2025; Chen et al., 2023; Lin et al., 2023; Xie

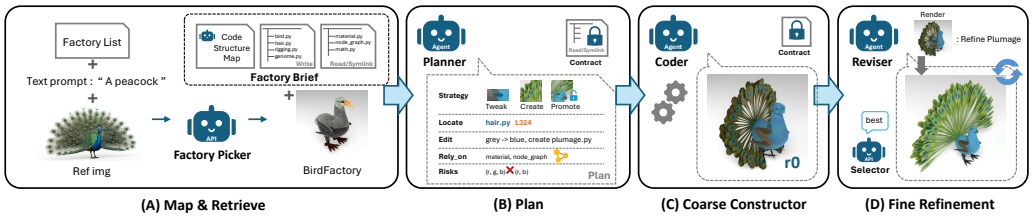

Figure 4: **Procedural asset generation through SCE.** (A) Factory mapping and retrieval. (B) Structured editing plan. (C) Coarse asset construction. (D) Render-guided fine refinement. Existing procedural algorithms are reused under explicit editing constraints.

et al., 2024; Wu et al., 2024; Meng et al., 2025) generally lack the reusable procedural structures and high-level parameters needed for procedural editing and seed-based geometric variation. In contrast, InfiniDream enables open-ended procedural asset generation beyond predefined factories and integrates it into scalable, fully procedural 3D world generation, while preserving procedural editability and seed-based diversity.

3 METHOD

We introduce **InfiniDream**, a VLM-driven framework for open-ended procedural 3D world generation. Given a text prompt, our framework selects a suitable initial scene and employs a Director to produce a structured Scene Plan specifying asset requirements, semantic placement constraints, and motion. Our framework consists of two core components: *Symbolically Constrained Asset Generation*, which creates novel procedural assets through constrained editing, and *Deterministic Frustum-Guided Grid Placement*, which translates semantic constraints into precise 3D object placements. Finally, procedural asset pooling, terrain refinement, and animation enable diverse, large-scale world generation.

3.1 SYMBOLICALLY CONSTRAINED ASSET GENERATION

Since directly providing a VLM with the entire Infinigen codebase often leads to imprecise editing (Fig. 3), we propose **Symbolically Constrained Editing (SCE)**, which extracts key code structures and explicit editing constraints, transforming a large procedural codebase into a manageable editing space (Fig. 4).

3.1.1 FACTORY MAPPING AND RETRIEVAL

Given a text description and an optional reference image, the system retrieves the most relevant Infinigen factory and constructs a *Factory Brief* (Fig. 4(A)). It contains a *Code Structure Map* identifying key functions and dependencies, an *Editable Surface* consisting of asset-specific modules dynamically identified through depth-limited dependency analysis, and *Linked Infrastructure* providing read-only access to shared low-level algorithms (e.g., node graphs) via symbolic links. Together, these components establish explicit code-level constraints that restrict editing while preserving the underlying procedural algorithms. Further implementation details are provided in Appendix G.

3.1.2 STRUCTURED PLANNING AND COARSE CONSTRUCTION

The Planner produces a structured *Editing Plan* specifying the editing strategy (*Strategy*), target locations (*Locate*), required modifications (*Edit*), dependencies (*Rely-on*), and potential risks (*Risks*), as shown in Fig. 4(B). The plan also accounts for seed-dependent logic to preserve controllable procedural variation.

We define three editing strategies: *Tweak* adjusts existing parameters and high-level logic; *Create* introduces new procedural functions; and *Promote* selectively extends editing permissions to protected modules when necessary. These strategies enable controlled expansion of the editing space.

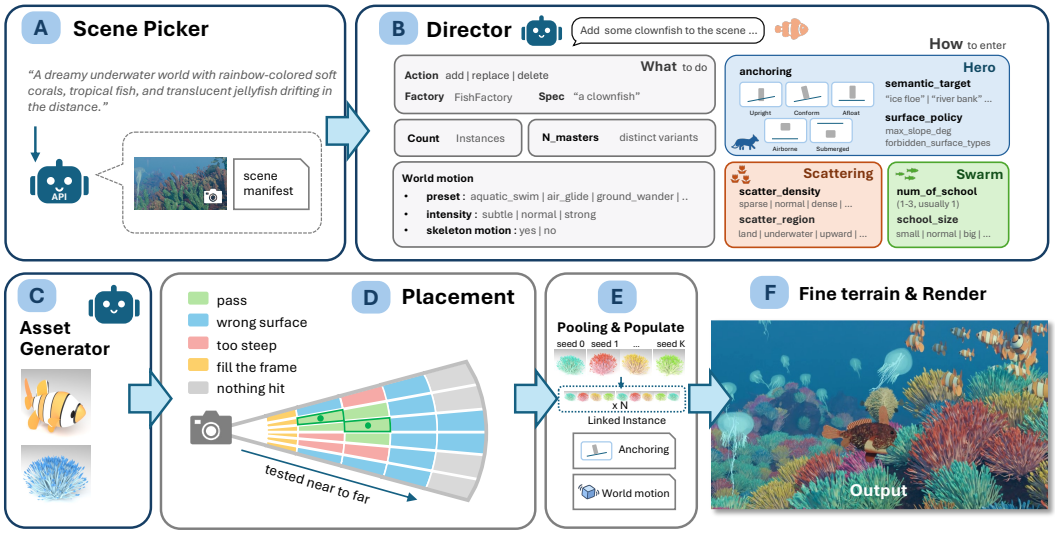

Figure 5: **Overview of InfiniDream's end-to-end world generation pipeline.** (A) Scene Picker collects the user instruction and current scene information. (B) Director generates a structured Scene Plan. (C) Asset Generator creates procedural assets. (D) Deterministic frustum-guided placement identifies valid object locations. (E) Population instantiates diverse assets. (F) Terrain refinement and rendering produce the final scene.

The Coarse Constructor executes the Editing Plan under these constraints (Fig. 4(C)), preserving the original factory entry points and function signatures to maintain compatibility with the population pipeline and random seed control. It reuses existing algorithms to construct a coarse procedural asset. SCE preserves seed-dependent randomness by adjusting sampling ranges rather than fixing parameters to match the reference image.

3.1.3 Fine Refinement

As shown in Fig. 4(D), the Reviser iteratively compares rendered results with the target description and reference image, identifying discrepancies in geometry, materials, and visual details. It then performs targeted code modifications within the established editing boundaries. A Selector identifies the best candidate across refinement iterations.

Through constrained editing and iterative refinement, SCE generates novel procedural assets while preserving procedural editability and seed-controlled diversity.

3.2 Semantic Placement Reasoning

Given a text prompt and current scene information, the Director first formulates a natural-language plan and converts it into a structured *Scene Plan* specifying editing operations (`add`, `replace`, and `delete`), asset requirements, and semantic placement constraints (Fig. 5(A–B)).

The Director selects among three placement modes: *Hero* for semantically significant objects, *Scattering* for large-area distributions, and *Swarm* for collective entities. Hero placement supports seven *Anchoring* types (`rooted`, `upright`, `tilted`, `conform`, `afloat`, `airborne`, and `submerged`). A *Semantic Target* and *Surface Policy* further specify target regions, permissible surface types, slope limits, and support conditions. Scattering and Swarm are controlled by density, coverage, and group size.

The Director also determines the instance count N, distinct procedural prototypes K, *World Motion*, and *Local Animation*. World Motion specifies object trajectories and swept envelopes for collision checking. VLM decisions are restricted to discrete semantic operations and constraints, leaving continuous geometric computations to deterministic placement.

3.3 Deterministic Frustum-Guided Grid Placement

To avoid repeated VLM-based coordinate prediction and costly render-based verification, we propose Deterministic Frustum-Guided Grid Placement, which translates the Director's semantic constraints into candidate 3D positions (Fig. 5(D)). Candidates are evaluated in a near-to-far order without additional VLM calls or iterative rendering.

