FurniMAS: An LLM-Based Multi-Agent System for Language-Guided Furniture Decoration

Anonymous ACL submission

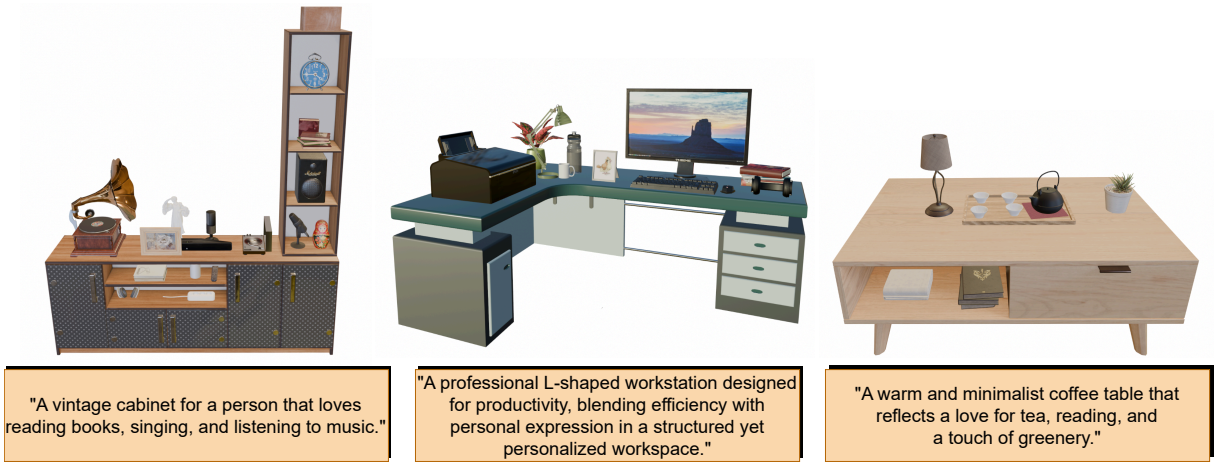

Figure 1: Our FurniMAS system interprets text descriptions and generates thoughtfully arranged, aesthetically pleasing furniture decorative outcomes that seamlessly align with user preferences and functional requirements.

Abstract

Furniture decoration is an important task in various industrial applications. However, achieving a high-quality decorative result is often time-consuming and requires specialized artistic expertise. To tackle these challenges, we explore how LLM-based multi-agent systems can assist in automating the decoration process. We propose FurniMAS, a multi-agent system for automatic furniture decoration. Specifically, given a human prompt and a household furniture item such as a working desk or a TV stand, our system suggests relevant assets with appropriate styles and materials, and arranges them on the item, ensuring the decorative result meets functionality, aesthetic, and ambiance preferences. FurniMAS assembles a hybrid team of LLM-based and heuristic agents, each fulfilling distinct roles in a typical decoration project. These agents collaborate through communication, logical reasoning, and validation to transform the requirements into the final outcome. Extensive experiments demonstrate that our FurniMAS significantly outperforms other baselines in generating high-quality 3D decor.

1 Introduction

Furniture decoration plays an important role in our daily lives (Zhang et al., 2023). A well-organized working desk can significantly enhance functionality, boost creativity, and increase productivity, while a thoughtfully decorated TV stand can greatly promote relaxation and alleviate stress. While decorating furniture can be fascinating, mastering it demands a keen sense of style, a knack for color harmony, and an eye for proportion when arranging various elements. These intricacies can quickly become a maze of complexity, leaving individuals feeling overwhelmed. An automated, intelligent decoration system that caters to users' preferences while navigating these complex design principles would represent a highly valuable and innovative tool. It would allow us to effortlessly transform our living spaces, making the process accessible and efficient for everyone. Additionally, an automatic decoration system will also largely benefit quick and effective environment construction in many applications in game development and robotics simulation (Wang et al., 2024b; Yang et al., 2024b; Lee et al., 2025). Developing such a system, however, is an under-explored and challenging task due to all the complexities of the decoration process with many possible assets and combinations (Patil et al., 2024; Chaudhuri et al., 2020).

Recently, Large Language Models (LLMs) with their commonsense knowledge have shown

significant advancements in various reasoning tasks (Deng et al., 2025; Xu et al., 2024; Yang et al., 2024b,c). (Deng et al., 2025) demonstrated the effectiveness of LLMs in visual layout planning tasks, while (Xu et al., 2024) leveraged LLMs for 3D avatar editing. Developing an LLM-based method for furniture decoration is, therefore, a promising approach. However, a shared limitation of most existing LLM-based methods lies in their *"single-agent"* design, where a single LLM handles the entire task. This approach was proven to be suboptimal for complex tasks like furniture decoration, which involves multiple stages – from defining the overall layout to selecting assets and their styles and materials, to arranging them appropriately (Guo et al., 2024; Yang et al., 2024a).

In this work, we aim to develop a *multi-agent* system utilizing state-of-the-art LLMs for furniture decoration. Although less explored in vision domains, multi-agent methods have been extensively proven to be effective in simulating real-world problem-solving environments that demand complex interactions and collaborative efforts of human groups, including software development (Hong et al., 2024; Qian et al., 2024), industrial control (Xia et al., 2023; Song et al., 2023), and human behavior simulation (Park et al., 2023; Peigné et al., 2025). A multi-agent system for furniture decoration would significantly facilitate ***(i) specification*** where different agents specialize in each area of decoration, ensuring that each aspect is handled with a high degree of expertise; ***(ii) collaborative decision-making*** where agents can collaborate to refine and improve the design, leading to a more cohesive and aesthetically pleasing outcome; and ***(iii) flexibility***, allowing individual agents or groups of agents to adjust their strategies in tasks like decor editing in response to shifts in user preferences, without affecting the entire system.

We introduce **FurniMAS** – an LLM-based multi-agent system for the challenging task of automatic furniture decoration. Given an empty furniture item (e.g., a working desk, a coffee table, a bookshelf) and a text prompt specifying human requirements, our agents collaborate and refine several decoration steps to arrive at the final outcome. Each agent in our system is specialized exclusively for only one specific task, fostering better system error analysis and the utilization of reasoning and instruction following capabilities of state-of-the-art LLMs. This design also enables effective and efficient decor editing, where only a specified agent or group of agents is required to take part in an editing task, which can be asset adding, removing, repositioning, or style changing. In summary, our main contributions are:

- We present FurniMAS, an LLM-based multi-agent system effectively tackling the task of furniture decoration.
- We perform intensive experiments to show that our method significantly surpasses other baselines in generating diverse and high-quality decor results.
- We additionally demonstrate the usefulness of our method in several decor editing tasks.

