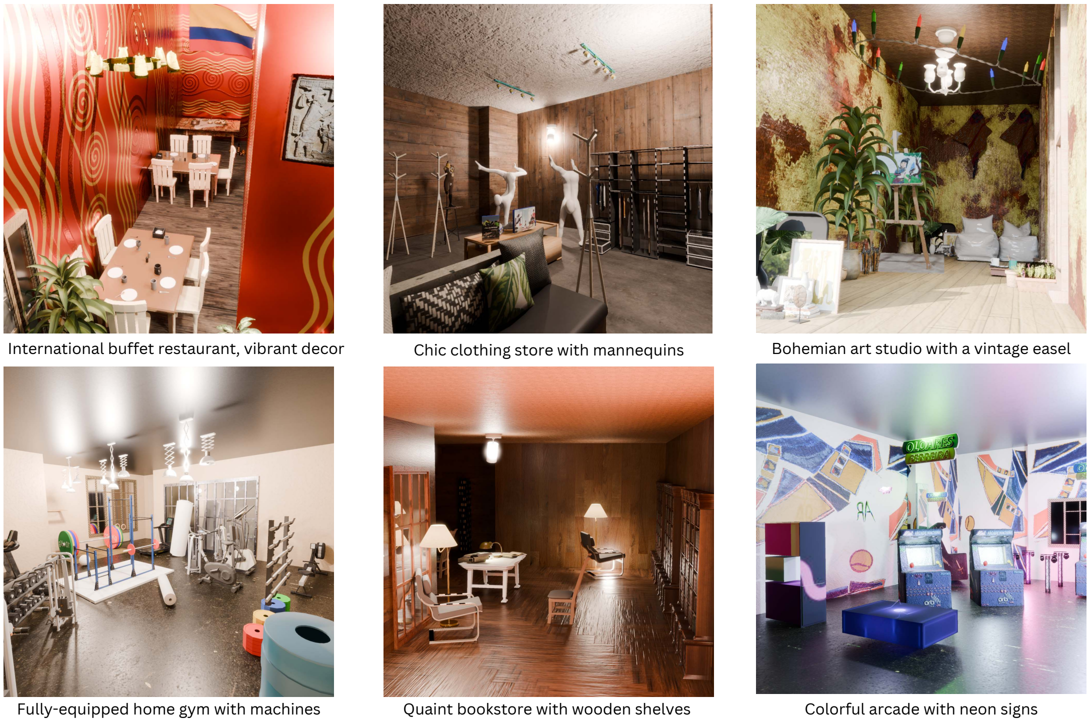
3D-GENERALIST: Vision-Language-Action Models for Crafting 3D Worlds

Fan-Yun Sun[1], Shengguang Wu[1], Christian Jacobsen[2], Thomas Yim[1], Haoming Zou[1], Alex Zook[2], Shangru Li[2], Yu-Hsin Chou, Ethem Can[2], Xunlei Wu[2], Clemens Eppner[2], Valts Blukis[2], Jonathan Tremblay[2], Jiajun Wu[1], Stan Birchfield[2†], Nick Haber[1†]

[1]Stanford University, [2]NVIDIA

https://ai.stanford.edu/~sunfanyun/3d-generalist/


International buffet restaurant, vibrant decor

Chic clothing store with mannequins

Bohemian art studio with a vintage easel

Fully-equipped home gym with machines

Quaint bookstore with wooden shelves

Colorful arcade with neon signs

Figure 1. Sample 3D environments generated by 3D-GENERALIST, demonstrating control over assets, layout, material, and lighting.

Abstract

Creating 3D graphics content for immersive and interactive worlds remains labor-intensive, limiting our ability to create large-scale synthetic data for training foundation models. Recent methods aim to alleviate this, but they often focus on a single aspect (e.g., layout) and do not improve generation quality by simply scaling computational resources. We recast 3D environment generation as a sequential decision-making problem, using Vision-Language Models (VLMs) as policies that output actions to jointly craft a 3D environment's layout, materials, lighting, and assets. Our framework, ***3D-Generalist****, trains VLMs to generate more prompt-aligned 3D environments via self-improvement fine-tuning. We demonstrate the effectiveness of* ***3D-Generalist*** *and our training strategy in generating simulation-ready 3D environments. We also demonstrate its quality and scalability for synthetic data generation by pretraining a vision foundation model on the generated data. After fine-tuning on downstream tasks, we show that it surpasses models pre-trained on meticulously human-crafted synthetic data and approaches results achieved when training with orders of magnitude larger real data.*

1. Introduction

3D graphics design is vital in gaming, augmented reality, virtual reality, and robotics simulation. However, creating immersive 3D worlds remains labor-intensive: artists and designers must select and create assets, apply materials, set up lights, and arrange these elements into a coherent environment. In this paper, we study scalable generation of *Simulation-Ready* 3D environments (collision-free scenes composed of 3D meshes) for downstream synthetic data applications or robotics task simulation, rather than implicit representations such as NeRF or Gaussian Splats.

Diffusion and large transformer models have opened new opportunities to accelerate or automate parts of the traditional graphics workflow. For example, Holodeck [51] and RoboCasa [28] use large language models (LLMs) to select, retrieve, and place 3D objects from asset libraries like Objaverse [12], while URDFormer [8] uses diffusion models to synthesize textures. However, these works scale poorly with compute, leaving a gap between their outputs and the quality demanded by downstream tasks. We aim to build a system that takes open-ended text prompts and outputs 3D environments while benefiting from additional computation.

