SceneX: Procedural Controllable Large-scale Scene Generation

Mengqi Zhou[1,2,3,4*], **Yuxi Wang**[5*], **Jun Hou**[5], **Shougao Zhang**[6],
Yiwei Li[1], **Chuanchen Luo**[7], **Junran Peng**[8†], **Zhaoxiang Zhang**[1,2,3,4,5]
[1]University of Chinese Academy of Sciences
[2]Institute of Automation, Chinese Academy of Sciences
[3]State Key Laboratory of Multimodal Artificial Intelligence Systems
[4]New Laboratory of Pattern Recognition
[5]Centre for Artificial Intelligence and Robotics
[6]China University of Geosciences Beijing
[7]Shandong University
[8]University of Science and Technology Beijing
{zhoumengqi2022, zhaoxiang.zhang}@ia.ac.cn, yuxiwang93@gmail.com, jrpeng4ever@126.com

Abstract

Developing comprehensive explicit world models is crucial for understanding and simulating real-world scenarios. Recently, Procedural Controllable Generation (PCG) has gained significant attention in large-scale scene generation by enabling the creation of scalable, high-quality assets. However, PCG faces challenges such as limited modular diversity, high expertise requirements, and challenges in managing the diverse elements and structures in complex scenes. In this paper, we introduce a large-scale scene generation framework, SceneX, which can automatically produce high-quality procedural models according to designers' textual descriptions. Specifically, the proposed method comprises two components, PCGHub and PCGPlanner. The former encompasses an extensive collection of accessible procedural assets and thousands of hand-craft API documents to perform as a standard protocol for PCG controller. The latter aims to generate executable actions for Blender to produce controllable and precise 3D assets guided by the user's instructions. Extensive experiments demonstrated the capability of our method in controllable large-scale scene generation, including nature scenes and unbounded cities, as well as scene editing such as asset placement and season translation.

Code — https://zhouzq1.github.io/SceneX/
Extended version — https://arxiv.org/abs/2403.15698

Introduction

In the realm of Artificial General Intelligence (AGI), developing comprehensive world models is crucial for understanding and simulating real-world scenarios. Recent advancements, such as those demonstrated by the Sora model (Brooks et al. 2024), show progress in capturing physical laws and generating realistic simulations. However, Sora's outputs often lack detailed geometry and structured information, limiting their editability and interactivity. To address these limitations, explicit world models provide a more robust solution. By constructing worlds with detailed mesh assets and evolving them according to predefined physical rules, these models leverage Physically Based Rendering (PBR) to ensure high geometric consistency and detailed visualization. Procedural modeling methods, such as those outlined by Lindenmayer et al. (Lindenmayer 1968), show great promise in generating realistic and intricate world models through adjustable parameters and rule-based systems using tools like *Blender*. For example, Infinigen (Raistrick et al. 2023) proposes a procedural generator to generate large-scale natural scenes encompassing terrain, weather, vegetation, and wildlife. (Lipp et al. 2011) and (Talton et al. 2011) use procedural modeling to generate city-level street or layout. Although these procedural approaches generate high-quality 3D assets, they are beginner-unfriendly and time-consuming due to the comprehensive grasp of generation rules, algorithmic frameworks, and individual parameters of procedural modeling. For instance, generating a city, as shown in Fig. 1, requires the effort of a professional PCG engineer working for over two weeks.

To address the above problems, existing works such as 3D-GPT (Sun et al. 2023) and SceneCraft (Hu et al. 2024) introduce an instruction-driven 3D modeling method by integrating an LLM agent with procedural generation software. Despite the successful collaboration with human designers to establish a procedural generation framework, these methods still exhibit significant limitations. Firstly, the restricted editing capabilities of fixed 3D resources prevent SceneCraft from attaining the same degree of precision in 3D resource editing as is achievable through procedural generation. Secondly, 3D-GPT is based on the procedural generation model Infinigen, which restricts its capacity to fully utilise existing PCG resources. This is exemplified by the inability to generate extensive terrain and limitless cities. Finally, the task of generating a large-scale scene is often composed of multiple related subtasks, such as the planning of the scene layout, the generation and placement of assets, and the adjustment of environmental details. However, there is currently no effective way to manage the execution order and dependencies of these tasks.

In this paper, we introduce the SceneX framework, con-

[*] Equal contribution.
[†] Corresponding author.

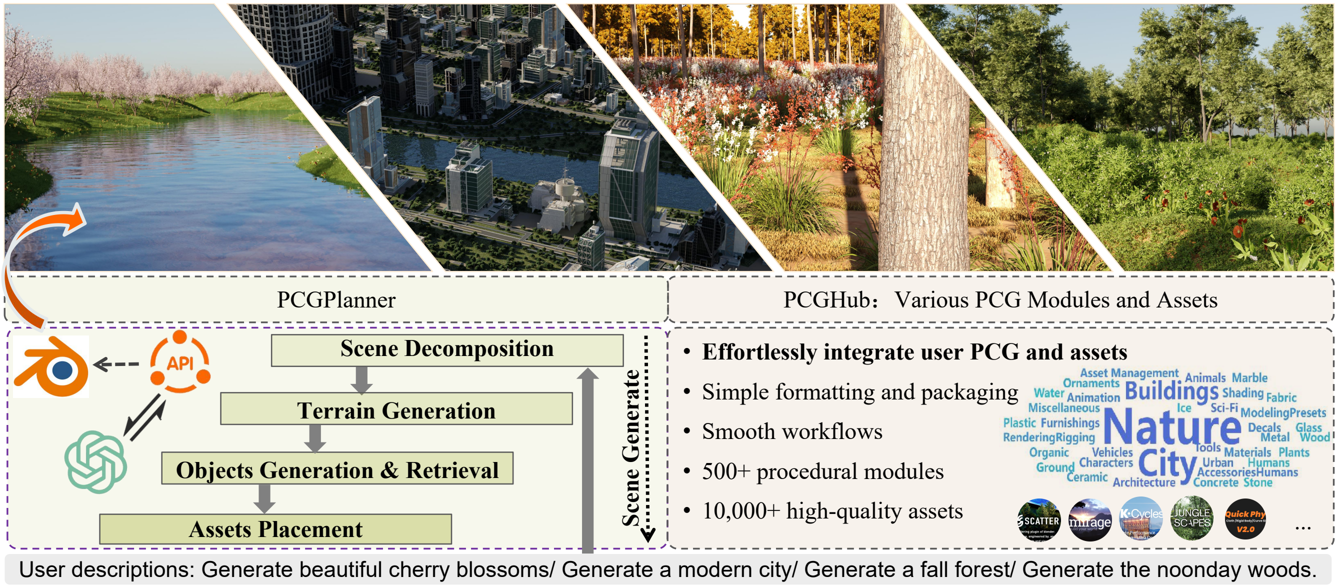


Figure 1: The proposed SceneX can create large-scale 3D natural scenes or unbounded cities automatically according to user instructions. The generated models are characterized by delicate geometric structures, realistic material textures, and natural lighting, allowing for seamless deployment in the industrial pipeline.