2 Related Works

3D Interior Design. Many previous works have explored the task of 3D indoor design. Traditional methods typically learn spatial knowledge priors from 3D scene databases (Ma et al., 2018; Tang et al., 2024a; Tan et al., 2019; Wang et al., 2021; Wei et al., 2023; Lin and Mu, 2024). Given limitations of learning from datasets of limited categories and other asset characteristics, recent works have employed LLMs for the design process, obtaining more diverse and pleasing outcomes (Yang et al., 2024b; Feng et al., 2024; Littlefair et al., 2025; Fu et al., 2024; Bai et al., 2025; Gu et al., 2025; Sun et al., 2025). However, all of these methods still share a common limitation, which is the sole focus on room-level design, primarily addressing the arrangement of furniture within a room. In contrast, the more nuanced task of decorating individual pieces of furniture is often overlooked or insufficiently addressed. This oversight leads to sparse design outcomes, creating a significant gap between these methods and real-world scenarios, where furniture is densely adorned with various assets. Thus, our FurniMAS system represents a necessary step toward creating more realistic AI-generated furniture decoration.

LLMs for Design. Large Language Models are large computational models trained on a huge amount of data, originally designed for natural language processing problems (Achiam et al., 2023; Team et al., 2023; Dubey et al., 2024). However, with extensive commonsense understandings and exceptional capabilities in reasoning (Wei et al., 2022) and layout understanding (Li et al., 2024b), LLMs have also been widely used in designing graphics (Huang et al., 2024), user interfaces (Liu

et al., 2024; Lu et al., 2023), slides (Mondal et al., 2024), and analog layouts (Liu et al., 2025). In particular, (Lu et al., 2023) explore the LLM's ability in generating mobile user interface through in-context learning. (Huang et al., 2024) leverage LLM to synthesize information graphic designs. Despite promising results obtained in these tasks, the use of LLMs in furniture decoration remains largely unexplored. In this paper, we investigate and demonstrate that LLMs excel at understanding and reasoning about decoration-related concepts, such as layout organization and style suggestions, thus empowering an effective decoration system.

LLM-Based Multi-Agent System. Recent years have witnessed the emergence of LLM-based multi-agent systems, which have demonstrated their superiority in tasks involving complicated interactions and collaboration among many elements (Guo et al., 2024; Wang et al., 2024a). For instance, several works successfully employed LLM-based multi-agent systems for the task of software development (Hong et al., 2024; Qian et al., 2024; Nguyen et al., 2025), reinforcement learning (Zhang et al., 2024a; Zhu et al., 2025), medical diagnosis (Tang et al., 2024b; Li et al., 2024a), and recommendation systems (Wang et al., 2025; Zhang et al., 2024b). A recent method introduced in (Çelen et al., 2024) presented promising initial results in developing a multi-agent framework for personalized interior design. This suggests a promising direction for applying multi-agent systems to other challenging computer vision problems. Our FurniMAS framework builds on this philosophy, focusing specifically on the task of language-guided furniture decoration.

3 FurniMAS

We propose FurniMAS, an LLM-based multi-agent system for the task of automatically decorating furniture items following human requirements. The overview of our system is indicated in Figure 2. The input to our system consists of an empty furniture item (e.g., a bedside table, a TV stand, a working desk, etc.) represented as a triangular mesh, a natural language human requirement, and a positive integer number N_{assets} specifying the number of desired assets to decorate the given piece of furniture. The system's output is a decorated scene that aligns with the user's preferences.

FurniMAS follows a multi-stage structure, with each stage handling a specific step in the decoration workflow. We leverage AutoGen (Wu et al., 2023) to construct our multi-agent system. Our LLM agents utilize OpenAI GPT-4o (Hurst et al., 2024) as their backbone due to this model's exceptional concept understanding and reasoning skills and, more importantly, its fast responding speed and cost-efficiency. FurniMAS is a hybrid combination of several LLM-based and heuristic agents working together. In each stage involving an LLM agent, the communication is moderated by a corresponding `Admin`. Importantly, we maintain a heuristic `Validator` agent in each stage to validate responses from other LLM agents (i.e., `Asset Selector`, `Stylist`, `Planner`) and make continuous revision requests whenever needed, e.g., when an agent makes suggestion for less than N_{assets} assets or when JSON responses do not follow the predefined JSON schemas. A stage is completed only when the suggestion made by the LLM agent successfully passes the corresponding `Validator`'s validation. This design approach enables effective control over the outputs of our LLM agents, eliminating the potential risks of hallucinations or reasoning failures.

Surface Extraction. As a piece of furniture can have several supporting surfaces on which assets can be placed, we first implement heuristic algorithms to extract detailed features of such surfaces, including the vertices of the surfaces' polygons, their areas, and their heights. These features are critical for later steps in our FurniMAS system. For example, determining the sizes of assets and the number of assets to place on each surface requires information about the surface area, and positioning assets on a specific surface requires information about the surface polygon.

Asset Selection. The next step in our system is to select a set of N_{assets} appropriate assets to decorate the furniture item. The task is primarily taken care of by an LLM-based `Asset Selector` agent, with feedback given by the `Asset Validator`. The information of a suggested asset is required to include the asset name (e.g., *"gaming keyboard"*, *"electric teapot"*, *"car toy"*, *"vase of sunflower"*, *"decorative box"*), its 3D bounding box size in centimeters, and the index of the surface to place it onto. The `Asset Selector` agent is encouraged to suggest a wide variety of asset names to diversify the decorative outcome, and there can be multiple instances of the same asset category. The agent is required to populate every surface if possible, with the total area occupied by assets on each surface