To scale with compute, we use the ability of models to self-correct and improve upon their own outputs [23, 24]. This approach mirrors the iterative refinement process of expert 3D artists, who render, inspect, identify mistakes, and refine with added detail and corrections. Guided by this insight, we introduce 3D-GENERALIST, a framework that unifies materials, lighting, assets, and layout by casting 3D environment generation as a sequential decision-making process. We use large multimodal models as *action policies* guided by observations of the current 3D world. 3D-GENERALIST comprises three modules: *Panoramic Environment Generation*, *Scene-Level Policy*, and *Asset-Level Policy*. Our key contributions include:

- *Panoramic Environment Generation*, which generates architectural layouts (i.e., floorplans and door/window placements) from text via a panoramic image-guided inverse graphics procedure.
- *Scene-Level Policy*, a VLM policy that jointly refines 3D layouts, materials, assets, and lighting. Its iterative self-improvement enables self-correction, allowing *Scene-Level Policy* to outperform SOTA baselines in prompt alignment while boosting general-domain visual grounding.
- *Asset-Level Policy*, a VLM policy that iteratively places assets on top of other *unlabeled* assets in a semantically aligned and physically plausible way.
- Showing 3D-GENERALIST's efficacy by training a ViT that surpasses counterparts trained on curated 3D data, rivaling performance from large real-world datasets.

2. Related Work

2.1. Prompt-driven 3D Scene Generation

Earlier forays into prompt-driven indoor scene synthesis relied on manual mappings between language and object placements. Methods such as Wordseye [10] and its follow-ups are symbolic and rule-based, requiring substantial manual effort to generalize to new domains and constraints [5, 6, 27, 30, 38]. Recent advances have explored two main directions. One line of work uses diffusion priors with representations such as neural radiance fields or Gaussian splats [15, 31, 37, 55, 56]. These scenes lack separable, manipulable objects and surfaces, making them unsuitable for synthetic data applications that require precise instance-level annotations or robotics applications that require simulated robot-object interactions. Another line of research focuses on generating *simulation-ready* scenes, often using intermediate representations (e.g., scene graphs or layouts) to arrange 3D assets from a repository [2, 4, 17, 22, 25, 33, 51–53]. Many focus on learning distributions over assets and layouts from datasets [18, 29, 43]. More recently, Large Language Models (LLMs) and Vision-Language Models (VLMs) have enabled open-vocabulary 3D scene synthesis without dependence on predefined labels or categories. For example, LayoutGPT [17] prompts LLMs to directly generate 3D layouts for indoor scenes. LayoutVLM [40] uses VLMs to generate 3D layouts by generating objective functions that can be differentiably optimized. Unlike prior work that mostly focuses on layout, 3D-GENERALIST jointly crafts layouts, materials, fixtures, and lighting.

2.2. Vision-Language Models for 3D Reasoning

While VLMs demonstrate strong general-domain image understanding, they struggle with 3D and spatial relations [7, 21, 45]. Many works enhance VLM spatial reasoning with visual markers, which anchor specific visual reference points in images [49]. These markers reduce reliance on textual cues alone and allow VLMs to interpret spatial layouts and visual hierarchies in images more directly. Others have curated datasets for training spatial reasoning [19]. For example, SpatialVLM [7] constructed a spatial reasoning dataset, enabling VLMs to perform tasks involving complex object placements and spatial configurations. SpatialRGPT [9] further strengthens VLM spatial understanding by leveraging multi-view inputs to improve three-dimensional reasoning and object alignment across perspectives. Similarly, 3D-GENERALIST uses multi-view images with VLMs to perform spatial reasoning tasks and fine-tunes the VLM on task-specific data.

3. 3D-GENERALIST

3D-GENERALIST takes a text prompt as input and outputs a 3D room with materials, fixtures (e.g., doors and windows),

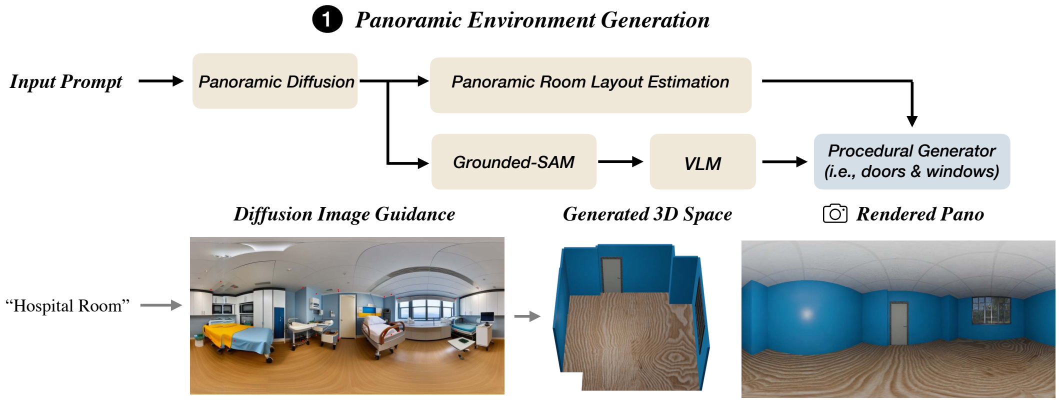


Figure 2. **Overview of 3D-Generalist's *Panoramic Environment Generation*.** We use panoramic diffusion to generate a guiding 360° scene image, then extract the corners, windows, and doors information using a room layout estimation model, Grounded-SAM, and a VLM, respectively. These predictions are then used to construct the 3D room with fixtures procedurally.

3D assets, and lighting. We treat detailed, prompt-aligned 3D environment creation as a *sequential decision-making* problem and iteratively refine scenes with our scene-level and asset-level policies. 3D-Generalist uses Vision-Language Models (VLMs) as policies to optimize 3D environments iteratively. Below, we outline the workflow and detail each of the three modules in 3D-Generalist.