sisting of PCGHub and PCGPlanner. PCGHub addresses the limitations of individual PCG modules, which are constrained by their inherent algorithms and predefined rules. By integrating diverse procedural modules and encapsulating them with corresponding APIs, PCGHub offers a platform for continuously incorporating new PCG capabilities, thus supporting the expansion and diversity of procedural generation resources. Due to constraints in procedural algorithms and predefined rules, unrestricted combination of all procedural methods is infeasible. Thus, we develop the PCGPlanner to coordinate PCG methods within these constraints. PCGPlanner accesses and integrates procedural modules, enabling effective coordination and seamless integration. Our work improves scene generation flexibility and diversity, reduces entry barriers for non-coders, and efficiently uses existing technologies for accessible, community-friendly procedural generation. Our SceneX possesses several key properties:

1. *Efficiency:* Benefiting from the proposed PCGHub and PCGPlanner, our SceneX can rapidly produces extensive, high-quality 3D assets, including terrain, city, and forest. Moreover, we only need a few hours to generate a large-scale city, whereas it would take a professional designer over two weeks.

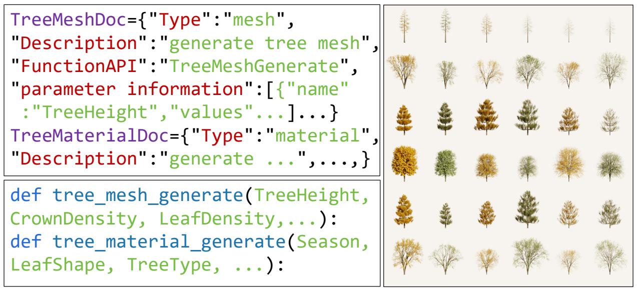
2. *Controllability:* SceneX can generate 3D scenes satisfying personalized demands. It achieves scene editing according to the descriptions, such as adding objects, object placement, translating the season, and so on.

3. *Diversity:* SceneX overcomes conventional generation constraints by integrating multiple subtasks, enabling flexible and diverse scene generation at scale.

Related Works

Learning Based 3D Generation. In recent years, 3D asset generation has witnessed rapid progress, combining the ideas of computer graphics and computer vision to realise the free creation of 3D content. Presently, predominant research in 3D asset generation primarily concentrate on creating individual objects (Poole et al. 2022; Liu et al. 2023; Lin et al. 2023a), 3D avatars (Zhang et al. 2023a; Kolotouros et al. 2024; Hong et al. 2022), and 3D scenes (Höllein et al. 2023; Zhang et al. 2024a; Fridman et al. 2024; Zhang et al. 2023c). Among these, ZeroShot123 (Liu et al. 2023) proposes a method based on a diffusion model, which realizes the 3D model of the target based on a picture. DreamFusion (Poole et al. 2022) proposes a NeRF-based approach that allows the model to generate a corresponding 3D model based on the input text. Compared with single object generation, it is more practical and challenging to generate large-scale scenes, including the generation of natural landforms (Hao et al. 2021; Liu et al. 2021; Chen, Wang, and Liu 2023) and borderless cities (Lin et al. 2023b; Li et al. 2023; Xie et al. 2024). CityDreamer (Xie et al. 2024) builds vast, large-scale 3D cities based on the layout of real cities, enhancing urban reconstruction accuracy and stability. SceneDreamer (Chen, Wang, and Liu 2023) proposes a method to generate 3D borderless scenes within 2D plots using BEV representations.

Procedural Based 3D Generation. Researchers have delved into the procedural generation of natural scenes (Gasch et al. 2022; Zhang et al. 2019) and urban scenes (Lipp et al. 2011; Talton et al. 2011; Vanegas et al. 2012; Yang et al. 2013) using Procedural Content Generation (PCG). For instance, PMC (Parish and Müller 2001) proposes a procedural way to generate cities based on 2D



```
TreeMeshDoc={"Type":"mesh",
"Description":"generate tree mesh",
"FunctionAPI":"TreeMeshGenerate",
"parameter information":[{"name"
 :"TreeHeight","values"...]...}
TreeMaterialDoc={"Type":"material",
"Description":"generate ...",...,}
```

```
def tree_mesh_generate(TreeHeight,
CrownDensity, LeafDensity,...):
def tree_material_generate(Season,
LeafShape, TreeType, ...):
```

Figure 2: Tree PCG API documentation, API functions, and tree generation results.

Capability	Num. PCG	Capability	Num. PCG
Terrain	12	Water	4
Weather	15	Snow	3
Vegetation	65	Assets placement	13
Buildings	37	Materials	361
Blocks	17	Dynamic People	4
Cities	12	Dynamic Vegetation	23
People	6	Dynamic Vehicles	3

Table 1: Overview of PCGHub capabilities.

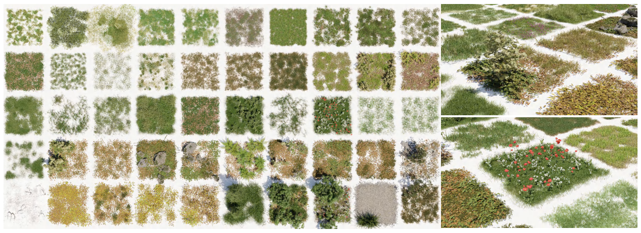


Figure 3: Visualization of scatter layout results, showcasing diverse ground cover effects generated using various vegetation assets.

ocean or city boundaries. It employs mathematical algorithms to generate blocks and streets and utilizes tailgating technology to generate the geometry of buildings. While these traditional computer graphic methods can produce high-quality 3D data, all parameters must be pre-entered into the procedurally generated process. This significantly constrains flexibility and practical usability. Infinigen (Raistrick et al. 2023; Zhang et al. 2024b) introduces a technique for procedurally generating realistic 3D natural objects and scenes. Although Infinigen generates infinitely assets, users are unable to customize the generated outcomes because of their specific requirements. In this paper, we propose a more convenient method to produce procedural assets.