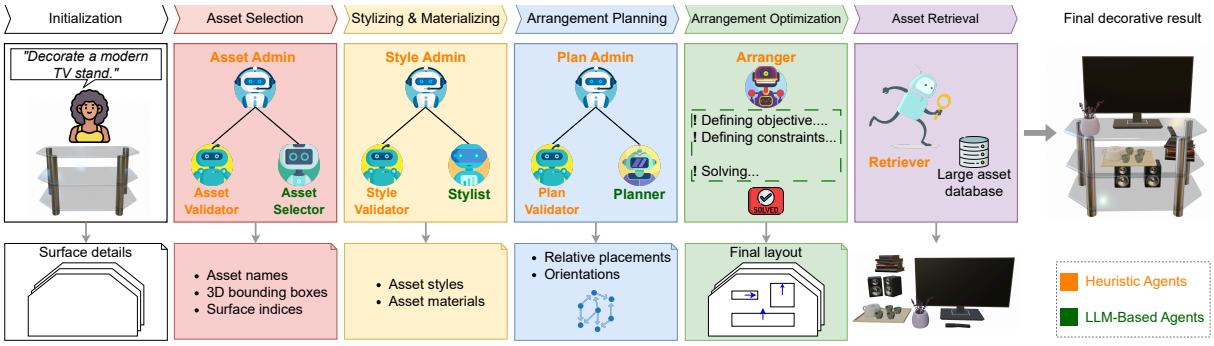

Figure 2: Overview of our FurniMAS multi-agent system. Through multiple stages of multi-agent collaboration, FurniMAS transforms the initial piece of furniture and user requirement into the final decorative outcome.

proportional to its area. Besides suggesting a few large assets that define the overall decorative concept, `Asset Selector` is prompted to include several small assets to improve the decorative details. In addition to aesthetics, we prompt the agent to make sure its suggestion ensures the functionality of the decorative outcome, which has to include essential assets that serve the user's practical purpose. In addition to checking the number of suggested assets and the JSON format of `Asset Selector`'s responses, in this stage, the `Asset Validator` also performs other crucial validations, such as ensuring all surfaces are populated and that no asset's bounding box width and length exceed the maximum width and length of its surface.

Stylizing and Materializing. Style and material are critical aspects of decoration, particularly in fulfilling aesthetic requirements. Therefore, we introduce a separate LLM-based `Stylist` agent responsible for suggesting appropriate styles and materials for assets recommended by the `Asset Selector`. In particular, the `Stylist` receives information about the human preference and N_{assets} from the asset selection stage and then suggests suitable styles (*"modern"*, *"traditional"*, *"vintage"*, etc.), and materials (*"wood"*, *"bronze"*, *"plastic"*, etc.) for every asset. A key requirement for the `Stylist` is to ensure consistency across all selected assets, aligning them with the user's aesthetic preferences. The `Stylist` is constrained to make selections from this predefined bank, enforced through targeted prompting and validation by `Style Validator` agent. This approach ensures that the final outcome adheres to a structured, coherent, and aesthetically pleasing design.

Arrangement Planning. After the stage of style and material selection, our system continues to perform the arrangement planning task. Particularly, in this stage, an LLM-based `Planner` agent is required to recommend the relative placements and orientations for every proposed asset, supervised by `Plan Validator`. The relative placement of an asset can be specified either in relation to its surface (global placement, one of the 3×3 divided regions) or in relation to another asset on the same surface (local placement). When referencing another asset, the placement can indicate relative position (i.e., *"left of"*, *"right of"*, *"in front of"*, *"behind"*, or *"on top of"*) and/or distance (i.e., *"near"* or *"far"*), and/or alignment (i.e., *"vertical-right"*, *"vertical-mid"*, *"vertical-left"*, *"horizontal-front"*, *"horizontal-mid"*, or *"horizontal-back"*). Regarding the orientation, the choices can be either *"forward"*, *"backward"*, *"left"*, or *"right"*. Besides the aesthetics, the `Planner` is prompted to make suggestions satisfying the functionality of the overall design. For example, given three assets of *"monitor"*, *"keyboard"*, and *"computer mouse"* on the same surface, the agent should suggest placing the *"monitor"* at the center of the surface, placing *"keyboard"* in front of the *"monitor"*, and placing *"computer mouse"* to the right of and near the *"monitor"*. The output of this stage characterizes a scene graph of the final decor.

Arrangement Optimization. The next step of our FurniMAS is to realize the scene graph from the arrangement planning stage, i.e., to find the exact positions and orientations for every asset. We design a heuristic `Arranger` agent to perform this task. This agent mathematically formulates the task as a constrained optimization problem. Particularly, variables are defined as the positions and orientations of assets. We define the objective function as the total satisfaction of soft constraints, which are defined based on the assets' local distance and alignment placements. Besides, hard constraints

are defined to avoid asset collisions, ensure all assets are within the surface polygons, and satisfy all global placements and local placements other than distance and alignment ones, which are, as mentioned, used for the objective function. Specifically, the `Arranger` aims to solve the following problem:

$$\text{maximize} s\left(\mathbf{p}, \mathbf{O}; \mathbf{S}\right) \triangleq \sum_{(i,j)\in\mathcal{L}^s} s_{ij}\left(\mathbf{p}_i, \mathbf{O}_i, \mathbf{p}_j, \mathbf{O}_j\right) \tag{1}$$

$$\begin{aligned} \text{s.t. } & \hat{h}_i\left(\mathbf{p}_i, \mathbf{O}_i; \mathbf{S}_i\right) \geqslant 0, \quad \forall i \in \mathcal{A}, \\ & \bar{h}_{ij}\left(\mathbf{p}_i, \mathbf{O}_i, \mathbf{p}_j, \mathbf{O}_j\right) \geqslant 0, \quad \forall i, j \in \mathcal{A}, \mathbf{S}_i = \mathbf{S}_j, \\ & \check{h}_i\left(\mathbf{p}_i, \mathbf{O}_i; \mathbf{S}_i\right) \geqslant 0, \quad \forall i \in \mathcal{G}, \\ & \tilde{h}_{ij}\left(\mathbf{p}_i, \mathbf{O}_i, \mathbf{p}_j, \mathbf{O}_j\right) \geqslant 0, \quad \forall (i,j) \in \mathcal{L}^h. \end{aligned} \tag{2}$$

Here $\mathcal{A}$ is the set of all assets. The set $\mathcal{L}^s \subseteq \mathcal{A} \times \mathcal{A}$ includes pairs of assets whose local distance or alignment placements are defined with respect to each other. $\mathcal{L}^h \subseteq \mathcal{A} \times \mathcal{A}$ represents pairs of assets whose local placements of other types are defined in relation to each other. $\mathcal{G} \subseteq \mathcal{A}$ is the set of assets that have `Planner`'s suggestions for global placements. $\mathbf{p}_i \in \mathbf{p}$, $\mathbf{O}_i \in \mathbf{O}$, and $\mathbf{S}_i \in \mathbf{S}$ are the position, orientation, and the surface polygon of object i, respectively. Note that in our optimization problem, $\mathbf{p}$ and $\mathbf{O}$ are variables and $\mathbf{S}$ are constants known after the surface extraction stage. s_{ij} are functions measuring the satisfaction of soft constraints defined by local distance and alignment placement relations between pairs of assets i and j. Besides, $\hat{h}_i$ are functions defining the hard constraints that make sure assets are within their surfaces, $\bar{h}_{ij}$ are those for hard constraints for collision avoidance among assets, $\check{h}_i$ are for global placements, and $\tilde{h}_{ij}$ are defined for local placements other than distance and alignment ones.