3.3.1 Frustum-based Candidate Generation

Given the specified placement region, we discretize the camera's image plane into a 2D grid and generate 3D candidate positions along the corresponding camera rays. For surface-supported objects, candidates are determined by ray intersections with scene geometry. For airborne and submerged objects, candidates are sampled at discrete depth intervals along each ray.

3.3.2 Constraint-based Filtering and Depth-aware Placement

Each candidate undergoes five deterministic checks: (1) *Visibility*, (2) *Collision* with motion swept envelopes, (3) *Inter-instance Spacing*, (4) *Surface Policy*, including surface type, slope limits, and full support, and (5) *Projected-area Band*, requiring $a_{\min} \le a(p) \le a_{\max}$.

Given depth-sorted candidates $\mathcal{C} = \{p_1, \dots, p_n\}$, where $g_j(p) \in \{0, 1\}$ indicates satisfaction of constraint j, we select the nearest valid candidate:

$$i^* = \min \left\{ i \;\middle|\; \bigwedge_{j=1}^{5} g_j(p_i) = 1 \right\}, \qquad p^* = p_{i^*}. \tag{1}$$

Combined with projected-area bounds, near-to-far search enables implicit size-aware placement, favoring closer positions for smaller objects and farther positions for larger ones. For multiple instances, the initial search depth increases progressively, while previously placed objects are included in subsequent collision checks. If no valid position exists, hierarchical constraint relaxation is applied (Appendix H).

3.4 Scalable Population and Animation

We combine the procedural assets generated in Sec. 3.1 with the validated placements from Sec. 3.3 through semantically controlled asset pooling, terrain refinement, and procedural animation (Fig. 5(E)).

Semantically Budgeted Asset Pooling. To reduce the repeated construction cost of complex procedural assets, we extend Infinigen's asset pooling mechanism to independent object population. The Director determines the number of instances N and distinct procedural prototypes K based on scene requirements. For assets supporting seed-based variation, we generate K prototypes using different factory seeds and create N instances through linked duplication:

$$\mathcal{M} = \{M_k = G(s_k)\}_{k=1}^{K}, \qquad I_i = \mathrm{Clone}\left(M_{((i-1) \bmod K)+1}\right). \tag{2}$$

Here, G denotes the procedural generator, s_k is the seed for the k-th prototype, and I_i is the i-th instance. This mechanism reduces expensive asset constructions from N to K while maintaining seed-based diversity across prototypes. Assets unsuitable for pooling are generated independently.

Population, Terrain Refinement, and Animation. Procedural assets are placed according to their Anchoring modes and integrated with Infinigen's fine terrain pipeline. Animation combines *World Motion* and *Local Animation*, with shared motion specifications ensuring consistency between placement constraints and actual trajectories. The final scene is then refined and rendered (Fig. 5(E–F)).

Table 1: **Object-level comparison.** Results on 100 tasks across six categories.

Method	CLIP↑	SigLIP↑	DINO↑	IR↑	Exec.↑
VLM (One-shot)	27.21	-3.23	.419	-1.17	85
VLM (Harness)	29.20	-1.11	.498	-.82	**99**
VIGA	27.20	-3.99	.395	-1.19	90
BlenderRAG	27.56	-2.43	.406	-1.15	91
Infinigen	22.84	-7.90	.253	-1.90	–
Ours	**30.07**	**-0.12**	**.554**	**-.62**	98

Table 2: **Scene-level comparison.** Results on 20 scene generation tasks.

Method	CLIP↑	SigLIP↑	DINO↑	IR↑
VLM (Harness)	28.85	-2.566	.321	-.894
VIGA	27.14	-3.145	.246	-1.433
Infinigen	25.21	-7.88	.182	-1.71
Code2Worlds	22.76	-8.654	.114	-2.014
InfiniDream	**29.64**	**-0.68**	**.361**	**-.877**

4 Experiments

We evaluate InfiniDream at both the object and scene levels, complemented by diversity experiments, ablation studies, and a user study.

4.1 Evaluation Protocol and Baselines

Object-level Evaluation. We compare InfiniDream against five baselines: VLM (one-shot), VLM (harness), VIGA (Yin et al., 2026), BlenderRAG (Rondelli et al., 2026), and Infinigen (Raistrick et al., 2023). Following 3DCodeBench (Gao et al., 2026), we adopt its codebase for the two VLM baselines, evaluating single-turn code generation and iterative refinement with a native coding-agent harness. The remaining baselines represent visual-feedback refinement, retrieval-augmented generation, and original Infinigen factory outputs, respectively.

Scene-level Evaluation. We compare large-scale world generation against VLM (harness), VIGA (Yin et al., 2026), Infinigen (Raistrick et al., 2023), and Code2Worlds (Zhang et al., 2026), representing agentic code generation, visual-feedback refinement, original procedural scene generation, and LLM-driven hierarchical procedural orchestration, respectively.

Implementation and Metrics. We use Claude Opus 4.8 for InfiniDream and all VLM-based baselines, with Blender 4.x for generation and rendering. We evaluate 100 object-level tasks across six categories and 20 scene-level tasks at resolutions of 1024×1024 and 960×540, respectively.

We report CLIP (Radford et al., 2021), SigLIP2 (Tschannen et al., 2025), DINOv3 (Siméoni et al., 2025), and ImageReward (Xu et al., 2023). At the object level, CLIP, SigLIP2, and ImageReward are averaged over four fixed views ($45°$, $135°$, $225°$, and $315°$), while DINOv3 compares the front view with one independently generated reference image. We additionally report code executability. Scene-level metrics use each method's primary camera view. Further details are provided in the Appendix.

Diversity Evaluation. We compare InfiniDream with TRELLIS.2 (Xiang et al., 2025a) on 15 text prompts. For each prompt, InfiniDream generates five variations using different random seeds, while TRELLIS.2 uses five independently generated reference images. We measure semantic consistency using SigLIP2 and visual diversity using average pairwise DINOv2 feature distance.

4.2 Experimental Results

Qualitative Comparisons Figures 6 and 7 present qualitative comparisons of InfiniDream with existing methods at the object and scene levels, highlighting differences in geometric detail, photorealism, and scene composition.

Quantitative Comparisons As shown in Tables 1 and 2, InfiniDream achieves the highest scores across all four perceptual metrics at both the object and scene levels.

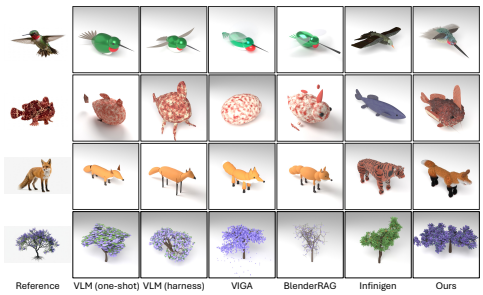

Figure 6: **Qualitative comparison of object-level generation.** Generated 3D assets under identical text prompts.



Figure 7: **From text to richly detailed 3D worlds.** Qualitative comparisons across four diverse environments, highlighting scene richness, visual fidelity, and semantic alignment. All methods use identical text prompts.

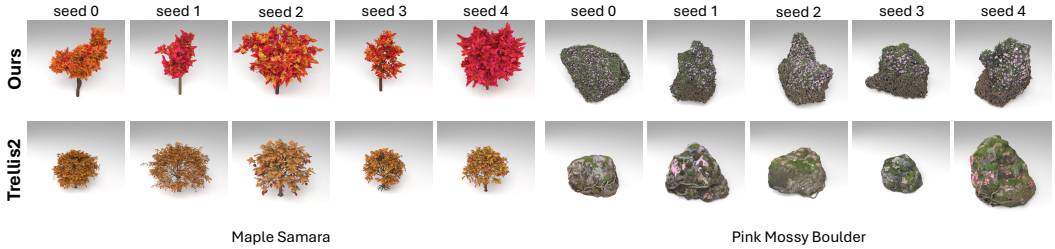

Figure 8: **High-quality and diverse procedural assets.** InfiniDream generates five variations by changing random seeds, while TRELLIS.2 uses five independently generated reference images.

