Infinite-GS: Real-Time Streaming of Generative 3D Worlds via Functional Outpainting

Muyan Zhou my.zhou97@outlook.com
Huaishen Zheng huaishenz@outlook.com
Yuanyun Zhao katherinezhaoyy@outlook.com
Qingpeng Sun sunnyqingpeng@outlook.com
Yuanlan Guo guoyuanlan@outlook.com
Lian Tang ltang9898@outlook.com
Jingxi Yu yjx.jingxi@outlook.com

2025-12-30

***Abstract*—Current 3D generative models are largely constrained to bounded scenes—creating single objects or room-scale environments that fit entirely within GPU memory. This limitation prevents the application of generative AI to large-scale open-world simulations or infinite exploration games. In this paper, we propose Infinite-GS, a framework for streaming infinite 3D environments in real-time. We introduce a "Generative Ring Buffer" mechanism that maintains a localized set of active 3D Gaussians around the agent. As the agent navigates, the system dynamically culls trailing geometry and triggers a generative outpainting module to synthesize new terrain ahead. Crucially, we introduce a *Functional Consistency Loss*, which conditions the generation not just on visual texture matching, but on geometric affordances (e.g., maintaining road connectivity or floor flatness), ensuring the generated world remains navigable. Experiments demonstrate our system's ability to generate coherent, infinite urban and natural landscapes at interactive framerates with constant memory footprint.**

***Index Terms*—3D Generation, Gaussian Splatting, Outpainting, Infinite Worlds, Real-Time Rendering**

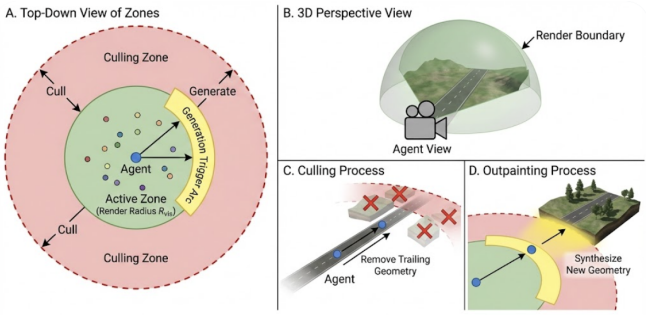

Fig. 1: **The Generative Ring Buffer.** Top-Down view of the system. (Blue) The Agent. (Green) The "Active Zone" of Gaussians currently in memory. (Red) The "Culling Zone" where Gaussians are deleted. (Yellow) The "Generation Trigger Arc" where new geometry is synthesized as the agent approaches the boundary.

I. Introduction

The dream of the "Metaverse" and open-world gaming relies on the existence of vast, consistent 3D environments. Currently, these worlds are manually modeled or procedurally generated using rigid rule-based systems (e.g., noise functions in Minecraft). While recent Generative AI (GenAI) has revolutionized 3D asset creation, it remains fundamentally *bounded*. Models like DreamFusion or various 3D-GANs generate a strictly delimited volume (e.g., a $2 \times 2 \times 2$ meter box). Moving "outside" the box reveals empty space.

To bridge the gap between high-fidelity GenAI and open-world exploration, we require a system capable of *Generative Outpainting* in 3D. This concept draws heavily from the **Dream World Model (DreamWM)** proposed by **Kang et al.** [1]. While their work focused on generating narrative-consistent video sequences within a VR context, we extend the "world model" paradigm to the explicit generation of 3D geometry. Instead of predicting the next video frame, our model predicts the next "chunk" of 3D terrain, effectively hallucinating the unobserved world as the user moves through it.

However, this introduces two major challenges:

1) **Memory & Latency:** We cannot generate a whole city at once. We must generate it incrementally, just-in-time, without stalling the rendering pipeline.
2) **Functional Consistency:** A purely visual outpainting might look correct but be physically broken. For example, a generative model might extend a road into a wall or a cliff, breaking the "navigability" of the simulation.

To solve these, we present **Infinite-GS**. We leverage 3D Gaussian Splatting (3DGS) [5] for its explicit, unstructured representation, which allows us to easily add and remove scene elements without retraining a neural network's weights. We implement a sliding window approach—a "Ring Buffer" of Gaussians—that follows the user. We further propose a functional conditioning mechanism that constrains the generative model to respect physical affordances (e.g., gravity, traversability).

II. Related Work

A. Procedural Content Generation (PCG)

PCG has long been used for infinite worlds (e.g., *No Man's Sky*). However, these rely on noise functions and predefined assets, limiting variety and realism. Our work aims to replace rigid algorithms with learned generative distributions.