3.1. Panoramic Environment Generation

This module, illustrated in Figure 2, initializes a base 3D room from the input text prompt, including walls, floors, and fixtures such as doors and windows. LLMs often struggle to predict room coordinates and door/window locations, resulting in overly simplistic or unrealistic rooms. We therefore first use a panoramic diffusion model [16] to generate a 360° image as guidance, then use an inverse graphics procedure to construct the 3D environment as follows:

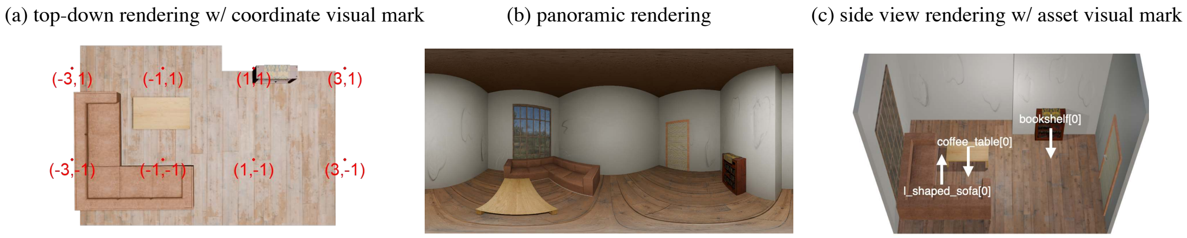
1. **Room Layout Estimation.** We take the panoramic image and use the HorizonNet [39] model to derive the basic room structure (e.g., walls, floors, ceiling).
2. **Fixture Segmentation.** We apply Grounded SAM [34] to segment windows and doors.
3. **VLM Annotation.** A Vision-Language Model (i.e., GPT-4o [1]) inspects each segmented region to determine its type (e.g., *single door*, *double door*, *sliding door*, or *folding door*) and materials (e.g., *door frame material*, *door material*, and *door knob material*).
4. **Procedural Generation.** We then procedurally construct rooms, doors, and windows at the corresponding 3D locations, as in [11].

3.2. Scene-Level Policy

This module adds assets to the initial 3D room generated by *Panoramic Environment Generation*. Models that generate physically based rendering (PBR) materials or 3D assets from text or images often produce outputs with significant artifacts. We therefore use a large repository of assets and materials with image-conditioned retrieval from diffusion-generated images. Our key insight is to build a model capable of **self-correction**. *Scene-Level Policy* employs a VLM as the policy model, taking the current state of the 3D environment as input and outputting action in code to modify the 3D environment. Figure 4 illustrates this process. Below, we describe the key components.

We use a domain-specific language (DSL) designed to represent 3D environments flexibly using a combination of code and natural language. Our Scene DSL defines key descriptors for scene elements such as floors, walls, ceilings, objects, and lighting. The *Category* descriptor specifies the type of element ({`floors, walls, ceilings, objects`}). The *Placement* descriptor encodes spatial attributes, including position $(x, y, z) \in \mathbb{R}^3$, rotation $\theta_z \in [0, 2\pi]$, and scale $s \in \mathbb{R}^3$, allowing multiple placements per element. The *Material* descriptor provides a natural language description of surface properties. The *Lighting* descriptor defines the type $t \in \{\text{point}, \text{directional}, \text{area}\}$, intensity $i \in \mathbb{R}^+$, and color $c = (r, g, b)$, where $r, g, b \in [0, 1]$ are normalized RGB values. More details are in the supplementary material.

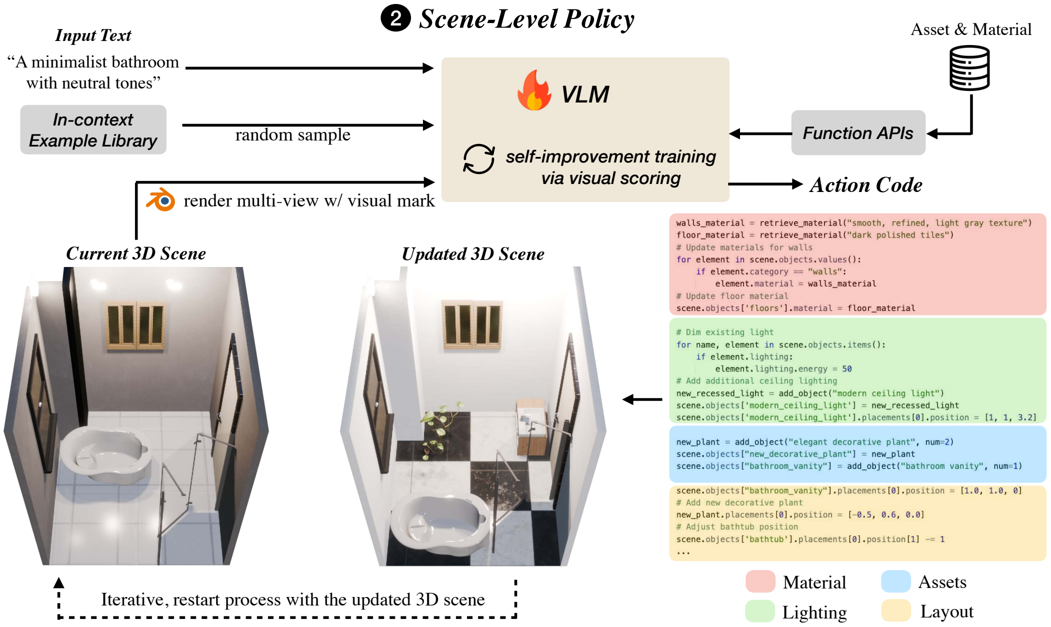
Vision-Language-Action Model Let S_0 represent the 3D environment we obtained from *Panoramic Environment Generation*, S_t be the 3D environment at iteration t, and let $\mathcal{P}$ denote the input text prompt. Our VLM, π_θ, parameterized by θ, takes multi-view images $\mathcal{I}_t$ of S_t (as shown in Figure 3) along with $\mathcal{P}$ as input and outputs an action code $a_t \sim \pi_\theta\big(a \mid \mathcal{I}_t, \mathcal{P}\big)$. We execute this action code using exposed tools and function APIs—such as those for retrieving assets or materials based on natural language—which updates the 3D environment, expressed as $S_{t+1} = f\big(S_t, a_t\big)$. Following the update, we render new images $\mathcal{I}_{t+1}$ from S_{t+1} to serve as the next state


(a) top-down rendering w/ coordinate visual mark

(b) panoramic rendering

(c) side view rendering w/ asset visual mark

Figure 3. **Overview of our Multi-View Representation $\mathcal{I}_t$.** This illustrates how we render a 3D environment and feed it as input to our VLM policy in *Scene-Level Policy*. The first view is overlaid with visual marks of the x-y coordinate system of the current environment, the second view is a panoramic render of the room, and the third is overlaid with the variable names for the existing asset instances.