Human Preference Study We conduct a human preference study with 30 participants across 10 object and 10 scene tasks. Participants compare results from competing methods based on semantic alignment and visual quality. Table 3 reports InfiniDream's preference rates for each criterion and overall.

Table 3: **Human preference study.** Preference rates for InfiniDream.

Level	Align. (%)	Quality (%)	Overall (%)
Object	81.3	73.7	**77.5**
Scene	91.0	75.3	**83.2**

4.3 Diversity Analysis

For each text prompt, InfiniDream generates five variations using different random seeds, while TRELLIS.2 uses five independently generated reference images. We evaluate semantic consistency using SigLIP2 and visual diversity using average pairwise DINOv2 feature distance. InfiniDream achieves greater visual diversity while maintaining comparable semantic alignment (Fig. 8, Table 4).

Table 4: **Diversity comparison.** Five variations per prompt.

Method	SigLIP2↑	DINOv2 Dist.↑
TRELLIS.2	**0.1656**	0.6084
Ours	0.1531	**0.6704**

4.4 Ablation Studies

Object-level Ablation We evaluate six ablation settings on 25 animal and plant tasks: (1) *w/o Ref. Image*, (2) *Random Factory*, (3) *w/o Planner*, (4) *Agent (Infinigen)*, (5) *w/o Refiner*, and (6) *3 Rounds*. The full pipeline allows up to six refinement rounds. For Agent (Infinigen), Claude Opus 4.8 is given the entire Infinigen codebase and Bash access to autonomously write and edit code,

Table 5: **Quantitative object-level ablation.** Evaluation of generation components and refinement strategies across 25 animal and plant tasks, measuring semantic alignment, visual similarity, and perceptual quality.

Configuration	CLIP↑	SigLIP-2↑	DINOv3↑	IR↑
w/o Ref. Image	28.68	-1.70	0.363	-0.978
Random Factory	27.39	-3.73	0.296	-1.520
w/o Planner	28.72	-2.41	0.368	-1.023
Agent (Infinigen)	29.01	-1.64	0.387	-0.968
w/o Refiner	28.80	-3.34	0.334	-1.420
3 Rounds	29.33	-1.51	0.438	-0.824
Full	**30.37**	**-0.81**	**0.494**	**-0.580**

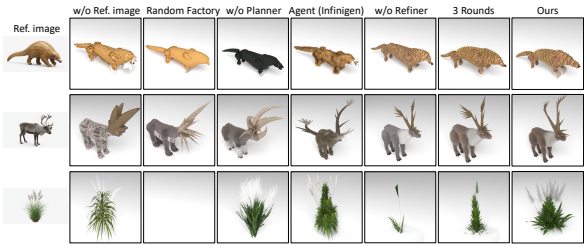

Figure 9: **Qualitative object-level ablation.** Visual comparison of different generation strategies, illustrating the effects of reference images, factory selection, structured planning, and iterative refinement on asset geometry, fine-grained details, and visual appearance.

Table 6: **Quantitative scene-level placement ablation.** Evaluation on 20 scenes measuring object readability, ground contact, and contact spread.

Configuration	Unread. (%)↓	Ground P95 (m)↓	Spread P95 (m)↓
w/o Grid Search	27.4	**0.15**	0.25
w/o Anchoring	40.3	0.48	0.74
w/o Semantic Target	14.0	**0.15**	0.38
Agent (Placement)	50.9	0.54	0.75
Full (Ours)	**7.8**	**0.15**	**0.24**



Figure 10: **Qualitative scene-level placement ablation.** Visual comparison of different placement strategies, illustrating the effects of semantic planning, grid-based placement, and anchoring constraints on object visibility, spatial arrangement, and scene composition.

render and inspect results, and determine when to stop within six rounds, without our structured planning or editing constraints. Quantitative and qualitative results are shown in Table 5 and Fig. 9.

Scene-level Ablation. We evaluate four settings on 20 scenes: (1) *w/o Grid Search*, using direct 3D coordinate prediction; (2) *w/o Anchoring*, reverting to Infinigen's grounding; (3) *w/o Semantic Target*, removing semantic surface constraints; and (4) *Agent (Placement)*, allowing a VLM agent to autonomously place our generated 3D assets in the same selected coarse scene given the text prompt, without our placement pipeline.

We evaluate three placement metrics: *Unreadable Rate*, the percentage of objects outside the camera view or occupying less than 0.1% of the image area; *Ground-contact Error*, the distance between an object's lowest contact point and its supporting surface, capturing floating or penetration; and *Contact Spread*, the difference between the maximum and minimum terrain-relative gaps across sampled contact points, capturing uneven grounding and partial suspension. We report the 95th percentile (P95) for the latter two metrics. For Contact Spread, P95 is computed over per-object values pooled across all requested-add objects in the 20 scenes. Results are shown in Table 6 and Fig. 10.

5 Conclusion

We presented **InfiniDream**, a VLM-driven framework for open-ended procedural 3D world generation. By combining Symbolically Constrained Asset Generation with semantic planning and deterministic placement, InfiniDream creates novel, photorealistic, and editable procedural assets and composes them into diverse, large-scale 3D worlds. Extensive experiments demonstrate its effectiveness at both the object and scene levels.

Limitations and Future Work. Our current animation pipeline relies on a predefined set of motion types. Future work could explore VLM-generated animation code to support more complex and customized motions. We also aim to extend InfiniDream to indoor environments with more complex spatial relationships and object interactions.

AI USE STATEMENT

We used generative AI tools to assist with code development, literature summarization, and manuscript drafting and editing. VLMs were also employed as core components of our framework and experimental baselines, as detailed in the paper. All AI-assisted content was reviewed and validated by the authors, who take full responsibility for this work.

REPRODUCIBILITY STATEMENT

We provide detailed implementation and evaluation protocols in the main paper and Appendix. To facilitate reproducibility, we will publicly release our source code, experimental configurations, benchmark prompts, and evaluation scripts. These resources will enable researchers to reproduce our experiments and extend InfiniDream to new procedural assets and 3D worlds.

REFERENCES

Rio Aguina-Kang, Maxim Gumin, Do Heon Han, Stewart Morris, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Qiuhong Anna Wei, Kailiang Fu, and Daniel Ritchie. Open-universe indoor scene generation using LLM program synthesis and uncurated object databases. *arXiv preprint arXiv:2403.09675*, 2024.

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating CAD code with vision-language models for 3D designs. In *International Conference on Learning Representations (ICLR)*, 2025.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-Design: Personalized LLM interior designer. In *Computer Vision – ECCV 2024 Workshops*, volume 15624 of *Lecture Notes in Computer Science*, pp. 217–234, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. SceneDreamer: Unbounded 3D scene generation from 2D image collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):15562–15576, 2023.

Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Set-the-scene: Global-local training for generating controllable NeRF scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 2920–2929, 2023.

Bingquan Dai, Li Ray Luo, Qihong Tang, Jie Wang, Xinyu Lian, Hao Xu, Minghan Qin, Xudong Xu, Bo Dai, Haoqian Wang, et al. MeshCoder: LLM-powered structured mesh code generation from point clouds. *arXiv preprint arXiv:2508.14879*, 2025.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 5982–5994, 2022.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13142–13153, 2023.

Jie Deng, Wenhao Chai, Junsheng Huang, Zhonghan Zhao, Qixuan Huang, Mingyan Gao, Jianshu Guo, Shengyu Hao, Wenhao Hu, Jenq-Neng Hwang, Xi Li, and Gaoang Wang. CityCraft: A real crafter for 3D city generation. *arXiv preprint arXiv:2406.04983*, 2024.

Yuhao Du, Shunian Chen, Wenbo Zan, Peizhao Li, Mingxuan Wang, Dingjie Song, Bo Li, Yan Hu, and Benyou Wang. Blenderllm: Training large language models for computer-aided design with self-improvement, 2024. URL https://arxiv.org/abs/2412.14203.