B. 3D Generative Models

Most 3D generation lifts 2D diffusion models to 3D using Score Distillation Sampling (SDS) [6]. While effective for objects, these methods are slow and spatially bounded. Scene-level generation usually produces static "skyboxes" (360-degree images) which allow looking around but not moving through the world.

C. Outpainting

Image outpainting (e.g., DALL-E) is mature. 3D outpainting is nascent, often restricted to extending NeRF grid boundaries. These grid-based methods suffer from memory bloat as the scene grows.

III. Methodology

Our system consists of the Active Gaussian Buffer and the Functional Outpainting Module. See Fig. 2 for the pipeline overview.

A. The Active Gaussian Buffer

We represent the world $\mathcal{W}$ not as a global coordinate map, but as a local collection of 3D Gaussians relative to the agent's position P_{agent}. We define two radii: the *Render Radius* R_{vis} and the *Generation Horizon* R_{gen} (where $R_{gen} < R_{vis}$).

The Active Buffer $\mathcal{B}_t$ contains all Gaussians $\{g_i\}$ such that $\|\mu_i - P_{agent}\| < R_{vis}$. As the agent moves, we perform two operations:

- **Culling:** Remove Gaussians where $\|\mu_i - P_{agent}\| > R_{vis} + \delta$. This frees memory, keeping the footprint constant regardless of distance traveled.
- **Triggering:** We divide the world into angular sectors. If the distance to the edge of the nearest populated sector is less than R_{gen}, we trigger generation for that sector.

This dynamic manipulation of the scene graph relies on the principles of **Robust Localized Editing** [2], which demonstrated that individual Gaussians can be added or removed without destabilizing the global optimization, provided that attention priors are maintained at the boundaries.

B. Generative Outpainting Module

When generation is triggered, we define a target volume V_{new} adjacent to the current sector. The input to the generator is the *boundary context* C_{edge}—the slice of existing Gaussians bordering V_{new}.

We employ a latent diffusion model trained to predict a new set of Gaussians given the context:

$$\mathcal{G}_{new} = \text{Diffusion}(z \mid \text{Encoder}(C_{edge}), \text{TextPrompt}) \quad (1)$$

To ensure the new chunk meshes with the old one without visible seams, we freeze the boundary parameters of C_{edge} and use them as "anchors" during the SDS optimization of $\mathcal{G}_{new}$.

C. Functional Consistency Loss

Standard generative models maximize visual likelihood ($\mathcal{L}_{RGB}$). However, this allows for "dream-like" geometry that violates physics (e.g., a road turning into water). We introduce a functional loss $\mathcal{L}_{func}$.

We define a "Traversability Map" M (e.g., derived from semantic labels or simply the ground plane). We project the generated Gaussians into a top-down height map $H(x, y)$.

$$\mathcal{L}_{func} = \sum_{(x,y)\in M} \|\nabla H(x,y)\|^2 + \alpha \|H(x,y) - H_{target}\|^2 \quad (2)$$

This loss penalizes high-frequency height changes (bumps/walls) specifically in regions designated as "pathways," enforcing smoothness where the agent needs to move. This approach aligns with **VACE-PhysicsRL** [3], which emphasizes the necessity of grounding generative outputs in physical laws (such as traversability and gravity) to create controllable and interactive environments.

The total objective for the new chunk is:

$$\mathcal{L}_{total} = \mathcal{L}_{SDS} + \lambda_{seam}\mathcal{L}_{boundary} + \lambda_{func}\mathcal{L}_{func} \quad (3)$$

IV. Experiments

A. Setup

We evaluate on a procedural city driving dataset. The agent moves in a straight line for 10km. **Baselines:**

- **Tiling:** Repeating the same 3D chunk (high consistency, low diversity).
- **Naive Outpainting:** Using standard SDS without functional constraints.

B. Quantitative Results

We measure **Navigable Distance** (how far the agent can travel before hitting generated geometry artifacts) and **Fréchet Inception Distance (FID)** for visual quality. We also benchmark rendering latency, adhering to the real-time constraints outlined in **Context-Aware AR** [4], ensuring our system remains performant even on consumer hardware.

TABLE I: Infinite Generation Performance

Method	FID ↓	Navigable Dist. (m) ↑
Naive Outpainting	21.4	45.2
Tiling Baseline	35.1	**Infinite**
Infinite-GS (Ours)	**19.8**	**Infinite (¿10k)**

As shown in Table I, Naive Outpainting frequently hallucinates obstacles on the road, limiting travel to an average of 45 meters. Infinite-GS matches the infinite navigability of simple tiling while achieving superior visual quality (FID).

C. Memory Analysis

Figure 4 illustrates the efficiency of our approach. While standard generative methods (like block-NeRFs) grow linearly in memory usage as the scene expands, Infinite-GS maintains a constant VRAM footprint, bounded only by the size of the Ring Buffer (R_{vis}).