In our implementation, each asset position $\mathbf{p}_i$ is characterized by two variables (x_i, y_i). Note that the height value of a specific asset is fixed and equal to the height of its surface. Besides, each asset orientation $\mathbf{O}_i$ is characterized by two boolean variables (r_{90}, r_{180}), presenting rotations of 90 and 180 degrees, respectively. For example, when both r_{90} and r_{180} are true, the asset is applied a 270-degree rotation. To solve the defined optimization problem, our `Arranger` employed Gurobi solver (Gurobi Optimization, LLC, 2024). The output of the arrangement optimization stage is a detailed layout that fully describes the final positions and orientations of all assets.

Asset Retrieval. The final step is retrieving assets based on suggestions made by previous agents and finalizing the decoration process. We design a heuristic `Retriever` agent utilizing OpenShape (Liu et al., 2023) to perform text-based 3D asset retrieval from the Objaverse dataset (Deitke et al., 2023). The textual queries are formed by proposed asset names, styles, and materials. To diversify the outcomes, for a text query, the `Retriever` randomly selects one asset among the top 10 candidates most semantically similar to the text embedding. After successfully retrieving the assets, the final decorative scenes are constructed using the layout determined throughout the arrangement optimization stage. Note that, in addition to retrieving assets from a database, our system is also fully compatible with any text-to-3D generative model.

4 Experiments

In this section, we conduct several experiments to evaluate the effectiveness of our proposed FurniMAS system both quantitatively and qualitatively. We also demonstrate the usefulness of our method in various decorative editing tasks.

4.1 Language-Guided Furniture Decoration

Baselines and Setup. To the best of our knowledge, there are no previous works that directly tackle the task of language-guided furniture decoration. We therefore adapt related methods of LLM-powered interior design as baselines in our experiments. In particular, we compare our FurniMAS with adapted versions of LayoutGPT (Feng et al., 2024), I-Design (Çelen et al., 2024), and Holodeck (Yang et al., 2024b). To ensure a fair comparison, we solely use GPT-4o (Hurst et al., 2024) for all baselines. For visual quality evaluation, we visualize and render 4 images (i.e., left side view, right side view, front view, and back view) of the decor results using Blender (Team, 2025). We use 200 furniture items in our experiment, the items are of various household categories and sourced from Sketchfab (Sketchfab, 2025). To ensure a comprehensive comparison, we conduct our experiment with an increasing number of assets, (N_{assets}), starting from 8, 16, to 32.

Evaluation Metrics. We evaluate all the methods using six metrics. In particular, following (Çelen et al., 2024), we use Out-of-Boundary Rate (OOB) and Bounding Box Loss (BBL) to evaluate the feasibility of the decorative outcomes. The OOB score measures the proportion of decorative scenes in which at least one asset is placed out-

Table 1: Language-guided furniture decoration results.

	8 assets				16 assets				32 assets			
	LayoutGPT	Holodeck	I-Design	Ours	LayoutGPT	Holodeck	I-Design	Ours	LayoutGPT	Holodeck	I-Design	Ours
↓ $OOB_{\%}$	0.72 (-0.72)	0.00 (-0.00)	0.00 (-0.00)	**0.00**	0.83 (-0.83)	0.00 (-0.00)	0.00 (-0.00)	**0.00**	1.00 (-1.00)	0.00 (-0.00)	0.00 (-0.00)	**0.00**
↓ BBL_{m^2}	0.30 (-0.30)	0.00 (-0.00)	0.19 (-0.19)	**0.00**	0.41 (-0.41)	0.00 (-0.00)	0.32 (-0.32)	**0.00**	1.14 (-1.14)	0.00 (-0.00)	0.88 (-0.88)	**0.00**
↑ Func	6.30 (+1.46)	6.65 (+1.11)	7.11 (+0.65)	**7.76**	6.32 (+1.68)	6.61 (+1.39)	7.39 (+0.61)	**8.00**	5.18 (+1.78)	5.37 (+1.59)	6.12 (+0.84)	**6.96**
↑ Layout	4.05 (+3.21)	6.87 (+0.39)	6.84 (+0.42)	**7.26**	3.20 (+3.94)	6.34 (+0.80)	6.65 (+0.49)	**7.14**	2.82 (+3.12)	4.31 (+1.63)	4.35 (+1.59)	**5.94**
↑ Scheme	4.19 (+4.65)	7.21 (+1.63)	7.63 (+1.21)	**8.84**	4.02 (+4.70)	7.35 (+1.37)	7.90 (+0.82)	**8.72**	3.09 (+4.67)	5.13 (+2.63)	5.00 (+2.76)	**7.76**
↑ Atmos	4.91 (+3.43)	7.05 (+1.29)	7.21 (+1.13)	**8.34**	4.37 (+3.93)	6.52 (+1.78)	7.13 (+1.17)	**8.30**	4.00 (+3.02)	5.30 (+1.72)	5.58 (+1.44)	**7.02**

side its designated surface, while the BBL score evaluates the average intersection volume of asset bounding boxes across all decorated outcomes. In addition, following previous works on 3D content generation (Wu et al., 2024; Li et al., 2024c; Khan et al., 2025; Çelen et al., 2024), we leverage GPT-4o as a human-aligned evaluator to grade the decorative scenes generated by the baselines from 1 to 10 based on their Blender renders. The rating criteria include four key aspects: Functionality (Func), which assesses whether the selected assets and their arrangement fulfill the intended practical purpose and meet user needs; Layout and Organization (Layout), which evaluates the spatial arrangement and positioning of assets on the furniture item; Style Scheme and Material (Scheme), which examines how well the styles and materials of the assets harmonize with each other and the given furniture item; and Aesthetic and Atmosphere (Atmos), which considers the overall visual appeal and the ambiance created by the decoration. The explanation of every aspect is provided to GPT-4o, and the model is required to give reasons for its grading. To obtain a stable evaluation result, we request GPT-4o to grade each decor scene 10 times on all four aspects and then compute the average scores. The final grade of a method is averaged across all of its generated scenes.