LLM Agents. Benefiting from knowledge hidden in the large-language model (LLMs) (Raffel et al. 2020; Achiam et al. 2023a; Chowdhery et al. 2023), researchers explore LLMs to address intricate tasks beyond canonical language processing domains. These tasks encompass areas such as mathematical reasoning (Imani, Du, and Shrivastava 2023; Wei et al. 2022), medicine (Yang et al. 2023; Jeblick et al. 2023), and planning (Zhang et al. 2023b; Gong et al. 2023; Huang et al. 2023). Thanks to powerful reasoning and generalization capabilities, LLMs act as practiced planners for different tasks. For example, (Huang et al. 2022) utilizes the expansive domain knowledge of LLMs on the internet and their emerging zero-shot planning capabilities to execute intricate task planning and reasoning. (Gong et al. 2023) investigates the application of LLMs in scenarios involving multi-agent coordination, covering a range of diverse task objectives. (Zeng et al. 2022) presents a modular framework that employs structured dialogue through prompts among multiple models. Moreover, specialized LLMs for particular applications have been explored, such as HuggingGPT (Shen et al. 2024) for vision perception tasks, VisualChatGPT (Wu et al. 2023) for multi-modality understanding, Voyager (Wang et al. 2023) and (Zhu et al. 2023), SheetCopilot (Li et al. 2024) for office software, and Codex (Chen et al. 2021a) for Python code generation. Inspired by existing works, we explore the LLM agent to PCG software, *e.g., Blender,* to provide automatic 3D assets generation.

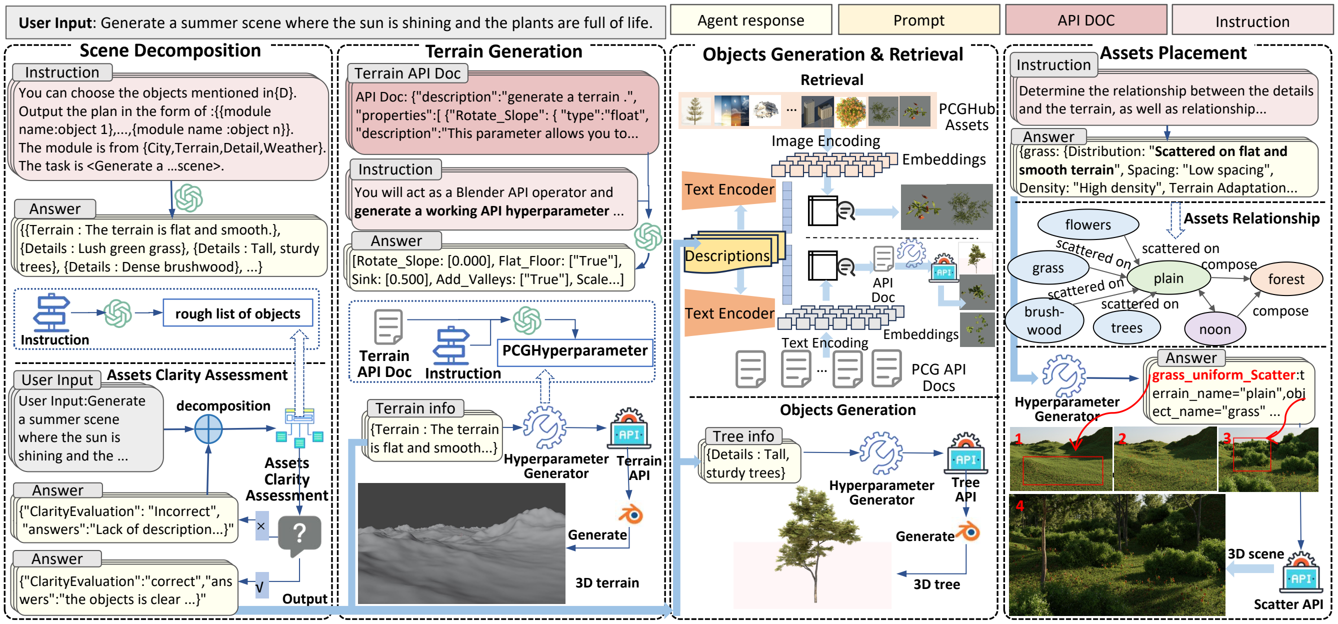
SceneX: All-in-One PCG Solution

We present the SceneX framework for versatile scene generation, which includes PCGHub and PCGPlanner. PCGHub integrates a vast array of procedural modules and 3D assets, offering extensive procedural capabilities, while PCGPlanner coordinates these procedural modules within a well-defined algorithmic structure. The framework seamlessly incorporates user-provided PCG and assets into the workflow. Leveraging user text input, the framework enables the efficient and precise generation of diverse, high-quality scenes.

PCGHub

For scene generation tasks, the diversity of scenes is intrinsically linked to the diversity of assets. Therefore, we introduce PCGHub, a platform that integrates a wide range of PCG modules and 3D assets. It provides detailed documentation and APIs for swift integration of varied PCG techniques, addressing the limitations of traditional methods and improving content realism. A summary of the extensive PCG modules available in PCGHub is provided in Table 1.

Procedural Assets. To develop PCGHub, we assemble a comprehensive collection of 172 procedural assets, categorized into natural environments, architecture, biology, environmental impacts, and science fiction elements. These assets are generated using Blender node graphs, which allow the creation of a virtually infinite range of variations through adjustments to geometry and shader parameters. From the original 2,362 parameters, we extract 263 human-interpretable parameters and encapsulate them into APIs. Each API is comprehensively documented, including functional descriptions, parameter specifications, and types, to facilitate their utilization via language models. Fig. 2 illustrates an example of API documentation, the corresponding APIs, and the resulting trees generated through these APIs. To further address the issue of limited PCG asset variety, we also collect 11,284 high-quality 3D static assets, enhancing the overall diversity and richness of the available assets.