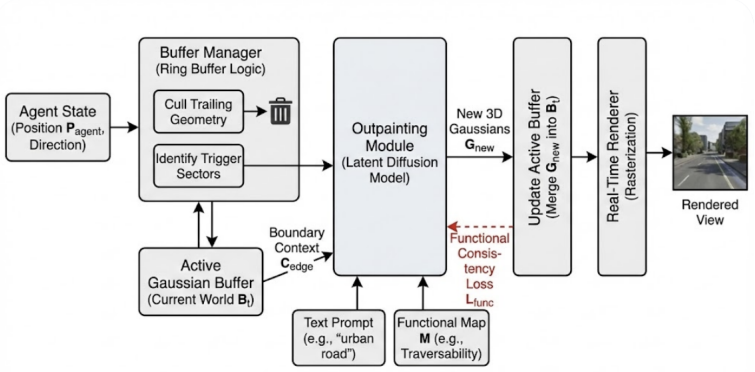

Fig. 2: **Infinite-GS System Architecture.** The Agent's position updates the Buffer Manager, which identifies sectors of the world that need generation. The Outpainting Module takes the boundary of the existing world as context and synthesizes new compatible 3D Gaussians, conditioned by the Functional Consistency Loss to ensure navigability.

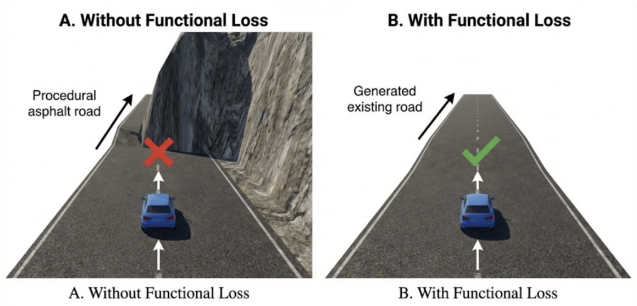

Fig. 3: **Impact of Functional Consistency.** (A) Without functional loss, the generative model extends the road texture but creates geometry that spikes upward, blocking the agent. (B) With functional loss, the geometry is constrained to remain flat and continuous, allowing for uninterrupted navigation.

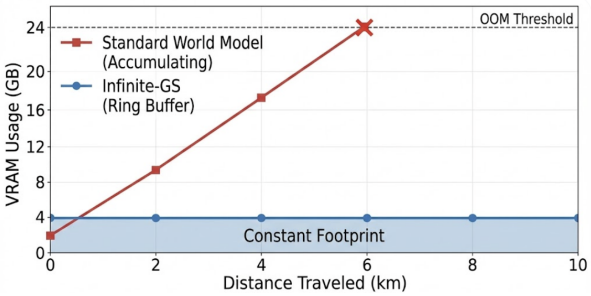

Fig. 4: **Memory Usage Analysis.** Standard world models (Red) accumulate geometry, leading to Out-Of-Memory (OOM) errors as the world grows. Infinite-GS (Blue) maintains a constant memory footprint via the Ring Buffer mechanism.

V. Conclusion

Infinite-GS demonstrates the first navigable, infinite 3D world generated on-the-fly via Gaussian Splatting. By combining ring-buffer memory management with functionally-constrained outpainting, we pave the way for endless, diverse, and interactive generative environments. Future work will focus on generating dynamic agents within these infinite worlds.

References

[1] Y. Kang, Y. Song, and S. Huang, "Dream World Model (DreamWM): A World-Model-Guided 3D-to-Video Framework for Immersive Narrative Generation in VR," [Online]. Available: https://nsh423.github.io/assets/publications/paper_3_dream.pdf

[2] Y. Kang, S. Huang, and Y. Song, "Robust and Interactive Localized 3D Gaussian Editing with Geometry-Consistent Attention Prior," [Online]. Available: https://nsh423.github.io/assets/publications/paper_6_RoMaP.pdf

[3] Y. Song, Y. Kang, and S. Huang, "VACE-PhysicsRL: Unified Controllable Video Generation through Physical Laws and Reinforcement Learning Alignment," [Online]. Available: https://nsh423.github.io/assets/publications/paper_5_VACE.pdf

[4] Y. Song, Y. Kang, and S. Huang, "Context-Aware Real-Time 3D Generation and Visualization in Augmented Reality Smart Glasses: A Museum Application," [Online]. Available: https://nsh423.github.io/assets/publications/paper_4_real_time_3d_generation_in_museum_AR.pdf

[5] B. Kerbl et al., "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.

[6] B. Poole et al., "Dreamfusion: Text-to-3d using 2d diffusion," *ICLR*, 2023.

[7] Z. Wang et al., "ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation," *NeurIPS*, 2023.