2 Scene-Level Policy
Input Text
"A minimalist bathroom with neutral tones"
In-context Example Library
random sample
render multi-view w/ visual mark
VLM
self-improvement training via visual scoring
Asset & Material
Function APIs
Action Code
Current 3D Scene
Updated 3D Scene
Iterative, restart process with the updated 3D scene
🟥 Material
🟦 Assets
🟩 Lighting
🟨 Layout

```
walls_material = retrieve_material("smooth, refined, light gray texture")
floor_material = retrieve_material("dark polished tiles")
# Update materials for walls
for element in scene.objects.values():
    if element.category == "walls":
        element.material = walls_material
# Update floor material
scene.objects['floors'].material = floor_material
```

```
# Dim existing light
for name, element in scene.objects.items():
    if element.lighting:
        element.lighting.energy = 50
# Add additional ceiling lighting
new_recessed_light = add_object("modern ceiling light")
scene.objects['modern_ceiling_light'] = new_recessed_light
scene.objects['modern_ceiling_light'].placements[0].position = [1, 1, 3.2]
```

```
new_plant = add_object("elegant decorative plant", num=2)
scene.objects["new_decorative_plant"] = new_plant
scene.objects["bathroom_vanity"] = add_object("bathroom vanity", num=1)
```

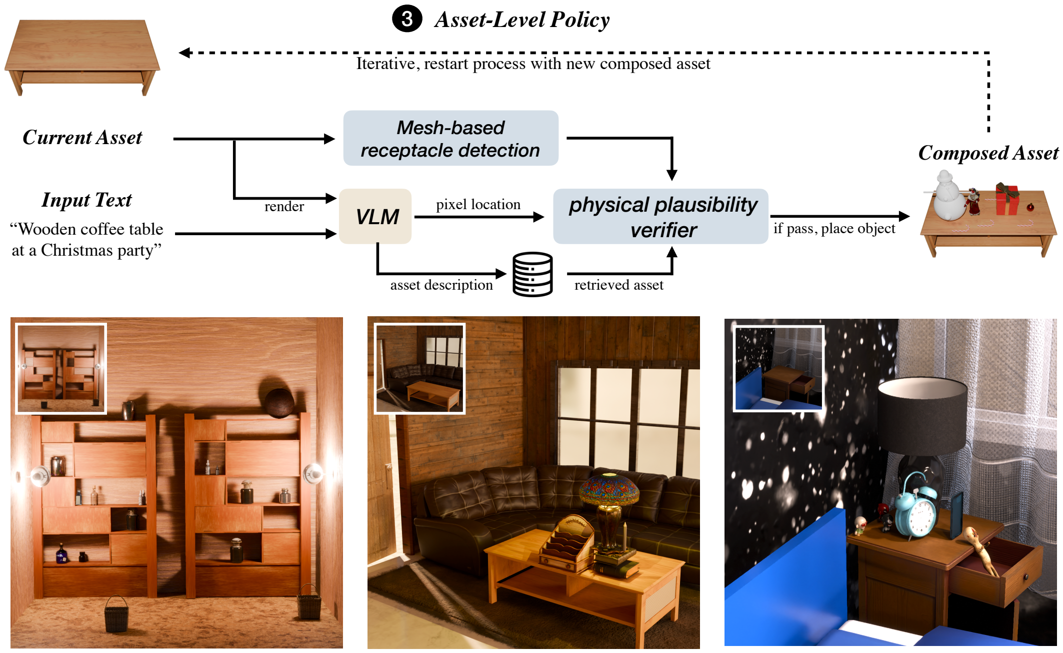
```
scene.objects["bathroom_vanity"].placements[0].position = [1.0, 1.0, 0]
# Add new decorative plant
new_plant.placements[0].position = [-0.5, 0.6, 0.0]
# Adjust bathtub position
scene.objects['bathtub'].placements[0].position[1] -= 1
...
```

Figure 4. **Overview of 3D-Generalist's *Scene-Level Policy*.** Starting with the current 3D scene, we render multiple views and combine the text prompt with an in-context example to guide a vision-language model (VLM) acting as an action policy. In each round, the VLM generates a program that updates assets, materials, layout, and lighting. We use a self-improvement training strategy to fine-tune the VLM to take actions that could lead to more prompt-aligned 3D scenes.

representation. In Figure 4's example, the action code adjusts the position of the bathtub, adds a bathroom vanity and two plants, and increases the scene's lighting.

Self-Improvement Fine-tuning Off-the-shelf VLMs often struggle to consistently generate action code that improves the resulting 3D environment. To address this, we use a self-improvement fine-tuning strategy. Let $\pi_\theta^{(i)}$ denote the VLM policy at the start of the i-th fine-tuning round. Each round begins by generating a set of tasks (i.e., text prompts). For each task, we initialize the 3D environment using *Panoramic Environment Generation*, and $\pi_\theta^{(i)}$ iteratively updates the environment. At each step, we generate multiple candidate actions and retain the top-scoring action sequences—those that yield the highest CLIP scores between environment renderings and the input prompt. We then update the policy via supervised fine-tuning, resulting in an improved policy $\pi_\theta^{(i+1)}$. This completes one self-improvement round, after which $\pi_\theta^{(i+1)}$ can generate new tasks and environments for further refinement.

We introduce an *in-context library*, a collection of action code examples that significantly enhances CLIP alignment scores. Empirically, we find that leveraging the *in-context library* during data generation substantially improves performance by promoting diversity and increasing the likelihood of discovering effective action sequences. We use GPT-4o [1] as the base model and its publicly available fine-tuning API in our experiments. More details can be found in the supplementary material.