User Input: Generate a summer scene where the sun is shining and the plants are full of life.

Instruction: You can choose the objects mentioned in{D}. Output the plan in the form of :{{module name:object 1},...,{module name :object n}}. The module is from {City,Terrain,Detail,Weather}. The task is <Generate a ...scene>.

Answer: {{Terrain : The terrain is flat and smooth.}, {Details : Lush green grass}, {Details : Tall, sturdy trees}, {Details : Dense brushwood}, ...}

User Input: User Input:Generate a summer scene where the sun is shining and the ...

Answer: {"ClarityEvaluation": "Incorrect", "answers":"Lack of description...}"

Answer: {"ClarityEvaluation":"correct","answers":"the objects is clear ...}"

Terrain API Doc: API Doc: {"description":"generate a terrain .", "properties":[{"Rotate_Slope": { "type":"float", "description":"This parameter allows you to...

Instruction: You will act as a Blender API operator and **generate a working API hyperparameter** ...

Answer: [Rotate_Slope: [0.000], Flat_Floor: ["True"], Sink: [0.500], Add_Valleys: ["True"], Scale...]

Terrain info: {Terrain : The terrain is flat and smooth...}

Tree info: {Details : Tall, sturdy trees}

Instruction: Determine the relationship between the details and the terrain, as well as relationship...

Answer: {grass: {Distribution: "**Scattered on flat and smooth terrain**", Spacing: "Low spacing", Density: "High density", Terrain Adaptation...

Answer: **grass_uniform_Scatter**:terrain_name="plain",object_name="grass" ...

Figure 4: SceneX framework converting user text input into diverse 3D scenes through four stages: scene decomposition stage, terrain generation stage, objects generation & retrieval stage and assets placement stage.

Procedural Layout Generators. Complex arrangements exist in various environments. To replicate these patterns, we predefine five types of layouts: scatter layout, which distributes assets randomly within a given area; grid layout, which arranges objects in a uniform grid pattern; linear layout, which arranges objects sequentially along a defined path, such as roads, rivers, or railways; nested layout, which organizes objects within a larger structure, such as buildings in a small neighborhood or attractions within a park; and area filling layout, which places objects according to specific rules to fill an entire designated area. For each layout type, we provide one or more corresponding procedural layout generators. By utilizing various objects for each layout type, we can generate a diverse range of scenes. Fig. 3 demonstrates the scatter layout's effectiveness.

PCGPlanner

PCGPlanner utilizes the resources provided by PCGHub for efficient scene generation. As illustrated in Fig. 4, the fully automated scene generation process comprises the following stages:(1) *Scene Decomposition*: Analyzes the scene requirements to identify the necessary assets; (2) *Terrain Generation*: Constructs the foundational terrain and applies the appropriate materials; (3) *Objects Generation & Retrieval*: Involves generating or importing the assets required for the scene; (4) *Asset Placement*: Utilizes diverse layout types and procedural generators to arrange assets within the scene.
Systematic Template. To endow the LLMs with the capability for modeling scenes, We introduce a systematic template to prompt LLMs in a scene modeling task. For each agent, the prompt P has a similar structure defined as P_{i} (R_{i}, T_{i}, D_{i}, F_{i}, E_{i}), where i corresponds to different subtasks . The constituents of the prompt are defined as follows:

- **Role** Each agent is given a specific role R which describes its responsibility in the scene generation process.
- **Task** T gives a detailed explanation of the goals for the agent. In the meanwhile, the constraints are also expounded executing these tasks.
- **Document** At each step, the agent is prompted by a knowledge document according to their task. We denote D as the collection that contains all the knowledge documentation pre-defined in PCGHub.
- **Format** F denotes the output format for each agent. To precisely and concisely convey information between agents, the output format of each agent is strictly defined.
- **Examples** Examples are provided as reference to help agents follow the format strictly.

The systematic template guides LLMs to produce accurate outputs, enhancing the success rate and executability rate.
Scene Decomposition Stage. As illustrated in Fig. 4, this stage converts user textual descriptions into a rough list of objects for the target scene. The detailed object requirements and their corresponding procedural modules are outlined in the knowledge documentation $D_{decomposition}$. The process can be formulated as follows:

$$\{o_1, \ldots, o_n\} \leftarrow LLM_{decomposition}(q, P_{decomposition})$$

where q is the user input, and o_i represents an object with its associated modules and descriptions structured as $\{\{$*module: description*$\}$, ... $\{$*module: description*$\}\}$.

Additionally, recognizing that user may omit essential details, we incorporate an Assets Clarity Assessment to identify and resolve ambiguities by actively querying users to obtain missing details for a complete preliminary plan.

Hyperparameter Generator. The Hyperparameter Generator (HyperparamGen) performs as an executor, which converts textual descriptions into PCG parameters and applies these parameters to APIs. Within PCGHub, each API is defined as a Python function with multiple predefined parameters. For each API, a corresponding knowledge document $D_\alpha \in D$ is provided to the Hyperparameter Generator. This document contains essential details required to generate the appropriate hyperparameters for executing the API. The hyperparameter generation process is formulated as follows:

$$\alpha^* \leftarrow \text{HyperparamGen}(\alpha, P_\alpha, o_i) \tag{1}$$

where α represents the initial hyperparameters and α^* denotes the optimized parameters. In cases where the description may include incomplete parameter information, the Hyperparameter Generator is instructed to automatically select feasible parameters from available options based on the other parameters. The inferred parameters are utilized to execute the corresponding APIs in the Blender Python environment, completing all tasks efficiently.

Terrain Generation Stage. This stage is responsible for terrain generation within the scene. Unlike other assets, terrain lacks distinct characteristics that can be effectively identified using CLIP (Radford et al. 2021). Thus, a specialized terrain generation PCG is employed to ensure accurate and detailed control. As shown in the second panel of Fig. 4, the PCG provides extensive control over terrain attributes, including geometry and material properties. It enables precise adjustments to terrain slope, elevation, and features such as valleys, ensuring that the generated terrain meets specific scene requirements with both flexibility and accuracy.