Paul Engstler, Aleksandar Shtedritski, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. SynCity: Training-free generation of 3D worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27585–27595, 2025.

Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 18225–18250, 2023.

Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. SceneScape: Text-driven consistent scene generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 39897–39914, 2023.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-vocabulary generation of structured and textured 3D homes. In *European Conference on Computer Vision (ECCV)*, pp. 52–70, 2024.

Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. GraphDreamer: Compositional 3D scene synthesis from scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21295–21304, 2024.

Yipeng Gao, Lei Shu, Genzhi Ye, Xi Xiong, Ameesh Makadia, Meiqi Guo, Laurent Itti, and Jindong Chen. 3dcodebench: Benchmarking agentic procedural 3d modeling via code, 2026. URL https://arxiv.org/abs/2606.01057.

Yunqi Gu, Ian Huang, Jihyeon Je, Guandao Yang, and Leonidas Guibas. BlenderGym: Benchmarking foundational model systems for graphics editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18574–18583, 2025.

Paul Guerrero, Miloš Hašan, Kalyan Sunkavalli, Radomír Měch, Tamy Boubekeur, and Niloy J. Mitra. MatFormer: A generative model for procedural materials. *ACM Transactions on Graphics*, 41(4):46:1–46:12, 2022.

Maxim Gumin, Do Heon Han, Seung Jean Yoo, Aditya Ganeshan, R. Kenny Jones, Kailiang Fu, Rio Aguina-Kang, Stewart Morris, and Daniel Ritchie. Procedural scene programs for open-universe scene generation: LLM-free error correction via program search. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 141:1–141:11, 2025. doi: 10.1145/3757377.3763930.

Chunchao Guo, Jinpeng Li, Yang Li, and Zilong Huang. Worldclaw: Agentic 3d open-world generation at scale, 2026. URL https://arxiv.org/abs/2608.05248.

Guangzhao He, Rundong Luo, Wei-Chiu Ma, and Hadar Averbuch-Elor. Thinking in blender: Staged executable inverse graphics with vision-language models, 2026.

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

Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7909–7920, 2023.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *International Conference on Learning Representations (ICLR)*, 2024.

Yiwei Hu, Paul Guerrero, Miloš Hašan, Holly Rushmeier, and Valentin Deschaintre. Generating procedural materials from text or image prompts. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–11, 2023.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 19252–19282. PMLR, 2024.

Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *Proceedings of the European Conference on Computer Vision*, 2024. doi: 10.1007/978-3-031-73024-5_18.

Zehuan Huang, Yuan-Chen Guo, Xingqiao An, Yunhan Yang, Yangguang Li, Zi-Xin Zou, Ding Liang, Xihui Liu, Yan-Pei Cao, and Lu Sheng. MIDI: Multi-instance diffusion for single image to 3D scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23646–23657, 2025.

HunyuanWorld Team, Zhenwei Wang, Yuhao Liu, Junta Wu, Zixiao Gu, Haoyuan Wang, Xuhui Zuo, Tianyu Huang, Wenhuan Li, Sheng Zhang, et al. HunyuanWorld 1.0: Generating immersive, explorable, and interactive 3D worlds from words or pixels. *arXiv preprint arXiv:2507.21809*, 2025.

R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeAssembly: Learning to generate programs for 3D shape structure synthesis. *ACM Transactions on Graphics*, 39(6), 2020.

R. Kenny Jones, David Charatan, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeMOD: Macro operation discovery for 3D shape programs. *ACM Transactions on Graphics*, 40(4), 2021.

R. Kenny Jones, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeCoder: Discovering abstractions for visual programs from unstructured primitives. *ACM Transactions on Graphics*, 42(4):49:1–49:17, 2023.

R. Kenny Jones, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeLib: Designing a library of programmatic 3D shape abstractions with large language models. *ACM Transactions on Graphics*, 2026. doi: 10.1145/3840293.

Abhishek Joshi, Beining Han, Jack Nugent, Max Gonzalez Saez-Diez, Yiming Zuo, Jonathan Liu, Hongyu Wen, Stamatis Alexandropoulos, Karhan Kayan, Anna Calveri, Tao Sun, Gaowen Liu, Yi Shao, Alexander Raistrick, and Jia Deng. Procedural generation of articulated simulation-ready assets. In *Conference on Robot Learning (CoRL)*, 2026. arXiv:2505.10755.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating sequential CAD designs from beginner-to-expert level text prompts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 7552–7579, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 9459–9474, 2020.

Beichen Li, Yiwei Hu, Paul Guerrero, Miloš Hašan, Liang Shi, Valentin Deschaintre, and Wojciech Matusik. Procedural material generation with reinforcement learning. *ACM Transactions on Graphics*, 43(6), 2024. doi: 10.1145/3687979.

Beichen Li, Rundi Wu, Armando Solar-Lezama, Changxi Zheng, Liang Shi, Bernd Bickel, and Wojciech Matusik. VLMaterial: Procedural material generation with large vision-language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. InfiniCity: Infinite-scale city synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22808–22818, 2023.

Youtian Lin, Yikang Yang, Zhanpeng Hu, Mengqi Zhou, Feihu Zhang, Xun Cao, Jiaheng Liu, and Yao Yao. Procedura: Agentic 3D modeling with procedural control. *arXiv preprint arXiv:2608.26238*, 2026.

Aristid Lindenmayer. Mathematical models for cellular interactions in development I. filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

Lu Ling, Chen-Hsuan Lin, Tsung-Yi Lin, Yifan Ding, Yu Zeng, Yichen Sheng, Yunhao Ge, Ming-Yu Liu, Aniket Bera, and Zhaoshuo Li. Scenethesis: A language and vision agentic framework for 3D scene generation. In *International Conference on Learning Representations (ICLR)*, 2026.

Ling Liu, Bingchen Gong, Amal Dev Parakkat, and Maks Ovsjanikov. 3DHarnessBench: Probing agentic 3D-to-code capabilities of frontier vision-language models. *arXiv preprint arXiv:2609.06535*, 2026a.

Xinhang Liu, Chi-Keung Tang, and Yu-Wing Tai. WorldCraft: Photo-realistic 3D world creation and customization via LLM agents. *arXiv preprint arXiv:2502.15601*, 2025.

Zishan Liu, Zecong Tang, RuoCheng Wu, Xinzhe Zheng, Jingyu Hu, Ka-Hei Hui, Haoran Xie, Bo Dai, and Zhengzhe Liu. Imagine a city: CityGenAgent for procedural 3D city generation. *arXiv preprint arXiv:2602.05362*, 2026b.

Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. Ll3m: Large language 3d modelers, 2025.

Jiaxi Lv, Yi Huang, Mingfu Yan, Jiancheng Huang, Jianzhuang Liu, Yifan Liu, Yafei Wen, Xiaoxin Chen, and Shifeng Chen. GPT4Motion: Scripting physical motions in text-to-video generation via blender-oriented GPT planning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1430–1440, 2024.

Dimitrios Mallis, Ahmet Serdar Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-Assistant: Tool-augmented VLLMs as generic CAD task solvers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7284–7294, 2025.

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. LT3SD: Latent trees for 3D scene diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 650–660, 2025.

Yanxu Meng, Haoning Wu, Ya Zhang, and Weidi Xie. SceneGen: Single-image 3D scene generation in one feedforward pass. In *International Conference on 3D Vision (3DV)*, 2026. arXiv:2508.15769.

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Transactions on Graphics*, 25(3):614–623, 2006.

F. Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. The synthesis and rendering of eroded fractal terrains. *ACM SIGGRAPH Computer Graphics*, 23(3):41–50, 1989.

Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 301–308, 2001.

Ofek Pearl, Itai Lang, Yuhua Hu, Raymond A. Yeh, and Rana Hanocka. GeoCode: Interpretable shape programs. *Computer Graphics Forum*, pp. e15276, 2025. doi: 10.1111/cgf.15276.

Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

Zhaofang Qian, Abolfazl Sharifi, Tucker Carroll, and Ser-Nam Lim. Scene co-pilot: Procedural text to video generation with human in the loop, 2024. URL https://arxiv.org/abs/2411.18644.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite photorealistic worlds using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12630–12641, 2023.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21783–21794, 2024.

Alexander Raistrick, Karhan Kayan, Jack Nugent, David Yan, Lingjie Mei, Meenal Parakh, Hongyu Wen, Dylan Li, Yiming Zuo, Erich Liang, and Jia Deng. ProcFunc: Function-oriented abstractions for procedural 3D generation in Python. *arXiv preprint arXiv:2604.26943*, 2026.

Daniel Ritchie, Paul Guerrero, R. Kenny Jones, Niloy J. Mitra, Adriana Schulz, Karl D. D. Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. *Computer Graphics Forum*, 42(2): 545–568, 2023.

Massimo Rondelli, Francesco Pivi, and Maurizio Gabbrielli. Blenderrag: High-fidelity 3d object generation via retrieval-augmented code synthesis, 2026.

Danila Rukhovich, Elona Dupont, Dimitrios Mallis, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-Recode: Reverse engineering CAD code from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9801–9811, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games*. Springer, 2016.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5515–5523, 2018.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomír Měch, and Wojciech Matusik. MATch: Differentiable material graphs for procedural material capture. *ACM Transactions on Graphics*, 39(6), 2020.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. Dinov3, 2025.

Ruben M. Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, 33(6):31–50, 2014.

Ondrej Stava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. Inverse procedural modelling of trees. *Computer Graphics Forum*, 33(6):118–131, 2014.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models, 2023.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable optimization of 3D layout via vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29469–29478, 2025a.

Weilin Sun, Xinran Li, Manyi Li, Kai Xu, Xiangxu Meng, and Lei Meng. Hierarchically-structured open-vocabulary indoor scene synthesis with pre-trained large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7122–7130, 2025b.

Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Měch, and Vladlen Koltun. Metropolis procedural modeling. *ACM Transactions on Graphics*, 30(2):11:1–11:14, 2011.

Hou In Ivan Tam, Hou In Derek Pun, Austin T. Wang, Angel X. Chang, and Manolis Savva. SceneMotifCoder: Example-driven visual program learning for generating 3D object arrangements. In *International Conference on 3D Vision (3DV)*, pp. 179–188, 2025.

Song Tang, Kaiyong Zhao, Lei Wang, Yuliang Li, Xuebo Liu, Junyi Zou, Qiang Wang, and Xiaowen Chu. UnrealLLM: Towards highly controllable and interactable 3D scene generation by LLM-powered procedural content generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19417–19435, 2025. doi: 10.18653/v1/2025.findings-acl.994.

Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to infer and execute 3D shape programs. In *International Conference on Learning Representations (ICLR)*, 2019.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL `https://arxiv.org/abs/2502.14786`.

Carlos A. Vanegas, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell. Inverse design of urban procedural models. *ACM Transactions on Graphics*, 31(6), 2012.

Dilin Wang, Hyunyoung Jung, Tom Monnier, Kihyuk Sohn, Chuhang Zou, Xiaoyu Xiang, Yu-Ying Yeh, Di Liu, Zixuan Huang, Thu Nguyen-Phuoc, Yuchen Fan, Sergiu Oprea, Ziyan Wang, Roman Shapovalov, Nikolaos Sarafianos, Thibault Groueix, Antoine Toisoul, Prithviraj Dhar, Xiao Chu, Minghao Chen, Geon Yeong Park, Rakesh Ranjan, and Andrea Vedaldi. Worldgen: From text to traversable and interactive 3d worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27124–27135, 2026a.

Rui-Huan Wang, Si-Tong Wei, Jia-Qi He, Heng-Yi Wei, Baoquan Chen, and Peng-Shuai Wang. aDSL: Agentic 3D creation via joint agent-program design. *arXiv preprint arXiv:2608.17975*, 2026b.

Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-CAD generation through infusing visual feedback in large language models. In *International Conference on Machine Learning (ICML)*, 2025.

Yian Wang, Xiaowen Qiu, Jiageng Liu, Zhehuan Chen, Jiting Cai, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Architect: Generating vivid and interactive 3D scenes with hierarchical 2D inpainting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 67575–67603, 2024.

Yuhao Wang, Puyi Wang, Linjie Li, Zhengyuan Yang, Kevin Qinghong Lin, and Yu Cheng. 3d-cos: A new 3d reconstruction paradigm based on vlm code synthesis, 2026c.

Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Transactions on Graphics*, 22(3):669–677, 2003.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6772–6782, 2021.

Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, and Pan Ji. BlockFusion: Expandable 3D scene generation using latent tri-plane extrapolation. *ACM Transactions on Graphics*, 43(4), 2024. doi: 10.1145/3658188.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation, 2025a. URL https://arxiv.org/abs/2512.14692.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2025b.

Haozhe Xie, Zhaoxi Chen, Fangzhou Hong, and Ziwei Liu. CityDreamer: Compositional generative model of unbounded 3D cities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9666–9675, 2024.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation, 2023. URL https://arxiv.org/abs/2304.05977.

Yutaro Yamada, Khyathi Chandu, Bill Yuchen Lin, Jack Hessel, Ilker Yildirim, and Yejin Choi. L3GO: Language agents with chain-of-3D-thoughts for generating unconventional objects. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (System Demonstrations)*, pp. 456–469, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 50528–50652, 2024a.

Yikang Yang, Zhanpeng Hu, Youtian Lin, Mengqi Zhou, Jingxi Xu, Feihu Zhang, Jiaheng Liu, and Yao Yao. P3D-Bench: Benchmarking MLLMs for parametric 3D generation and structural reasoning. *arXiv preprint arXiv:2606.11152*, 2026.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3D embodied AI environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16227–16237, 2024b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shaofeng Yin, Jiaxin Ge, Zora Zhiruo Wang, Chenyang Wang, Xiuyu Li, Michael J. Black, Trevor Darrell, Angjoo Kanazawa, and Haiwen Feng. Vision-as-inverse-graphics agent via interleaved multimodal reasoning, 2026.

Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T. Freeman, Forrester Cole, Deqing Sun, Noah Snavely, Jiajun Wu, and Charles Herrmann. WonderJourney: Going from anywhere to everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6658–6667, 2024.

Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T. Freeman, and Jiajun Wu. WonderWorld: Interactive 3D scene generation from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5916–5926, 2025.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. *ACM Transactions on Graphics*, 43(4), 2024a.

Shougao Zhang, Mengqi Zhou, Yuxi Wang, Chuanchen Luo, Rongyu Wang, Yiwei Li, Zhaoxiang Zhang, and Junran Peng. CityX: Controllable procedural content generation for unbounded 3D cities. *arXiv preprint arXiv:2407.17572*, 2024b.

Shuyuan Zhang, Chenhan Jiang, Zuoou Li, and Jiankang Deng. ShapeCraft: LLM agents for structured, textured and interactive 3D modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Yi Zhang, Yunshuang Wang, Zeyu Zhang, and Hao Tang. Code2worlds: Empowering coding llms for 4d world generation, 2026.

Yunzhi Zhang, Zizhang Li, Matt Zhou, Shangzhe Wu, and Jiajun Wu. The scene language: Representing scenes with programs, words, and embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24625–24634, 2025b.

Wang Zhao, Yan-Pei Cao, Jiale Xu, Yuejiang Dong, and Ying Shan. DI-PCG: Diffusion-based efficient inverse procedural content generation for high-quality 3D asset creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11061–11072, 2025.

Mengqi Zhou, Yuxi Wang, Jun Hou, Shougao Zhang, Yiwei Li, Chuanchen Luo, Junran Peng, and Zhaoxiang Zhang. Scenex: Procedural controllable large-scale scene generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(10):10806–10814, 2025. doi: 10.1609/aaai.v39i10.33174.

Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. GALA3D: Towards text-to-3D complex scene generation via layout-guided generative Gaussian splatting. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62108–62118, 2024.