3 Asset-Level Policy
Iterative, restart process with new composed asset
Current Asset
Input Text
“Wooden coffee table at a Christmas party”
render
Mesh-based receptacle detection
VLM
pixel location
physical plausibility verifier
if pass, place object
Composed Asset
asset description
retrieved asset

“Shelves in a walk-in pantry” “Coffee table in a vintage living room” “Nightstand in a dreamy children’s room”

Figure 5. **Overview of 3D-GENERALIST’s *Asset-Level Policy*.** We qualitatively showcase its capability to handle diverse placement tasks, such as placing assets between shelves (i.e., as shown in the leftmost example), stacking assets on top of one another (i.e., the books and lamps as shown in the middle example), and placing assets in the open shelf of a nightstand (i.e., the figurine in the rightmost example).

3.3. Asset-Level Policy

Our scene-level VLM policy in *Scene-Level Policy* often omits smaller objects (e.g., books, plates, utensils) and focuses on larger, defining assets. In contrast, placing small assets requires a different approach to ensure physical plausibility.

To address this, we introduce *Asset-Level Policy*, which refines the environment by composing smaller assets with “receptacle objects” (e.g., shelves, tables, counters), closely mimicking the construction process of physically grounded environments. *Asset-Level Policy* is equipped with physical plausibility verifiers applied prior to placement. The system is inherently extensible, supporting custom plug-in verifiers such as physics engines or logic-based constraints.

The process begins with GPT-4o determining whether a 3D asset qualifies as a “receptacle object” that can host smaller items. Once identified as a receptacle, we initiate an iterative process to place these smaller assets. Before the first round, we have a base object $\mathcal{O}$ and a text prompt $\mathcal{P}$ (“wooden office desk in a writer’s home”). The process for the k-th round:

Mesh-based Placeable Surface Detection. We use a mesh-based surface detection approach to find valid surfaces on the receptacle object (or on previously placed items, allowing for stacking); see supplementary for details. We then render the receptacle object from a randomly sampled angle on a hemisphere. We sample a new camera angle each round to expose different placeable surfaces across iterations.

VLM as a Placement Policy. The rendered image $\mathcal{I}'_k$ (distinct from the multi-view renderings $\mathcal{I}_k$ in *Scene-Level Policy*) and $\mathcal{P}$ are fed into a separate policy model π'_ϕ, a pretrained VLM from [13]. We choose [13] because it excels at the pixel-level precision required for asset placement, in contrast to GPT-4o in *Scene-Level Policy*, which requires stronger semantic reasoning. Formally, the action chosen in round k can be denoted as $a'_k \sim \pi'_\phi\big(a \mid \mathcal{I}'_k,\ \mathcal{P}\big)$. Here, an action is represented as $a'_k = (o'_k, p'_k)$, where o'_k denotes the pixel location in the rendered image indicating the placement position, and p'_k represents the text description used to retrieve 3D assets from the repository.

3D Placement and Environment Update. Using the pixel location o'_k generated by [13] and associated camera parameters, we generate a 3D ray to identify a precise placement point on the mesh. If the location intersects one of the previously identified valid surfaces, we retrieve the asset specified by p'_k. After verifying that this asset can be realistically placed (via mesh collision checks), we place it and repeat this process with the newly composed asset. By iterating these steps, *Asset-Level Policy* recursively

Table 1. Physical plausibility and semantic alignment, using the same metrics and setting as Tab. 2 of LayoutVLM [40]. All scores range from 0 to 100 (↑ is better). Due to the iterative nature, the model can self-correct asset placement over rounds of iteration, improving physical plausibility.

	Physics		Semantics		Overall Score
	CF	IB	Pos.	Rot.	PSA
LayoutGPT	83.8	24.2	**80.8**	78.0	16.6
Holodeck	77.8	8.1	62.8	55.6	5.6
I-Design	76.8	34.3	68.3	62.8	18.0
LayoutVLM	81.8	94.9	77.5	73.2	58.8
3D-Generalist	**99.0**	**98.0**	78.2	**79.1**	**67.9**

adds fine details—often overlooked by scene-level policies—into the 3D environment; for example, a book can be placed on the table, then a pen can be placed on the book.

4. Experiments

We evaluated 3D-Generalist to answer the following questions:

(a) How does 3D-Generalist compare to existing methods on generating simulation-ready 3D environments?
(b) Does our self-improvement fine-tuning strategy enable *Scene-Level Policy* to iteratively refine 3D environments to be more prompt-aligned?
(c) Can *Asset-Level Policy* effectively place small objects in a semantically coherent manner?
(d) Does our method enable effective scaling of 3D data generation for training robust visual feature extractors?

4.1. Simulation-Ready 3D Environment Generation

Following existing works [20, 32, 40, 54, 56], we evaluated simulation-ready 3D environments by measuring physical plausibility and semantic coherence. We measured physical plausibility using the *Collision-Free Score (CF)* and *In-Boundary Score (IB)*. We enforced all assets to be placed, with remaining assets randomly placed if a method failed. We assessed semantic coherence using *Positional Coherency (Pos.)* and *Rotational Coherency (Rot.)*, measuring alignment with the input prompt. To evaluate semantic coherence across layouts without ground truth, we used *GPT-4o* to score layouts based on top-down and side-view renderings and the language instructions. We computed the *Physically-Grounded Semantic Alignment Score (PSA)* as the GPT-4o rating weighted by physical plausibility. Scores range from 0 to 100, with higher scores indicating better performance. We did not enable gravity in all experiments, and we considered a collision to occur when the intersecting area between two meshes exceeded 0.1 m^2 in Blender.

Comparative Study To answer (a), we chose LayoutGPT [17], Holodeck [51], and LayoutVLM [40], the state-of-the-art *sim-ready* 3D environment generation methods, as baselines. While there are many other works on 3D scene generation like Physcene [50] and DiffuScene [43], they are either limited to a fixed set of object categories (i.e., not *open-vocabulary* methods) or rely on representations such as Gaussian splats or NeRFs rather than using reasoning models (i.e., LLMs, VLMs) with an object-centric representation. 3D-Generalist outperformed all baselines using the same set of prompts used in Holodeck [51] (over 250 prompts across 50 room types). 3D-Generalist took 1-3 minutes to generate a single scene on A100s when starting from the initial prompt.