Objects Generation & Retrieval Stage. Based on the detailed object descriptions provided by the Scene Decomposition Stage, we need to search within PCGHub for assets to directly import or suitable APIs to generate procedural assets. To achieve accurate retrieval, the Objects Generation & Retrieval Stage manages this process by encoding these descriptions into embeddings for retrieval. A pre-trained CLIP model is employed: text-to-text retrieval is used for procedural assets APIs, and text-to-image retrieval is used for 3D static assets. As depicted in the third panel of Fig. 4, each static asset in PCGHub is represented by a 768-dimensional vector derived from its rendering image. These vectors are compared with the input description embedding, and one of the top five most similar results is randomly selected based on cosine similarity and imported into the Blender scene.

$$\text{Asset}_j \leftarrow \text{Retrieval}(o_j) \tag{2}$$

where $\text{Asset}_j \in \Gamma$ represents the retrieved static asset from the asset collection Γ. For text-to-text retrieval, the most relevant API is also selected by its functional description embedding. If the retrieval result is an asset, it will be directly imported into the scene project. On the other hand, the retrieved API is to be passed to the Hyperparameter Generator.

Assets Placement. In large-scale scene generation tasks, asset placement is inherently complex. It involves not only identifying spatial relationships between objects but also determining how these relationships should be represented. To address this, we propose a method based on asset relationships to guide asset placement. As illustrated in the fourth panel of Fig. 4, the LLM, guided by the predefined layout types, selects the appropriate layout for each object. Subsequently, the LLM defines the spatial relationships among assets in the scene. Each asset relationship pair is processed by the Hyperparameter Generator, which generates the necessary parameters for asset placement. These parameters are then utilized by the Procedural Layout Generators to arrange the assets accordingly in Blender.

Experiments

The goals of our experiments are threefold: (i) to verify the capability of SceneX for generating photorealistic large-scale scenes, including nature scenes and cities, (ii) to demonstrate the effectiveness of SceneX for personalized editing, such as adding or changing,(iii) to compare different LLMs on the proposed benchmark.

Benchmark Protocol

Dataset. To evaluate the effectiveness of proposed SceneX, we use GPT-4 to generate high-quality 50 scene descriptions, 50 asset descriptions, and 20 asset editing descriptions. The scene descriptions involve natural scenes and cities. Then, we feed them to our SceneX to generate corresponding models, which are used to perform quantitative and qualitative comparisons.

Models. When generating and editing the 3D scenes, we adopt the leading GPT-4 as the large language model with its public API keys. To ensure the stability of LLM's output, we set the decoding temperature as 0.

Metrics. We use Executability Rate (ER@1) and Success Rate (SR@1) to evaluate LLMs on our SceneX. The former measures the proportion of proposed actions that can be executed, and the latter is used to evaluate action correctness (Chen et al. 2021b). Moreover, to quantify aesthetic quality, we adopt a unified judgment standard as a reference. We divide the aesthetics of generated scenes into five standards: Poor (1-2 points)/Below Average (3-4 points)/Average (5-6 points)/Good (7-8 points)/Excellent (9-10 points). We enlisted 35 volunteers to assess the quality of our generation, including 5 PCG experts. We use the average score (AS) and average expert score (AES) to evaluate our method.

Main Results

Generation Quality. We begin by showcasing several examples of our SceneX for generating large-scale nature scenes and unbounded cities. The results are shown in Fig. 5. From the results, we can observe that the proposed SceneX can produce highly realistic scenes in both natural scenes and cities. Moreover, the generated content is correctly corresponding to the provided texture descriptions. These demonstrate the power and effectiveness of our proposed LLM-driven automatic 3D scene generation framework. Fig. 6 shows the qualitative comparison results between the learning-based methods for city scene reconstruction work and SceneX. From the results, we can observe that learning-based methods commonly suffer similar problems:

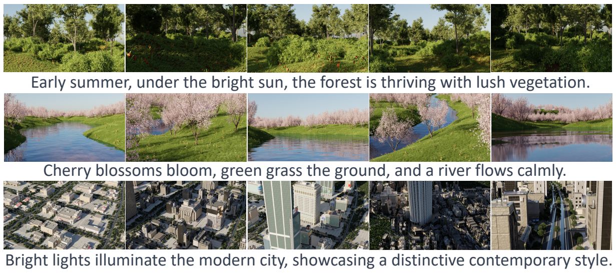

Early summer, under the bright sun, the forest is thriving with lush vegetation.
Cherry blossoms bloom, green grass the ground, and a river flows calmly.
Bright lights illuminate the modern city, showcasing a distinctive contemporary style.

Figure 5: Visualization of the generation quality for large-scale scenes and cities.

Method	AS	AES
Magic 3D (Lin et al. 2023a)	4.48	3.50
DreamFusion (Poole et al. 2022)	4.55	3.60
Text2Room (Höllein et al. 2023)	5.73	6.10
CityDreamer (Xie et al. 2024)	5.47	6.80
Infinigen (Raistrick et al. 2023)	5.42	6.00
3D-GPT (Sun et al. 2023)	4.94	6.20
WonderJ (Yu et al. 2024)	5.28	6.00
SceneX (Ours)	**7.83**	**7.70**

Table 2: Comparative analysis of average score (AS) and average expert score (AES).

low 3D consistency and building structural distortions. For example, PersistanNature (Chai et al. 2023) and InfiniCity (Lin et al. 2023b) both appear to have severe deformation in the whole scene level. SceneDreamer (Chen, Wang, and Liu 2023) and CityDreamer (Xie et al. 2024) have better structure consistency, but the building quality is still relatively low. These factors limit their large-scale application in industry. In comparison, the proposed SceneX generates highly realistic and well-structured urban scenes without the issues of structural distortions and layout defects compared to learning-based methods. These results demonstrate the effectiveness of SceneX for large-scale city scene generation.

Aesthetic Evaluation. To better evaluate the generation quality of SceneX, we collect the results of related works involving text-to-3D work and Blender-driven 3D generation. These results are subjected to aesthetic evaluation by a panel comprising of 35 voluntary contributors and 5 experts in 3D modeling, with the scoring criteria outlined in Section 4.1. As shown in Table 2, our scores for AS and AES surpass the second-highest scores by 2.10 and 0.9 points, respectively. Compared to the other works, our project reaches a good level, indicating the high generation quality of SceneX.