Quantitative Results. Experimental results are shown in Table 1. It can be observed that our method is significantly better than other methods. FurniMAS ensures the physical plausibility of its generated scenes (with zero OOB and BBL) by enforcing hard constraints in the constrained optimization problem in the arrangement optimization stage. Similarly, the asset positioning search method in Holodeck (Yang et al., 2024b) provides the same capability. While I-Design achieves zero OOB, it does not guarantee collision-free asset placement. On GPT-4o's metrics, FurniMAS also demonstrates its superior performance, consistently achieving the highest scores across all aspects and asset quantity settings. Overall, all methods show a decline in performance as the number of assets increases. The highest scores for Layout, Scheme, and Atmos are generally observed with 8 assets, while Func peaks at 16 assets. We hypothesize that 16-asset scenarios provide sufficient items to meet practical user needs without excessive clutter that would hinder functionality. More cluttered scenes with 32 assets pose the greatest challenge for all methods. However, even in this case, our method delivers commendable results, whereas all other methods perform at a subpar to average level.

Qualitative Results. We present the qualitative results of all methods in generating decorative scenes following textual descriptions in Figure 3. The results indicate that our FurniMAS system exhibits a significantly stronger ability to create aesthetically pleasing scenes that align with user functionality needs and design preferences.

4.2 Open-Vocabulary Decorative Editing

An essential feature of an effective decoration method is its ability to perform decorative editing given an available decorated scene based on human instructions. In this section, we further assess the adaptability of all methods to the task of language-driven decorative editing. We modify the workflows of all baseline methods to accommodate this setting. Our experiment is based on four types of editing tasks: inserting/removing assets, changing assets, resizing assets, and rotating/repositioning assets. We conduct our experiment on 200 decorated scenes, applying all four types of editing tasks to randomly selected assets in each scene. In addition to using the four aforementioned GPT-4o metrics to evaluate the edited decorative outcomes, we introduce an additional metric, called Edit, also based on GPT-4o, to assess the level of editing task fulfillment. The results shown in Table 2 indicate that our proposed FurniMAS outperforms other baselines by a large margin on all five metrics, showcasing its usefulness in decor editing tasks.

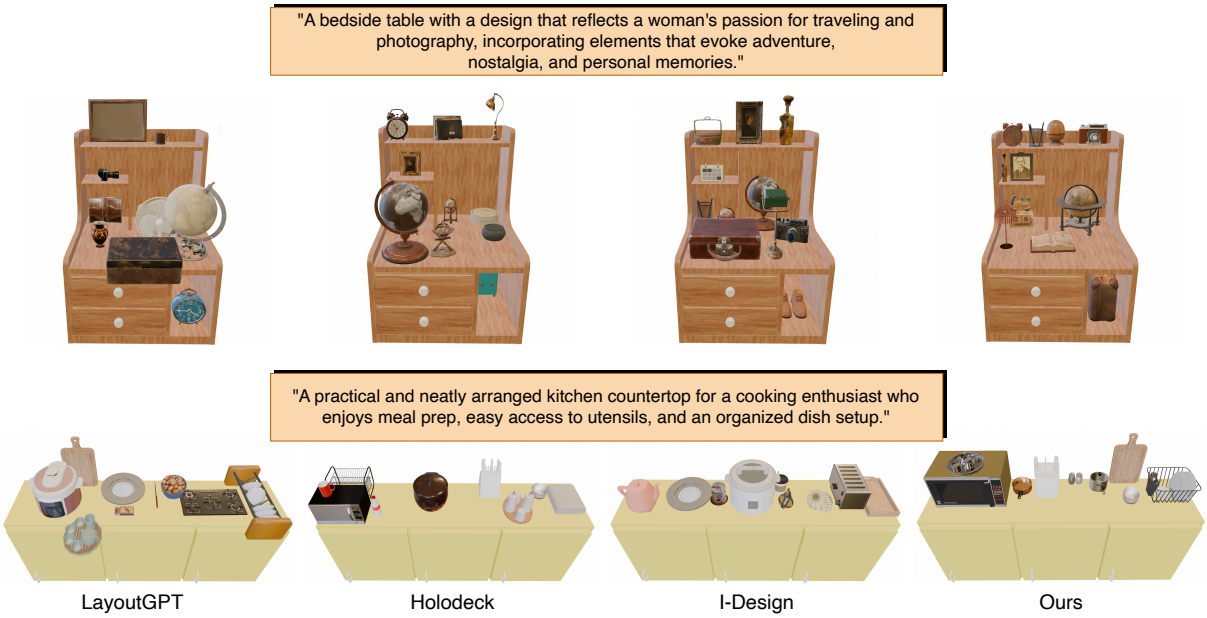

Figure 3: Furniture decoration qualitative results.

Table 2: Decorative editing results.

	LayoutGPT	Holodeck	I-Design	Ours
↑ Func	6.30 (+1.64)	6.48 (+1.46)	7.13 (+0.81)	**7.94**
↑ Layout	4.17 (+2.90)	6.15 (+0.92)	6.47 (+0.60)	**7.07**
↑ Scheme	3.76 (+4.76)	6.82 (+1.70)	7.63 (+0.89)	**8.52**
↑ Atmos	4.55 (+3.89)	6.63 (+1.81)	7.22 (+1.22)	**8.44**
↑ Edit	6.22 (+2.09)	7.01 (+1.30)	8.00 (+0.31)	**8.31**

Table 3: Agent analysis results.

	5 agents	6 agents	7 agents (Ours)
↑ Func	7.19 (+0.38)	7.52 (+0.05)	**7.57**
↑ Layout	6.10 (+0.68)	6.53 (+0.25)	**6.78**
↑ Scheme	7.83 (+0.61)	8.01 (+0.43)	**8.44**
↑ Atmos	7.00 (+0.89)	7.22 (+0.67)	**7.89**

4.3 Agent Analysis

In this section, we further verify the design of our multi-agent system. Recall that our system consists of six heuristic agents – `System Admin`, three `Validators`, `Arranger`, and `Retriever`, and three LLM-based agents – `Asset Selector`, `Stylist`, and `Planner`. To evaluate the effectiveness of assigning the three tasks of asset selection, stylizing and materializing, and arrangement planning to separate agents, we retain all heuristic agents while constructing different combinations of LLM agents as baselines for our experiment.