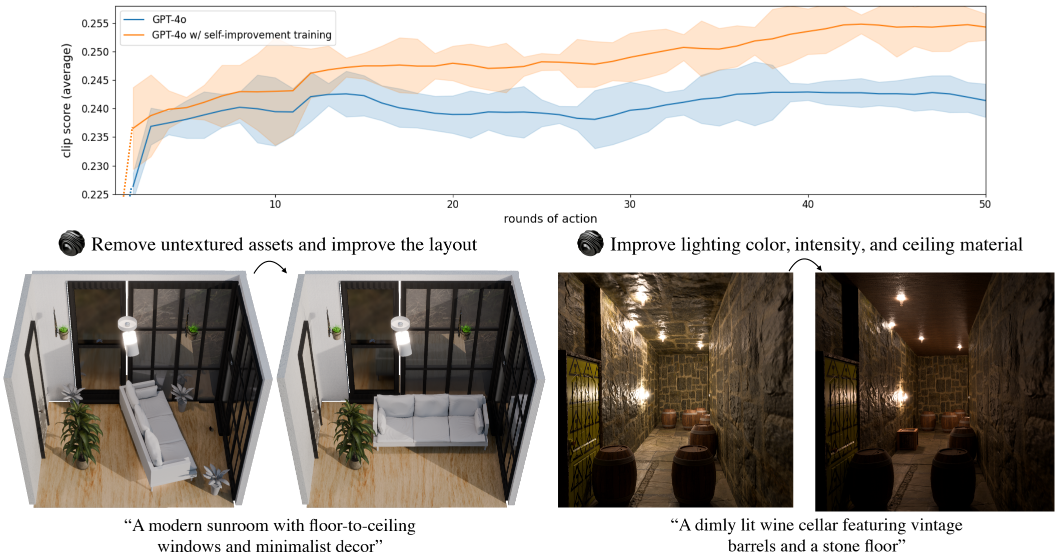
Table 2. Comparative Analysis and Ablation Study on 3D-Generalist's *Scene-Level Policy*. "# Training Runs" refer to the total number of times we perform (self-improvement) fine-tuning on the VLM, and "# Actions" refer to the number of actions taken by the VLM in *Scene-Level Policy* per scene generation. We report the CLIP score of the final generated scene.

Method	**# Training Runs**	**# Actions**	**CLIP Score**
Random noise (lower bound)	-	-	0.026
LayoutGPT	-	-	0.228
Holodeck	-	-	0.231
LayoutVLM	-	-	0.239
3D-Generalist	3	10	**0.275**
3D-Generalist w/o fine-tuning	0	10	0.252
3D-Generalist w/o in-context library	0	10	0.237
3D-Generalist ablations	3	3	0.254
	2	3	0.251
	1	3	0.248
	0	3	0.242
	0	0	0.159

Scene-Level Policy Evaluation To answer (b), we assessed our self-improving fine-tuning strategy using quantitative metrics and qualitative examples (CLIP scores in Table 2 and Figure 6). The x-axis in Figure 6 represents "rounds of action" within a generation sequence. While GPT-4o (baseline) showed some early refinement, our fine-tuned model demonstrated significantly stronger iterative improvement. Qualitative examples highlight *Scene-Level Policy*'s self-correction.

Additional Ablation on Fine-tuning We explored how self-improvement fine-tuning impacted VLM performance in general domains. Since our approach trained the VLM to generate better action code from image observations, we hypothesized that it enhanced visually grounded understanding transferable to general-domain imagery. To verify this, we evaluated 3D-Generalist's *Scene-Level Policy* on two commonly used visual hallucination benchmarks (i.e., **Object HalBench** [36] and **AMBER** [44]) that directly assess the accuracy of the VLM's visual grounding capabilities (details in supplementary).

Results in Table 3 show that the base VLM GPT-4o benefited significantly from our self-improvement fine-tuning, with the fine-tuned 3D-Generalist exhibiting lower hal-



“A modern sunroom with floor-to-ceiling windows and minimalist decor”

“A dimly lit wine cellar featuring vintage barrels and a stone floor”

Figure 6. **Evaluation of the self-improving fine-tuning strategy.** The top graph shows averaged results, indicating that GPT-4o improves 3D prompt alignment iteratively only after self-improvement fine-tuning. The bottom examples highlight 3D-GENERALIST's self-correction: refining untextured assets and layout in one case, and adjusting lighting and ceiling material in another.

lucination rates across both benchmarks and multiple metric levels, except for a slight decrease in object coverage (COVER) on AMBER. These results demonstrate that, in addition to generating 3D scenes more aligned with the prompt, our self-improvement fine-tuning also benefited VLMs' general-domain visual understanding by reducing visual hallucinations.

Asset-Level Policy Evaluation In our experiments, GPT-4o was responsible for determining whether a 3D asset qualified as a receptacle object, while another VLM, Molmo-7B [13], provided visual cues for pixel-level placement. As shown in Figure 5's qualitative examples, *Asset-Level Policy* placed books on shelves, stacked plates on tables, and filled open shelves with suitable items.

To answer **(c)** quantitatively, we ran the experiment on 15 “receptacle assets” split equally into three categories: small tables (nightstand, student desk, bar, side table, foyer table), large tables (coffee table, glass coffee table, kitchen countertop, metal table, office desk), and multi-level assets (bookshelf - open back, bookshelf - closed back, pantry shelf, work shelf, warehouse shelf). After placement, we rendered from four equally spaced camera positions around the object and reported the maximum CLIP similarity score between the rendered image and the input text prompt. For *Asset-Level Policy*, we set the maximum number of successful asset placements to 10.