A Additional Object-level Comparisons

We provide additional qualitative comparisons of object-level generation across diverse asset categories in Fig. 11. All methods are evaluated using identical text prompts under the same experimental protocol described in Sec. 4.1. These additional examples further illustrate differences in geometric complexity, material details, and visual fidelity among the compared methods.

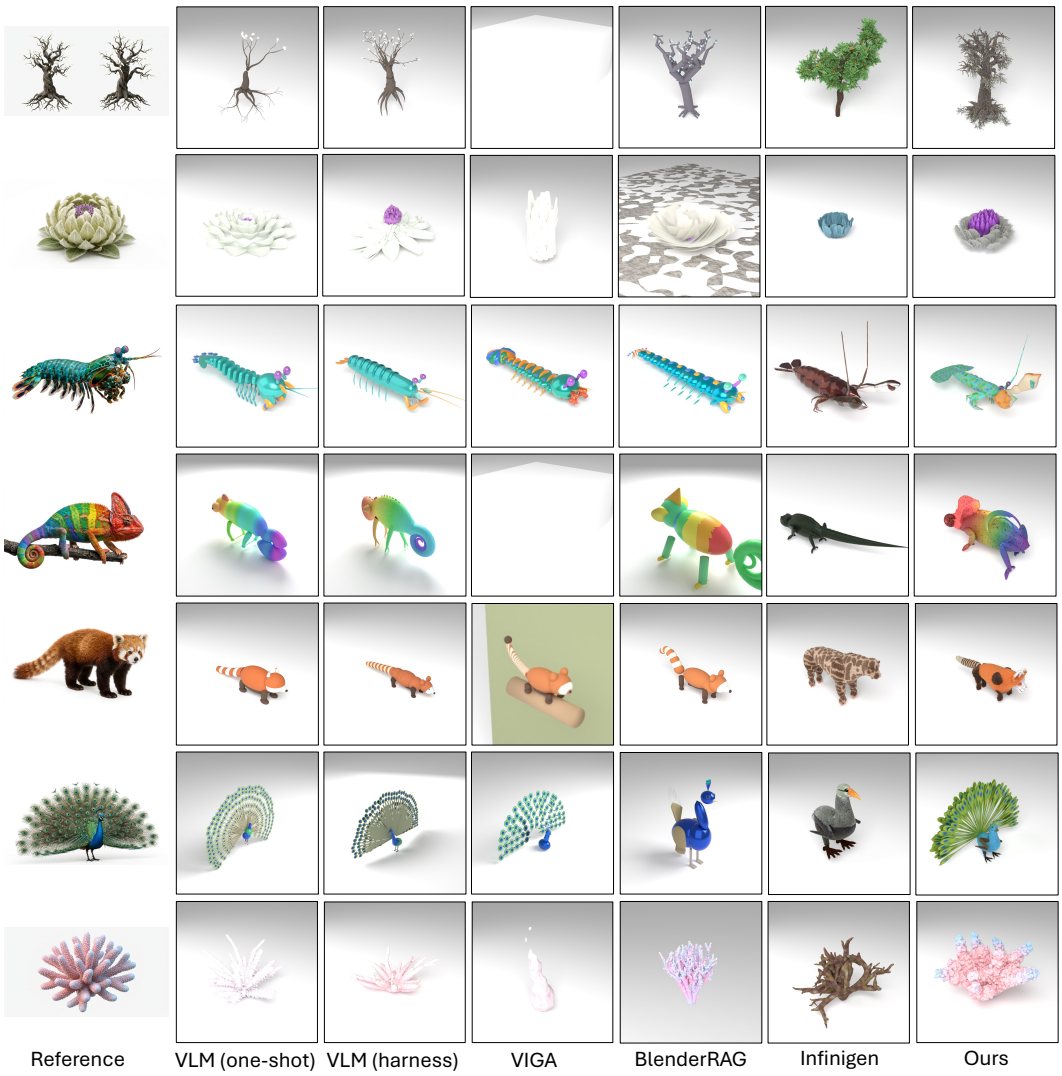

Figure 11: **Additional qualitative comparisons of object-level generation.** More examples comparing InfiniDream with existing methods under identical text prompts.

B Additional Scene-level Comparisons

We provide additional qualitative comparisons of scene-level generation across diverse environments in Fig. 12. All methods are evaluated using identical text prompts under the same experimental protocol described in Sec. 4.1. These additional results illustrate differences in scene composition, geometric detail, semantic alignment, and visual fidelity across the compared methods.

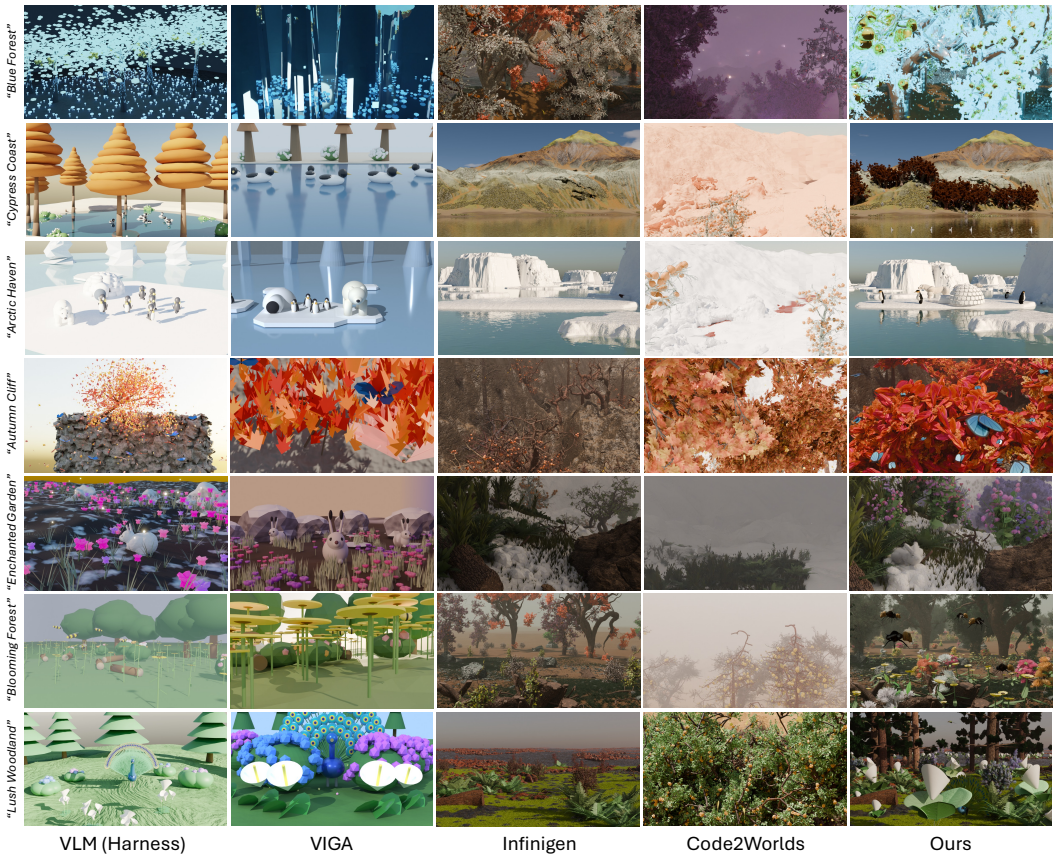

Figure 12: **Additional qualitative comparisons of scene-level generation.** More examples of 3D world generation across diverse environments under identical text prompts.

C Additional Qualitative Comparison with GPT-6 Astra

We provide an additional qualitative comparison with GPT-6 Astra in Fig. 13. This comparison is included as supplementary evidence beyond the main benchmark suite. Importantly, InfiniDream still uses Claude Opus 4.8 as the underlying VLM, while GPT-6 Astra serves as an additional strong proprietary baseline. Under identical text prompts, the results highlight differences in visual fidelity, geometric detail, material richness, and overall semantic composition.