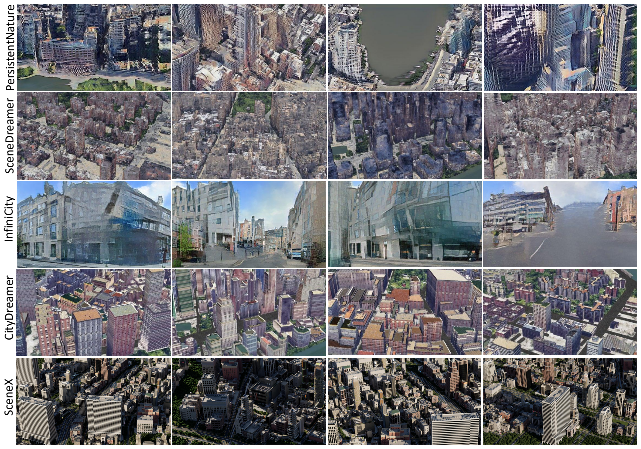


Figure 6: Comparative results on city generation.

	V-L/14	V-B/16	V-B/32
WonderJ (Yu et al. 2024)	18.78	25.70	25.45
Text2Room (Höllein et al. 2023)	23.51	30.10	29.29
Magic 3D (Lin et al. 2023a)	27.86	31.78	31.94
DreamFusion (Poole et al. 2022)	**29.40**	**35.37**	**31.60**
BlenderGPT	21.23	25.65	26.19
3D-GPT (Sun et al. 2023)	18.67	25.80	25.59
SceneCraft (Hu et al. 2024)	22.04	25.82	25.30
SceneX (Ours)	22.82	27.82	26.89

Table 3: Assessing prompt-rendered result similarity with various models.

	150m × 150m		500m × 500m		2.5km × 2.5km	
	Scene	City	Scene	City	Scene	City
Human	1h	40min	3h	4h	-	>3w
Infinigen	14min	-	-	-	-	-
SceneX	**2min**	**1.5min**	**10min**	**6min**	-	**20h**

Table 4: Comparing the time required for natural scene generation and city generation at different terrain scales.

To evaluate the consistency between text inputs and the generated assets, we calculate the CLIP similarity between input text and rendered images. To better illustrate the results, we utilize three different CLIP models for testing, including ViT-L/14 (V-L/14), ViT-B/16 (V-B/16) and ViT-B/32 (V-B/32), respectively. The detailed results are displayed in Table 3. We compare representative text-to-3D approaches (*e.g.* WonderJ (Yu et al. 2024), Text2Room (Höllein et al. 2023), and DreamFusion (Poole et al. 2022)) and Blender-driven 3D generation works (*e.g.* BlenderGPT, 3D-GPT (Sun et al. 2023), and SceneCraft (Hu et al. 2024)). Although the similarity scores of text-to-3D methods are higher, it is reasonable because their training or optimization includes the text-to-image alignment process. Compared to the Blender-driven 3D generation works, SceneX achieves the highest score, indicating its capability to accurately execute the input prompts and generate results.

Efficiency Evaluation. To illustrate the efficiency of SceneX, we provide the time required of our method compared with Infinigen (Raistrick et al. 2023) and human craft. The experiments are performed on a server equipped with dual Intel Xeon Processors (Skylake architecture), each with 20 cores, totaling 80 CPU cores. Additionally, we consult 3D PCG experts to determine the time needed to construct the same natural and urban scenes. Table 4 shows SceneX is 7 times faster than Infinigen in generating 150m × 150m scenes, and 30 times faster than manual creation of large-scale cities by 3D PCG experts.

	Task	Document	Examples	Role	ER@1	SR@1
1	✓				16.00	25.00
2		✓			42.00	47.62
3	✓	✓			84.00	71.43
4			✓		92.00	73.91
5	✓		✓		92.00	76.09
6		✓	✓		92.00	76.09
7	✓	✓	✓		92.00	78.26
8		✓	✓	✓	94.00	78.72
9	✓	✓	✓	✓	**94.00**	**80.85**

Table 5: Results of different prompt components for tree asset generation. Examples, Role, Document and Task represent four integral components of the prompt template.

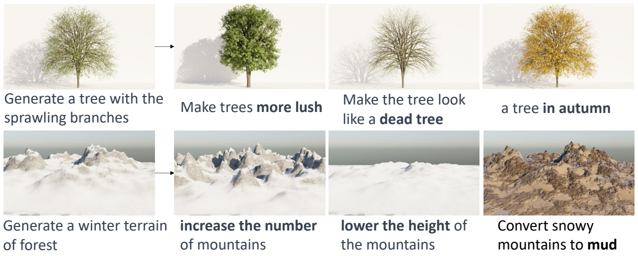

Generate a tree with the sprawling branches
Make trees **more lush**
Make the tree look like a **dead tree**
a tree **in autumn**
Generate a winter terrain of forest
increase the number of mountains
lower the height of the mountains
Convert snowy mountains to **mud**

Figure 7: Visualization of the personalized editing results.

Personalized Editing Results. To demonstrate the capability of our method for personalized editing, we conduct experiments on 3D asset generation guided by users' instructions. The results are shown in Fig. 7. It is evident that the changes in the edited text are closely related to the modifications in 3D assets. SceneX demonstrates a versatile, highly controllable, and personalized editing ability by manipulating 3D assets from various perspectives. These results demonstrate that our method supports user-instruction editing, significantly reducing the difficulty of 3D asset generation and accelerating the industrial production process.