Across the 15 test cases, the base receptacle objects (without any other objects) had an average CLIP score of 0.264. The baseline method produced composed assets with an average CLIP score of 0.269, and our *Asset-Level Policy* produced an average score of **0.281** , demonstrating the ability to place assets on “receptacle assets” in a prompt-aligned and physically plausible way [1] Qualitatively, we observed that the baseline inpainting-based method regularly generated many overlapping objects and could not place an item behind another. Our method, however, had access to multiple views during the iterative process and could spread out object placement more naturally. Refer to Figure 7 and the supplementary material for qualitative comparisons that showcase the importance of leveraging semantic knowledge in the VLM (i.e., Molmo) to predict asset placement.

Table 3. Our self-improvement fine-tuning strategy demonstrates positive transfer that effectively reduces VLM's visual hallucinations in general-domain images.

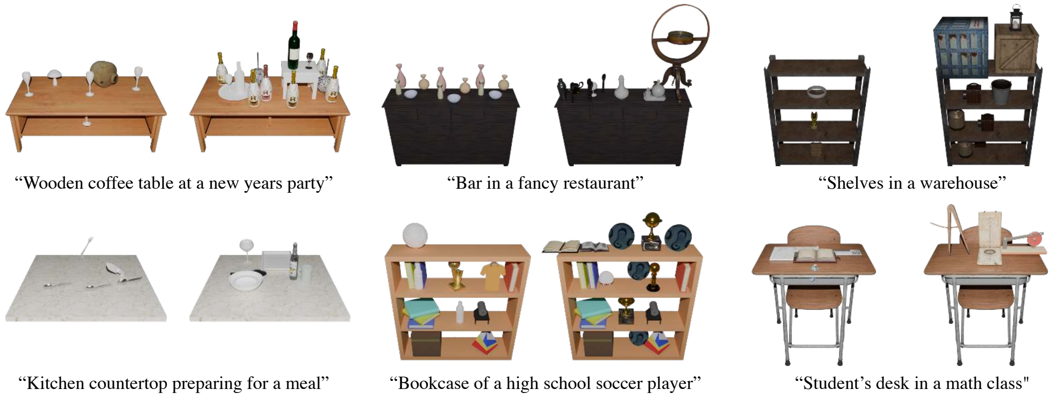
	Object HalBench		**AMBER-Generative**			
Method	CHAIRs↓	CHAIRi↓	CHAIR↓	COVER↑	HAL↓	COG↓
3D-GENERALIST (GPT-4o)	10.3	5.4	3.3	**61.8**	16.5	0.8
3D-GENERALIST (GPT-4o **finetuned**)	**7.7**	**4.6**	**3.2**	60.8	**15.7**	**0.7**

4.2. Downstream Applications

To demonstrate 3D-GENERALIST's capability in scaling up 3D data, we conducted large-scale pretraining of vision encoders on generated synthetic data. We rendered millions of images from generated scenes using Omniverse to create a

[1] During our evaluation process, an asset can only be placed if the resulting scene is physically plausible (i.e., no mesh collision).

[1] The training GPU hours for the original Florence-v2 is estimated.



"Wooden coffee table at a new years party"

"Bar in a fancy restaurant"

"Shelves in a warehouse"

"Kitchen countertop preparing for a meal"

"Bookcase of a high school soccer player"

"Student's desk in a math class"

Figure 7. Comparison between the baseline inpainting method (left) and 3D-GENERALIST's *Asset-Level Policy* (right).

Table 4. Downstream application: Pretraining Florence-v2 on generated datasets and reporting fine-tuned ImageNet performance.

Pretraining Dataset	# Labels	ImageNet-1K Top 1 ↑
Hypersim	861,080	0.727
3D-GENERALIST	861,080	0.731
3D-GENERALIST	12,175,588	0.776
Florence 2 (real)	5,000,000,000	0.786

large-scale dataset and compared feature extractors trained on our synthetic data with those trained on HyperSim [35], an artist-crafted dataset, along with a real-world dataset.

In our experiments, we first trained a vision encoder using the Florence 2 [47] framework with a DaViT-B [47] backbone. Next, we initialized an image classification model from the pretrained checkpoint and fine-tuned the model on ImageNet1K [14]. The only variable in our experiments was the selection of the pretraining dataset. We converted Hypersim into Florence 2 format with three tasks: 2D object detection, semantic segmentation, and bounding box to segmentation. Doing so resulted in just over 860K labels, and we also generated a dataset consisting of the same number of labels using 3D-GENERALIST. Finally, since 3D-GENERALIST can scale datasets with ease, we generated an additional 11.5M labels for pretraining.

All results are provided in Table 4 and compared to fine-tuning on the publicly available DaViT-B weights from Florence 2, which was trained on 5B real-world labels. All experiments were conducted on a 96×H100s cluster, with the runs below 1 million labels taking roughly 96 hours and the larger run of 12 million labels taking 200 hours. With commonsense and visual spatial reasoning from VLMs, 3D-GENERALIST produced high-fidelity scenes that better reflected real-world object distributions and layouts. This was associated with more effective transfer on real-world downstream tasks. Furthermore, our synthetically generated environments scaled up training data for vision models, achieving comparable or superior performance to manually curated datasets such as Hypersim while reducing data collection effort.

5. Conclusion

We introduce **3D-Generalist**, a framework that formulates text-driven 3D world creation as a sequential decision-making problem. The framework can be integrated with image-to-3D reconstruction [42, 48], image-to-PBR material generation [26] or 3D asset generation models [3], enabling a fully generative graphics pipeline. Beyond content creation, the use of explicit 3D representations also facilitates generative simulation—enabling the automatic generation of environments for training RL policies [41, 46].