The results are reported in Table 3. The 9-agent configuration represents the full FurniMAS system, where each distinct agent handles only a distinct task. The 8-agent baseline merges asset selection with stylizing and materializing into one agent, while the 7-agent baseline consolidates all three tasks into a single LLM agent. From the results, we observe that our full 9-agent FurniMAS system achieves the best performance across all metrics, followed by the 8-agent system. The performance gap between the 8-agent and 9-agent systems is smaller for Layout and Func compared to the gap for Scheme and Atmos. We hypothesize that this is because, in the 8-agent setup, arrangement planning, which has a significant impact on functionality and organization, remains handled by a dedicated agent. In contrast, stylizing and materializing, which primarily influence Scheme and Atmos scores, are merged with asset selection. However, when all three tasks are combined into a single agent in the 7-agent system, performance in Func and Layout drops significantly, as arrangement planning is no longer handled separately. The results reaffirm the effectiveness of our multi-agent system and emphasize the importance of specialization in the complex task of furniture decoration, where the highest performance is achieved when each specific task is handled by a dedicated agent.

4.4 Asset Arrangement Analysis

Since asset arrangement is a critical aspect of our problem and a key contribution of our system, we conduct an experiment to evaluate the layouts produced by the arrangement planning and optimization steps of FurniMAS, comparing them with

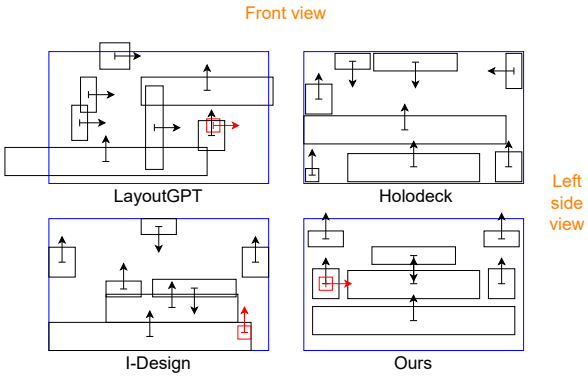

Figure 4: 2D layout projection examples. Red boxes represent assets that are arranged on top of another one.

Table 4: Results of GPT-4o evaluation.

	LayoutGPT	Holodeck	I-Design	Ours
↑ GPT-4o grade	4.21 (+3.62)	6.72 (+1.11)	6.95 (+0.88)	**7.83**

those generated by other methods. Our experiment is performed on 100 surfaces. To ensure a fair comparison, we provide each method with the same text description and asset set for every surface, applying only its respective arrangement procedures. We then use GPT-4o to assess the spatial arrangement and organization of the generated layouts based on their 2D projections. The final scores are averaged across all surfaces for each baseline. Table 4 shows that layouts generated by FurniMAS are significantly better than those generated by its counterparts, demonstrating the effectiveness of our `Planner` and `Arranger` with its arrangement algorithm. The 2D projection examples of generated layouts are presented in Figure 4.

4.5 Ablation Study

Performance on different types of furniture? We evaluate the performance of our system on four different types of household furniture, including workspace furniture, living space furniture, bedroom furniture, and kitchen furniture. Results are reported in Table 5, where FurniMAS gives the best performance on workspace furniture, with the highest scores obtained on all four metrics. Besides, the system also performs generally well on living space furniture. In contrast, bedroom and kitchen furniture items are more challenging for FurniMAS, with the lowest score obtained when decorating furniture items in the kitchen space. However, in general, the results demonstrate the generalization of our method on different types of furniture, where commendable scores are obtained in all cases.

Table 5: Results on four types of furniture.

	Workspace	Living space	Bedroom	Kitchen
↑ Func	**7.73**	7.62	7.41	7.54
↑ Layout	**7.02**	6.93	6.70	6.69
↑ Scheme	**8.75**	8.49	8.33	8.37
↑ Atmos	**8.21**	7.86	7.95	7.60

Table 6: Results on different LLM backbones.

	GPT-4	GPT-4.5	GPT-5	GPT-4o (Ours)
↑ Func	7.66	**7.70**	7.52	7.57
↑ Layout	6.65	6.92	**7.04**	6.78
↑ Scheme	8.17	8.31	8.40	**8.44**
↑ Atmos	7.94	8.16	**8.22**	7.89
↓ Time_{min}	2.50	2.29	2.18	**1.53**
↓ $\text{Cost}_{\$}$	0.46	0.94	0.16	**0.06**

Performance with different LLMs. We evaluate the performance of our system using various LLM backbones, including GPT-4o, GPT-4, GPT-4.5, and GPT-5 (OpenAI, 2025). To provide a comprehensive comparison, we additionally report the average API call costs and the average time required to complete the three stages – asset selection, stylizing and materializing, and arrangement planning – that involve LLM agents. These metrics are averaged over 200 decoration tasks for each LLM. Our CPU configuration is AMD EPYC 7742 64-core. Results are shown in Table 6, where GPT-5 generally obtains the best performance on four quality metrics, followed by GPT-4.5. This aligns with their better understanding and reasoning skills compared to other models. However, these models are considerably slow due to their heavy reasoning processes, and GPT-4.5 is a very high-cost model. GPT-4o, in contrast, offers the best balance. It delivers competitive quality scores (with the highest score in Scheme), has the fastest execution time, and is the most cost-efficient option. The results further validate our decision to use GPT-4o as the backbone of our multi-agent system.

5 Conclusion

We propose FurniMAS, a multi-agent system powered by LLMs that generates aesthetically pleasing and preference-aligned decorative scenes from pieces of household furniture. We demonstrate the superiority of FurniMAS in several furniture decoration experiments. We hope that our promising results will inspire further research on the challenging problem of decoration, ultimately benefiting a wide range of real-world applications.