Ablation Study

To analyze the impact of various components within the systematic template, we conduct an ablation study based on the tree plugin in PCGHub. During the experiment, we utilize the dataset from Section 4.1 for testing, maintaining a consistent format. We incrementally add or remove different parts of the systematic template, using ER@1 and SR@1 as metrics to observe the impact of various components on the system. The results are shown in Table 5. It is evident from the results that augmenting the Task, Document, Examples and Role components contributes to an increase in ER@1 and SR@1. Among these, the inclusion of the Example component results in the most significant improvement, resulting in a maximum increase of 76.00% and 51.09% in ER@1 and SR@1 respectively. Conversely, the Role component has the least impact, with maximum increases of 2.00% and 2.59% in ER@1 and SR@1 after its addition. These results suggest that the setting of Task significantly impacts the performance of LLMs. By clarifying the goals and requirements of the task, the system can better understand the user's intent and better meet the needs when generating output. Examples play an important role in designing proxy prompts. By providing concrete examples, the system can better understand the user's intent and needs and produce high-quality output related to the input text. Documentation provides background information that can help the system better meet user expectations when generating output.

Model	Scene Generation		Asset Generation	
	ER@1	SR@1	ER@1	SR@1
Llama2-7B(Touvron et al. 2023)	30.00	53.33	38.00	57.89
Llama2-13B(Touvron et al. 2023)	44.00	59.09	54.00	66.66
Mistral(Jiang et al. 2023)	76.00	68.42	94.00	85.11
Gemma-2B(Team et al. 2024)	6.00	33.33	22.00	45.45
Gemma-7B(Team et al. 2024)	36.00	55.56	68.00	73.53
GPT-3.5-turbo(Brown et al. 2020)	66.00	60.60	82.00	82.93
GPT-4(Achiam et al. 2023b)	**86.00**	**86.05**	**96.00**	**85.42**

Table 6: Comparing the performance of different language models in natural scene generation and city generation.

Comparing with Different LLMs

To investigate the performance of different variants of large language models (LLM) in SceneX, we test public LLM APIs like gpt-3.5-turbo, gpt-4 and several external open-source LLMs in this subsection. To ensure the stability of LLM outputs, we set the temperature of LLM to 0 for all experiments. We conduct experiments on 3D scene and asset generation based on 50 scenario descriptions and 50 object descriptions in Sec. 4.1. The results are presented in Table 6. It is evident that gpt-4 delivers the superior performance, with Mistral closely following as the second-best. Due to its performance and lower hardware requirements, the open-source Mistral is a highly appealing option. When compared to asset generation, the executability and success rates noticeably decline during the generation of large-scale natural scenes, a trend that can be attributed to the increased task complexity. In particular, as the number of components involved in the system expands, the LLM may face challenges in maintaining their accuracy. Nonetheless, our method exhibits consistent performance across different LLMs, maintaining high levels of executability and success rates.

Conclusion

We present SceneX, a framework for automated large-scale scene generation from text descriptions. It comprises PCGHub, a repository of procedural assets and API documents, and PCGPlanner, which generates Blender actions to create 3D assets based on user instructions. SceneX can generate a 2.5 km × 2.5 km city in hours instead of weeks of professional work. Experiments validate its effectiveness in scene generation and editing.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. U21B2042, No. 62320106010, No. 62072457), and the 2035 Innovation Program of CAS, and the InnoHK program.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023a. Gpt-4 technical report. *arXiv:2303.08774*.

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023b. Gpt-4 technical report. *arXiv:2303.08774*.

Brooks, T.; Peebles, B.; Holmes, C.; DePue, W.; Guo, Y.; Jing, L.; Schnurr, D.; Taylor, J.; Luhman, T.; Luhman, E.; et al. 2024. Video generation models as world simulators.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Proc. NIPSAdvances in neural information processing systems*, 33: 1877–1901.

Chai, L.; Tucker, R.; Li, Z.; Isola, P.; and Snavely, N. 2023. Persistent Nature: A Generative Model of Unbounded 3D Worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20863–20874.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021a. Evaluating large language models trained on code. *arXiv:2107.03374*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021b. Evaluating large language models trained on code. *arXiv:2107.03374*.

Chen, Z.; Wang, G.; and Liu, Z. 2023. Scenedreamer: Unbounded 3d scene generation from 2d image collections. *IEEE transactions on pattern analysis and machine intelligence*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Fridman, R.; Abecasis, A.; Kasten, Y.; and Dekel, T. 2024. Scenescape: Text-driven consistent scene generation. *Advances in Neural Information Processing Systems*, 36.

Gasch, C.; Sotoca, J.; Chover, M.; Remolar, I.; and Rebollo, C. 2022. Procedural modeling of plant ecosystems maximizing vegetation cover. *Multimedia Tools and Applications*, 81.

Gong, R.; Huang, Q.; Ma, X.; Vo, H.; Durante, Z.; Noda, Y.; Zheng, Z.; Zhu, S.-C.; Terzopoulos, D.; Fei-Fei, L.; et al. 2023. Mindagent: Emergent gaming interaction. *arXiv preprint arXiv:2309.09971*.

Hao, Z.; Mallya, A.; Belongie, S.; and Liu, M.-Y. 2021. Gancraft: Unsupervised 3d neural rendering of minecraft worlds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14072–14082.

Höllein, L.; Cao, A.; Owens, A.; Johnson, J.; and Nießner, M. 2023. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7909–7920.

Hong, F.; Zhang, M.; Pan, L.; Cai, Z.; Yang, L.; and Liu, Z. 2022. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *arXiv:2205.08535*.

Hu, Z.; Iscen, A.; Jain, A.; Kipf, T.; Yue, Y.; Ross, D. A.; Schmid, C.; and Fathi, A. 2024. SceneCraft: An LLM Agent for Synthesizing 3D Scenes as Blender Code. In *Forty-first International Conference on Machine Learning*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, 9118–9147. PMLR.

Huang, W.; Wang, C.; Zhang, R.; Li, Y.; Wu, J.; and Fei-Fei, L. 2023. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv:2307.05973*.

Imani, S.; Du, L.; and Shrivastava, H. 2023. Mathprompter: Mathematical reasoning using large language models. *arXiv:2303.05398*.

Jeblick, K.; Schachtner, B.; Dexl, J.; Mittermeier, A.; Stüber, A. T.; Topalis, J.; Weber, T.; Wesp, P.; Sabel, B. O.; Ricke, J.; et al. 2023. ChatGPT makes medicine easy to swallow: an exploratory case study on simplified radiology reports. *European radiology*, 1–9.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Kolotouros, N.; Alldieck, T.; Zanfir, A.; Bazavan, E.; Fieraru, M.; and Sminchisescu, C. 2024. Dreamhuman: Animatable 3d avatars from text. *Advances in Neural Information Processing Systems*, 36.