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 4

[2] 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. 2

[3] Maciej Bala, Yin Cui, Yifan Ding, Yunhao Ge, Zekun Hao, Jon Hasselgren, Jacob Huffman, Jingyi Jin, JP Lewis, Zhaoshuo Li, et al. Edify 3d: Scalable high-quality 3d asset generation. *arXiv preprint arXiv:2411.07135*, 2024. 8

[4] Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-design: Personalized llm interior designer. *arXiv preprint arXiv:2404.02838*, 2024. 2

[5] Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 2028–2038, 2014. 2

[6] Angel X Chang, Mihail Eric, Manolis Savva, and Christopher D Manning. Sceneseer: 3d scene design with natural language. *arXiv preprint arXiv:1703.00050*, 2017. 2

[7] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14455–14465, 2024. 2

[8] Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024. 2

[9] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision language model. *arXiv preprint arXiv:2406.01584*, 2024. 2

[10] Bob Coyne and Richard Sproat. Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, 2001. 2

[11] 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. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 3

[12] 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*, pages 13142–13153, 2023. 2

[13] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024. 5, 7

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 8

[15] Dave Epstein, Ben Poole, Ben Mildenhall, Alexei A Efros, and Aleksander Holynski. Disentangled 3D scene generation with layout learning. *arXiv preprint arXiv:2402.16936*, 2024. 2

[16] Mengyang Feng, Jinlin Liu, Miaomiao Cui, and Xuansong Xie. Diffusion360: Seamless 360 degree panoramic image generation based on diffusion models. *arXiv preprint arXiv:2311.13141*, 2023. 3

[17] 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. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023. 2, 6

[18] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012. 2

[19] Ankit Goyal, Kaiyu Yang, Dawei Yang, and Jia Deng. Rel3D: A minimally contrastive benchmark for grounding spatial relations in 3d. *Advances in Neural Information Processing Systems*, 33:10514–10525, 2020. 2

[20] 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*, pages 7909–7920, 2023. 6

[21] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023. 2

[22] 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 *Forty-first International Conference on Machine Learning*, 2024. 2

[23] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023. 2

[24] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024. 2

[25] Chenguo Lin and Yadong Mu. InstructScene: Instruction-driven 3D indoor scene synthesis with semantic graph prior. *arXiv preprint arXiv:2402.04717*, 2024. 2

[26] Ivan Lopes, Fabio Pizzati, and Raoul de Charette. Material palette: Extraction of materials from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4379–4388, 2024. 8

[27] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018. 2

[28] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024. 2

[29] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021. 2

[30] Akshay Gadi Patil, Supriya Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. Advances in data-driven analysis and synthesis of 3d indoor scenes. In *Computer Graphics Forum*, page e14927. Wiley Online Library, 2024. 2

[31] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion. In *2024 International Conference on 3D Vision (3DV)*, pages 651–663. IEEE, 2024. 2

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 6

[33] Ohad Rahamim, Hilit Segev, Idan Achituve, Yuval Atzmon, Yoni Kasten, and Gal Chechik. Lay-a-scene: Personalized 3d object arrangement using text-to-image priors. *arXiv preprint arXiv:2406.00687*, 2024. 2

[34] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 3

[35] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV) 2021*, 2021. 8

[36] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. *arXiv preprint arXiv:1809.02156*, 2018. 6

[37] Jonas Schult, Sam Tsai, Lukas Höllein, Bichen Wu, Jialiang Wang, Chih-Yao Ma, Kunpeng Li, Xiaofang Wang, Felix Wimbauer, Zijian He, et al. Controlroom3d: Room generation using semantic proxy rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6201–6210, 2024. 2

[38] Lee M Seversky and Lijun Yin. Real-time automatic 3d scene generation from natural language voice and text descriptions. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 61–64, 2006. 2

[39] Cheng Sun, Chi-Wei Hsiao, Min Sun, and Hwann-Tzong Chen. Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1047–1056, 2019. 3

[40] 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. *arXiv preprint arXiv:2412.02193*, 2024. 2, 6

[41] Fan-Yun Sun, Harini SI, Angela Yi, Yihan Zhou, Alex Zook, Jonathan Tremblay, Logan Cross, Jiajun Wu, and Nick Haber. Factorsim: Generative simulation via factorized representation. *Advances in Neural Information Processing Systems*, 37:87438–87472, 2024. 8

[42] Fan-Yun Sun, Jonathan Tremblay, Valts Blukis, Kevin Lin, Danfei Xu, Boris Ivanovic, Peter Karkus, Stan Birchfield, Dieter Fox, Ruohan Zhang, et al. Partial-view object view synthesis via filtering inversion. In *International Conference on 3D Vision (3DV)*, pages 453–463, 2024. 8

[43] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20507–20518, 2024. 2, 6

[44] Junyang Wang, Yuhang Wang, Guohai Xu, Jing Zhang, Yukai Gu, Haitao Jia, Jiaqi Wang, Haiyang Xu, Ming Yan, Ji Zhang, et al. Amber: An llm-free multi-dimensional benchmark for mllms hallucination evaluation. *arXiv preprint arXiv:2311.07397*, 2023. 6

[45] Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Yixuan Li, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *arXiv preprint arXiv:2406.14852*, 2024. 2

[46] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023. 8

[47] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4818–4829, 2024. 8

[48] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3D mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 8

[49] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4v. *arXiv preprint arXiv:2310.11441*, 2023. 2

[50] Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16262–16272, 2024. 6

[51] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024. 2, 6

[52] Kaixin Yao, Longwen Zhang, Xinhao Yan, Yan Zeng, Qixuan Zhang, Lan Xu, Wei Yang, Jiayuan Gu, and Jingyi Yu. Cast: Component-aligned 3d scene reconstruction from an rgb image. *ACM Transactions on Graphics (TOG)*, 44(4): 1–19, 2025.

[53] Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Transactions on Visualization and Computer Graphics*, 22(2):1138–1148, 2016. 2

[54] Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bolei Zhou, Sergey Tulyakov, and Hsin-Ying Lee. SceneWiz3D: Towards text-guided 3d scene composition. *arXiv preprint arXiv:2312.08885*, 2023. 6

[55] Shijie Zhou, Zhiwen Fan, Dejia Xu, Haoran Chang, Pradyumna Chari, Tejas Bharadwaj, Suya You, Zhangyang Wang, and Achuta Kadambi. Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting. In *European Conference on Computer Vision*, pages 324–342. Springer, 2024. 2

[56] 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. *arXiv preprint arXiv:2402.07207*, 2024. 2, 6