Figure 13: **Additional qualitative comparison with GPT-6 Astra.** We compare InfiniDream, implemented with Claude Opus 4.8, against GPT-6 Astra on representative examples under identical text prompts. The results provide additional evidence of InfiniDream's ability to generate visually rich and semantically coherent procedural assets and scenes.

D Additional Diversity Results

We provide additional qualitative diversity comparisons between InfiniDream and TRELLIS.2 in Fig. 14. For each text prompt, InfiniDream generates five procedural variations by changing random seeds, while TRELLIS.2 generates five assets from independently generated reference images. These examples further illustrate the diversity of InfiniDream's procedural assets while preserving semantic consistency and visual quality.

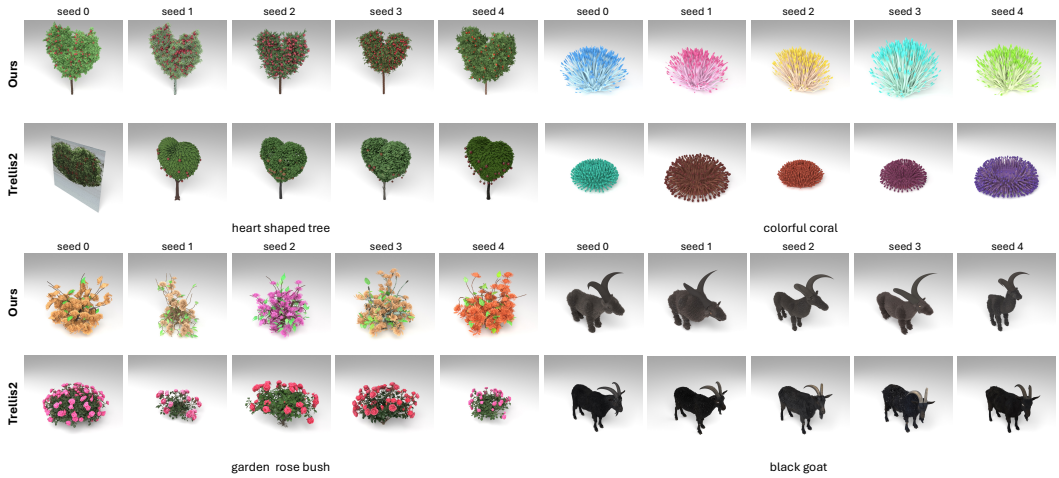

Figure 14: **Additional qualitative diversity comparisons with TRELLIS.2.** Five variations are generated for each text prompt. InfiniDream achieves procedural diversity by varying random seeds, whereas TRELLIS.2 uses independently generated reference images.

E Object-level Comparison with Code2Worlds

We further compare InfiniDream with Code2Worlds on the 29 out of 100 object-level tasks supported by Code2Worlds. Due to its limited factory coverage, we report this comparison separately from the main benchmark.

As shown in Fig. 15 and Table 7, InfiniDream achieves higher scores across all four metrics. Code2Worlds can exploit the parameter space of existing factories, such as adjusting the color of a fish, but cannot introduce novel procedural features beyond predefined generators. In contrast, InfiniDream extends existing procedural code to generate new geometric structures and fine-grained details, enabling more flexible open-ended asset generation.

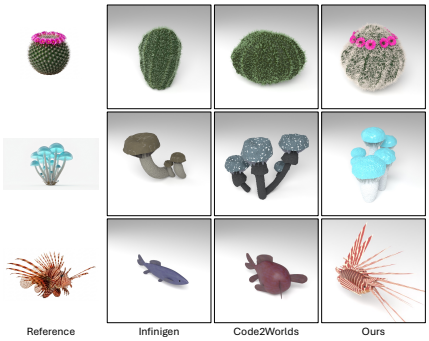

Figure 15: **Object-level comparison with Code2Worlds.**

Table 7: **Quantitative comparison.** Results on 29 supported tasks.

Metric	Ours	Code2Worlds
CLIP ↑	**30.24**	22.56
SigLIP2 ↑	**0.49**	-7.76
ImageReward ↑	**-0.35**	-1.94
DINOv3 ↑	**0.583**	0.281

F Man-Made Asset Generation

Beyond natural assets, InfiniDream can retrieve and extend factories for man-made objects. As shown in Fig. 16, while VIGA performs well on geometric objects, InfiniDream achieves comparable structural quality with rich geometric and material details, demonstrating its versatility across both organic and man-made assets.

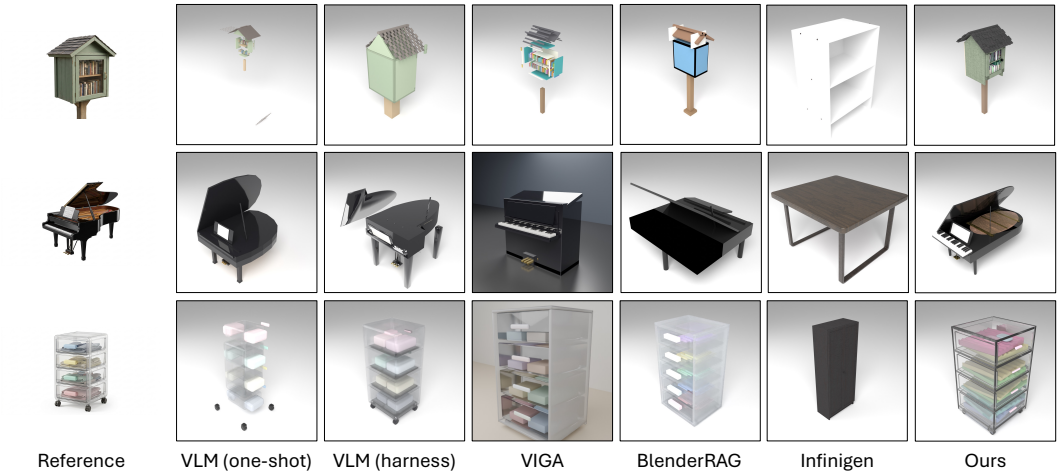

Figure 16: **Qualitative comparison of man-made asset generation.** InfiniDream produces structurally coherent man-made objects with rich geometric and material details.

G Implementation Details of Symbolically Constrained Editing

Reference Synthesis and Factory Retrieval. We use Gemini 3 Pro Image to generate an isolated reference object against a white background. Claude Opus 4.8 retrieves the most relevant Infinigen factory, with deterministic lexical matching as a fallback.

Factory Brief Construction. Starting from the selected factory, we perform AST-based import graph traversal with a maximum depth of two. Asset-specific modules form the Editable Surface, while shared modules remain read-only Linked Infrastructure. Unrelated dependencies are pruned, and a VLM summarizes the remaining code into a cached Code Structure Map.

Constrained Code Editing. The Planner produces an Editing Plan specifying Strategy, Locate, Edits, Rely-on, and Risks. The Constructor can modify existing modules (Tweak), promote protected modules (Promote), or synthesize new procedural modules (Create). Editing is restricted to the working directory, preserving the original factory interface and seed-dependent randomness.

Iterative Refinement. Modified modules are integrated into a symbolic-link overlay of the original Infinigen codebase. A vision-based Verifier guides targeted refinement, while execution failures trigger traceback-based repair. The pipeline supports up to six rounds with early stopping. A vision-based Selector chooses the best successful candidate, whose code snapshot is used to reconstruct the final executable generator.

H Implementation Details of Deterministic Frustum-Guided Grid Placement

Candidate Generation. We discretize the Director's screen-space region into a 9×5 grid (17×9 for six or more instances). Surface candidates are obtained through ray-scene intersections, while airborne and submerged candidates are sampled along camera rays using geometric depth intervals ($\times 1.3$, up to 80 m). Larger objects are placed first, with deterministic hashing ensuring reproducibility. The entire placement process requires no additional VLM calls.