Li, H.; Su, J.; Chen, Y.; Li, Q.; and ZHANG, Z.-X. 2024. SheetCopilot: Bringing software productivity to the next level through large language models. *Advances in Neural Information Processing Systems*, 36.

Li, Y.; Jiang, L.; Xu, L.; Xiangli, Y.; Wang, Z.; Lin, D.; and Dai, B. 2023. Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3205–3215.

Lin, C.-H.; Gao, J.; Tang, L.; Takikawa, T.; Zeng, X.; Huang, X.; Kreis, K.; Fidler, S.; Liu, M.-Y.; and Lin, T.-Y. 2023a. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 300–309.

Lin, C. H.; Lee, H.-Y.; Menapace, W.; Chai, M.; Siarohin, A.; Yang, M.-H.; and Tulyakov, S. 2023b. Infinicity: Infinite-scale city synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 22808–22818.

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

Lipp, M.; Scherzer, D.; Wonka, P.; and Wimmer, M. 2011. Interactive modeling of city layouts using layers of procedural content. In *Computer Graphics Forum*, volume 30, 345–354. Wiley Online Library.

Liu, A.; Tucker, R.; Jampani, V.; Makadia, A.; Snavely, N.; and Kanazawa, A. 2021. Infinite Nature: Perpetual View Generation of Natural Scenes from a Single Image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Liu, R.; Wu, R.; Van Hoorick, B.; Tokmakov, P.; Zakharov, S.; and Vondrick, C. 2023. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9298–9309.

Parish, Y.; and Müller, P. 2001. Procedural Modeling of Cities. volume 2001, 301–308.

Poole, B.; Jain, A.; Barron, J. T.; and Mildenhall, B. 2022. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551.

Raistrick, A.; Lipson, L.; Ma, Z.; Mei, L.; Wang, M.; Zuo, Y.; Kayan, K.; Wen, H.; Han, B.; Wang, Y.; et al. 2023. Infinite Photorealistic Worlds using Procedural Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12630–12641.

Shen, Y.; Song, K.; Tan, X.; Li, D.; Lu, W.; and Zhuang, Y. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

Sun, C.; Han, J.; Deng, W.; Wang, X.; Qin, Z.; and Gould, S. 2023. 3d-gpt: Procedural 3d modeling with large language models. *arXiv:2310.12945*.

Talton, J. O.; Lou, Y.; Lesser, S.; Duke, J.; Mech, R.; and Koltun, V. 2011. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2): 11–1.

Team, G.; Mesnard, T.; Hardin, C.; Dadashi, R.; Bhupatiraju, S.; Pathak, S.; Sifre, L.; Rivière, M.; Kale, M. S.; Love, J.; et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv:2403.08295*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*.

Vanegas, C.; Kelly, T.; Weber, B.; Halatsch, J.; Aliaga, D.; and Müller, P. 2012. Procedural Generation of Parcels in Urban Modeling. *Computer Graphics Forum*, 31: 681–690.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv:2305.16291*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Wu, C.; Yin, S.; Qi, W.; Wang, X.; Tang, Z.; and Duan, N. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv:2303.04671*.

Xie, H.; Chen, Z.; Hong, F.; and Liu, Z. 2024. Citydreamer: Compositional generative model of unbounded 3d cities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9666–9675.

Yang, K.; Ji, S.; Zhang, T.; Xie, Q.; and Ananiadou, S. 2023. On the evaluations of chatgpt and emotion-enhanced prompting for mental health analysis. *arXiv:2304.03347*.

Yang, Y.-L.; Wang, J.; Vouga, E.; and Wonka, P. 2013. Urban pattern: layout design by hierarchical domain splitting. *ACM Trans. Graph.*, 32(6).

Yu, H.-X.; Duan, H.; Hur, J.; Sargent, K.; Rubinstein, M.; Freeman, W. T.; Cole, F.; Sun, D.; Snavely, N.; Wu, J.; et al. 2024. Wonderjourney: Going from anywhere to everywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6658–6667.

Zeng, A.; Attarian, M.; Ichter, B.; Choromanski, K.; Wong, A.; Welker, S.; Tombari, F.; Purohit, A.; Ryoo, M.; Sindhwani, V.; et al. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv:2204.00598*.

Zhang, C.; Chen, Y.; Fu, Y.; Zhou, Z.; Yu, G.; Wang, B.; Fu, B.; Chen, T.; Lin, G.; and Shen, C. 2023a. StyleAvatar3D: Leveraging Image-Text Diffusion Models for High-Fidelity 3D Avatar Generation. *arXiv:2305.19012*.

Zhang, C.; Yang, K.; Hu, S.; Wang, Z.; Li, G.; Sun, Y.; Zhang, C.; Zhang, Z.; Liu, A.; Zhu, S.-C.; et al. 2023b. Proagent: Building proactive cooperative ai with large language models. *CoRR*.

Zhang, J.; Li, X.; Wan, Z.; Wang, C.; and Liao, J. 2024a. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*.

Zhang, J.; Wang, C.-b.; Qin, H.; Chen, Y.; and Gao, Y. 2019. Procedural modeling of rivers from single image toward natural scene production. *The Visual Computer*, 35.

Zhang, Q.; Wang, C.; Siarohin, A.; Zhuang, P.; Xu, Y.; Yang, C.; Lin, D.; Zhou, B.; Tulyakov, S.; and Lee, H.-Y. 2023c. SceneWiz3D: Towards Text-guided 3D Scene Composition. *arXiv:2312.08885*.

Zhang, S.; Zhou, M.; Wang, Y.; Luo, C.; Wang, R.; Li, Y.; Yin, X.; Zhang, Z.; and Peng, J. 2024b. CityX: Controllable Procedural Content Generation for Unbounded 3D Cities. *arXiv:2407.17572*.

Zhu, X.; Chen, Y.; Tian, H.; Tao, C.; Su, W.; Yang, C.; Huang, G.; Li, B.; Lu, L.; Wang, X.; et al. 2023. Ghost in the Minecraft: Generally Capable Agents for Open-World Enviroments via Large Language Models with Text-based Knowledge and Memory. *arXiv:2305.17144*.