Constraints and Relaxation. Projected area is estimated from the proxy bounding box, with a preferred minimum of 1.2%, a relaxed minimum of 0.2%, and an absolute maximum of 25%. Collision checks account for motion-dependent swept envelopes, while surface placement additionally enforces slope limits and four-corner footprint support. When no valid candidate is found, con-

straints are progressively relaxed and the search region is expanded. Water-related constraints remain enforced, and all relaxations are recorded in the resulting placement plan.

I Baseline Implementation Details

Baseline Implementations. We adopt the official implementations of all baselines, adding Claude Opus 4.8 API adapters where necessary. The VLM (one-shot) and VLM (harness) baselines follow 3DCodeBench, with the latter supporting autonomous code execution and iterative refinement.

Object-level Baselines. All methods receive identical text prompts and reference images, except BlenderRAG, which only supports text inputs. All assets are evaluated under identical camera, lighting, background, and rendering settings. Failed generations are retained as blank renders rather than excluded.

Scene-level Baselines. All scene-level methods receive identical text prompts and follow their respective official scene generation and camera selection pipelines. For InfiniDream, the Scene Picker automatically retrieves the most relevant initial coarse scene based on the input prompt from a library of 70 pre-generated environments (14 biomes with five seeds each), which is then used as the basis for subsequent procedural editing. The other baselines do not use this library. Code2Worlds selects from its available procedural factories even when no closely matching category exists.

Failure Handling. We resolve version- and environment-related execution issues wherever possible to ensure that baselines run as intended. We do not report a separate scene-level executability metric.

J Ablation Study Details

Object-level Ablation. We provide additional implementation details for the six ablation settings: (1) *w/o Ref. Image* removes the reference image from the generation pipeline; (2) *Random Factory* replaces factory retrieval with random selection; (3) *w/o Planner* removes structured planning and replaces the Editing Plan with empty content; (4) *Agent (Infinigen)* gives a Claude Opus 4.8 coding-agent harness access to the entire Infinigen codebase and Bash tools, allowing autonomous code generation, rendering, and refinement without SCE constraints, for up to six rounds; (5) *w/o Refiner* removes fine-level refinement; and (6) *3 Rounds* limits the refinement budget to three rounds, compared with six in the full method.

Scene-level Ablation. We evaluate four placement variants: (1) *w/o Grid Search* removes discrete semantic planning and grid-based placement, allowing the Director to predict continuous 3D coordinates directly; (2) *w/o Anchoring* replaces our seven anchoring types with Infinigen's original grounding mechanism; (3) *w/o Semantic Target* removes semantic surface constraints from grid search, allowing placements on semantically inappropriate surfaces; and (4) *Agent (Placement)* provides a VLM agent with our generated 3D assets, the text prompt, and the same selected coarse scene, allowing autonomous placement without our placement pipeline.

Contact Spread. For each object, we select the lowest 12% of sampled points and cluster them into four contact regions using k-means. The lowest point in each region is selected, and its vertical gap to the underlying terrain is computed via ray casting. Contact Spread is defined as the difference between the maximum and minimum gaps, capturing uneven grounding independently of terrain elevation. We report the nearest-rank 95th percentile (P95) of per-object contact spreads, pooled across all requested-add objects in the 20 scenes.

K Detailed Capability Comparison

We provide a detailed capability comparison of representative 3D generation methods in Table 8. The comparison covers three asset-level and five scene-level capabilities, highlighting InfiniDream's integration of open-ended procedural asset generation with scalable world construction.

Table 8: **Capability comparison of representative 3D generation methods.** We compare three asset-level and five scene-level capabilities. "Novel Proc. Assets" denotes the creation of procedural generators beyond predefined factories, while "Proc. Editing" includes editable procedural representations. "Placement Reasoning" refers to semantic spatial planning, whereas "Semantic Anchoring" requires semantic support-surface selection and geometry-aware contact alignment. ✓ indicates explicit support, △ partial support, and × support not established under our criteria.

	Asset-level			Scene-level				
Method	**Novel Proc. Assets**	**Seed Diversity**	**Proc. Editing**	**Text-to-Scene**	**Placement Reasoning**	**Semantic Anchoring**	**Scalable Population**	**Animation**
VIGA (Yin et al., 2026)	✓	×	✓	✓	✓	×	×	✓
LL3M (Lu et al., 2025)	✓	×	✓	×	×	×	×	×
Infinigen (Raistrick et al., 2023)	×	✓	✓	×	×	×	✓	✓
3D-GPT (Sun et al., 2023)	×	✓	✓	✓	✓	×	✓	✓
SceneX (Zhou et al., 2025)	×	✓	✓	✓	✓	△	✓	✓
Code2Worlds (Zhang et al., 2026)	×	✓	✓	✓	✓	×	✓	✓
WorldClaw (Guo et al., 2026)	×	×	×	✓	✓	✓	✓	×
WorldGen (Wang et al., 2026a)	×	×	×	✓	✓	△	✓	×
InfiniDream	✓	✓	✓	✓	✓	✓	✓	✓

L HUMAN EVALUATION DETAILS

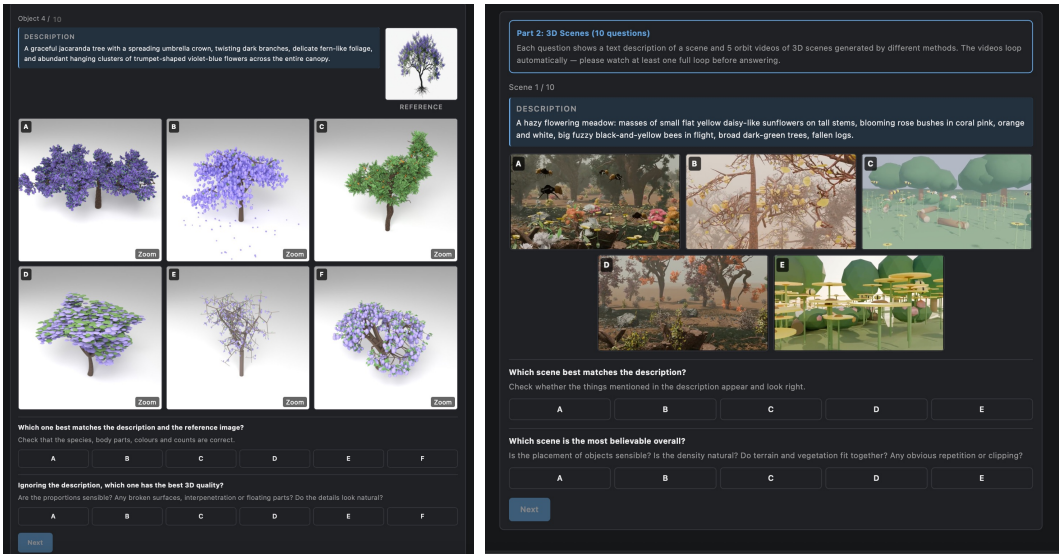

Figure 17: **Human evaluation interface.** Participants compare anonymized generation results and independently select their preferred method for semantic alignment and visual quality at both the object and scene levels.

Study Setup. We conduct a forced-choice preference study on 10 object and 10 scene generation tasks. For each object, participants are shown the text prompt, a held-out reference image not provided to any generation method, and front-view renders (1024×1024) from six methods: InfiniDream, BlenderRAG, VIGA, VLM (one-shot), VLM (harness), and Infinigen.

For each scene, participants compare 5-second orbit videos (854×480) from five methods: InfiniDream, VIGA, VLM (harness), Code2Worlds, and Infinigen. All results are presented without cropping or background normalization.

The object subset covers animals, marine life, plants, and natural props, while the scene subset spans seven distinct environment types.

Evaluation Protocol. As illustrated in Fig. 17, all methods are anonymized, and both task order and method positions are randomized independently for each participant. Each task contains two forced-choice questions evaluating semantic alignment and visual quality.

For objects, participants select the result that best matches the description and reference image, and the result with the highest 3D quality. For scenes, participants evaluate text alignment and overall scene realism, considering object placement, naturalness, and visual artifacts.

Each participant completes 40 judgments across both levels. We recruit 30 participants, yielding 20 judgments per level per participant.