6 Limitations

Despite its strong performance across many furniture decoration tasks, our FurniMAS system still has room for improvement. First, the results showed that the performance of our method drops considerably when the number of assets increases intensively. An improvement in the asset arrangement algorithm or implementing an adaptive asset selection method based on available surface area and balance between functionality and minimalism is promising to better address this challenge. In addition, our system currently supports only basic asset placement operations. While this covers most decoration scenarios, real-world applications may require more complex functions such as hanging or draping. Expanding support for these operations would enhance the system's versatility and better meet user needs. Finally, while automation in furniture decoration significantly reduces labor and speeds up the process, allowing human-in-the-loop feedback would be a crucial feature. This capability would enhance customization, facilitate better user satisfaction, and allow for iterative refinements based on personal preferences. We leave these enhancements for future development.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint*.

Tongyuan Bai, Wangyuanfan Bai, Dong Chen, Tieru Wu, Manyi Li, and Rui Ma. 2025. Freescene: Mixed graph diffusion for 3d scene synthesis from free prompts. In *CVPR*.

Ransome Epie Bawack, Samuel Fosso Wamba, Kevin Daniel André Carillo, and Shahriar Akter. 2022. Artificial intelligence in e-commerce: a bibliometric study and literature review. *Electronic markets*.

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. 2024. I-design: Personalized llm interior designer. In *ECCV Workshop CV4Metaverse*.

Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. 2020. Learning generative models of 3d structures. In *Computer Graphics Forum*.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2023. Objaverse: A universe of annotated 3d objects. In *CVPR*.

Wei Deng, Mengshi Qi, and Huadong Ma. 2025. Global-local tree search in vlms for 3d indoor scene generation. In *CVPR*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint*.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2024. Layoutgpt: Compositional visual planning and generation with large language models. *NeurIPS*, 36.

Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. 2024. Anyhome: Open-vocabulary generation of structured and textured 3d homes. In *ECCV*.

Zeqi Gu, Yin Cui, Zhaoshuo Li, Fangyin Wei, Yunhao Ge, Jinwei Gu, Ming-Yu Liu, Abe Davis, and Yifan Ding. 2025. Artiscene: Language-driven artistic 3d scene generation through image intermediary. In *CVPR*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *IJCAI*.

Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. https://www.gurobi.com/.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *ICLR*.

Qirui Huang, Min Lu, Joel Lanir, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2024. Graphimind: Llm-centric interface for information graphics design. *arXiv preprint*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint*.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. 2025. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. *NeurIPS*.

Vikram Kumaran, Jonathan Rowe, Bradford Mott, and James Lester. 2023. Scenecraft: automating interactive narrative scene generation in digital games with large language models. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Sangmin Lee, Sungyong Park, and Heewon Kim. 2025. Dynscene: Scalable generation of dynamic robotic manipulation scenes for embodied ai. In *CVPR*.

Junkai Li, Siyu Wang, Meng Zhang, Weitao Li, Yunghwei Lai, Xinhui Kang, Weizhi Ma, and Yang Liu. 2024a. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint*.

Weiming Li, Manni Duan, Dong An, and Yan Shao. 2024b. Large language models understand layout. *arXiv preprint*.

Xiao-Lei Li, Haodong Li, Hao-Xiang Chen, Tai-Jiang Mu, and Shi-Min Hu. 2024c. Discene: Object decoupling and interaction modeling for complex scene generation. In *SIGGRAPH Asia 2024 Conference Papers*.

Chenguo Lin and Yadong Mu. 2024. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. In *ICLR*.

Gabrielle Littlefair, Niladri Shekhar Dutt, and Niloy J Mitra. 2025. Flairgpt: Repurposing llms for interior designs. In *Computer Graphics Forum*.

Bingyang Liu, Haoyi Zhang, Xiaohan Gao, Zichen Kong, Xiyuan Tang, Yibo Lin, Runsheng Wang, and Ru Huang. 2025. Layoutcopilot: An llm-powered multi-agent collaborative framework for interactive analog layout design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. 2023. Openshape: Scaling up 3d shape representation towards open-world understanding. *NeurIPS*.

Yimeng Liu, Misha Sra, and Chang Xiao. 2024. Crowdgenui: Enhancing llm-based ui widget generation with a crowdsourced preference library. *arXiv preprint*.

Yuwen Lu, Ziang Tong, Qinyi Zhao, Chengzhi Zhang, and Toby Jia-Jun Li. 2023. Ui layout generation with llms guided by ui grammar. *ICML Workshop on AI and HCI*.

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

Ishani Mondal, S Shwetha, Anandhavelu Natarajan, Aparna Garimella, Sambaran Bandyopadhyay, and Jordan Boyd-Graber. 2024. Presentations by the humans and for the humans: Harnessing llms for generating persona-aware slides from documents. In *EACL*.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. 2024. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *RSS Workshop: Data Generation for Robotics*.

Minh Huynh Nguyen, Thang Phan Chau, Phong X Nguyen, and Nghi DQ Bui. 2025. Agilecoder: Dynamic collaborative agents for software development based on agile methodology. In *ACM International Conference on AI Foundation Models and Software Engineering*.

OpenAI. 2025. Openai models. https://platform.openai.com/docs/models.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of The 36th Annual ACM Symposium on User Interface Software and Technology*.

Akshay Gadi Patil, Supriya Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. 2024. Advances in data-driven analysis and synthesis of 3d indoor scenes. In *Computer Graphics Forum*.

Pierre Peigné, Mikolaj Kniejski, Filip Sondej, Matthieu David, Jason Hoelscher-Obermaier, Christian Schroeder de Witt, and Esben Kran. 2025. Multi-agent security tax: Trading off security and collaboration capabilities in multi-agent systems. In *AAAI*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. In *ACL*.

Inc. Sketchfab. 2025. Sketchfab. https://sketchfab.com/feed.

Lei Song, Chuheng Zhang, Li Zhao, and Jiang Bian. 2023. Pre-trained large language models for industrial control. *arXiv preprint*.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. 2025. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *CVPR*.

Fuwen Tan, Song Feng, and Vicente Ordonez. 2019. Text2scene: Generating compositional scenes from textual descriptions. In *CVPR*.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. 2024a. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *CVPR*.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024b. Medagents: Large language models as collaborators for zero-shot medical reasoning. In *ACL*.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, and 1 others. 2024. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint*.

Blender Team. 2025. Blender. https://www.blender.org/.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*.

Mingze Wang, Chongming Gao, Wenjie Wang, Yangyang Li, and Fuli Feng. 2025. Tunable llm-based proactive recommendation agent. In *ACL*.

Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. 2021. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. 2024b. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. In *ICML*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022. Emergent abilities of large language models. *arXiv preprint*.

Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. 2023. Lego-net: Learning regular rearrangements of objects in rooms. In *CVPR*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint*.

Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. 2024. Gpt-4v (ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*.

Yuchen Xia, Manthan Shenoy, Nasser Jazdi, and Michael Weyrich. 2023. Towards autonomous system: flexible modular production system enhanced with large language model agents. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Yunqiu Xu, Linchao Zhu, and Yi Yang. 2024. Gg-editor: Locally editing 3d avatars with multimodal large language model guidance. In *ACM Multimedia 2024*.

Qisen Yang, Zekun Wang, Honghui Chen, Shenzhi Wang, Yifan Pu, Xin Gao, Wenhao Huang, Shiji Song, and Gao Huang. 2024a. Psychogat: A novel psychological measurement paradigm through interactive fiction games with llm agents. In *ACL*.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, and 1 others. 2024b. Holodeck: Language guided generation of 3d embodied ai environments. In *CVPR*.

Zeyuan Yang, Jiageng Liu, Peihao Chen, Anoop Cherian, Tim K Marks, Jonathan Le Roux, and Chuang Gan. 2024c. Rila: Reflective and imaginative language agent for zero-shot semantic audio-visual navigation. In *CVPR*.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. 2024a. Building cooperative embodied agents modularly with large language models. *ICLR*.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024b. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM on Web Conference 2024*.

Xi Zhang, Jiangtao Du, and David Chow. 2023. Association between perceived indoor environmental characteristics and occupants' mental well-being, cognitive performance, productivity, satisfaction in workplaces: A systematic review. *Building and Environment*.

Guobin Zhu, Rui Zhou, Wenkang Ji, and Shiyu Zhao. 2025. Lamarl: Llm-aided multi-agent reinforcement learning for cooperative policy generation. *IEEE RA-L*.

A FurniMAS's Broader Application

While we demonstrated the effectiveness of our system in designing furniture decorative outcomes, we believe that FurniMAS has many other downstream applications. Firstly, while Holodeck (Yang et al., 2024b) benefits robot navigation, FurniMAS can be used to automatically generate cluttered scenes for robot manipulation tasks (Nasiriany et al., 2024; Tao et al., 2024). In addition, our system can also enhance environment construction in game development (Kumaran et al., 2023) and improve e-commerce through automated product visualizations (Bawack et al., 2022).

B Bank of Styles and Materials

We detail the bank of styles and materials that constrain the `Stylist`'s suggestions. There are 31 style options, which are *"Contemporary"*, *"Coastal"*, *"Scandinavian"*, *"Shabby Chic"*, *"Transitional"*, *"Modern"*, *"Mid-century"*, *"Retro"*, *"Minimalist"*, *"Traditional"*, *"Farmhouse"*, *"Antique"*, *"Industrial"*, *"Rustic"*, *"Vintage"*, *"Mission"*, *"French"*, *"Art Deco"*, *"Victorian"*, *"Chippendale"*, *"Country"*, *"Craftsman"*, *"Shaker"*, *"Queen Anne"*, *"Hepplewhite"*, *"Louis XVI"*, *"Asian"*, *"Jacobean"*, *"Colonial"*, *"Federal"*, and *"Sheraton"*. Regarding materials, there are 17 options, which are *"wood"*, *"plywood"*, *"marble"*, *"paper"*, *"fibre"*, *"plastic"*, *"glass"*, *"textile"*, *"iron"*, *"steel"*, *"gold"*, *"silver"*, *"bronze"*, *"cotton"*, *"linen"*, *"leather"*, and *"paper"*. Our bank comprises widely recognized and commonly used styles and materials in professional design and decoration.

C Handling Rugged Surfaces

In real-world furniture items, supporting surfaces are not always completely flat. For that reason, we develop FurniMAS so that it can handle both flat and rugged surfaces. In particular, in surface extraction, we use a height tolerance of 2 cm to allow the identification of rugged surfaces. To place assets in Blender, we employ a drop operator to make sure assets conform to the surface geometry. While handling more complex surfaces could further improve the generalizability and is an interesting future improvement, we find that the current design allows us to handle most of the cases in decorating furniture.

D Relative Placement Demonstrations

Figure 5 and Figure 6 showcase examples of global placement and alignment placement constraints from the arrangement planning and optimization stages to enhance understanding and visualization.

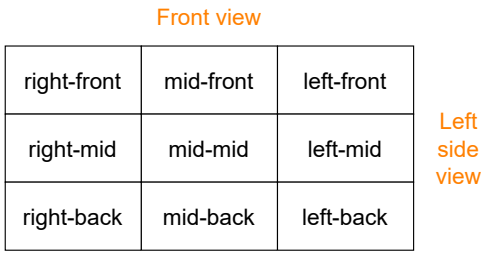

Figure 5: Global placement demonstration.

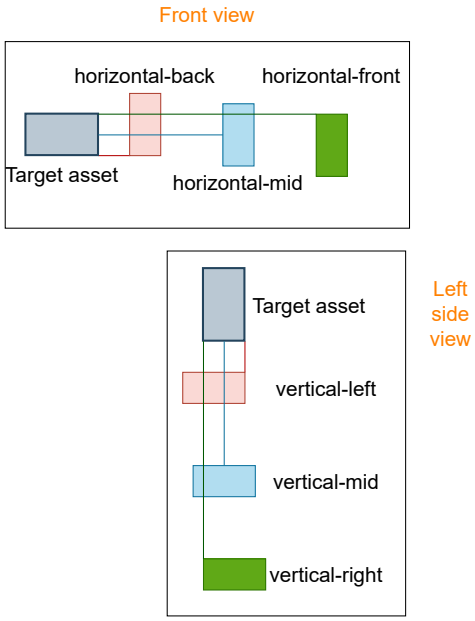

Figure 6: Alignment placement demonstration.

E Roles of heuristic Validators

`Validators` are critical in ensuring that the LLM agents' output covers a sufficient number of assets and can be processed by the next stage, and guaranteeing the feasibility of the asset arrangement. For example, without the `Asset Validator`, the Asset Selection stage could produce an output containing an asset whose size is incompatible with its surface, leading to an implausible arrangement. To validate the necessity of the `Validators`, we conduct an experiment on 100 decoration tasks and report the frequency with which the corresponding LLM agent produces an erroneous output in Table 7. The results reaffirm the effectiveness of our

method across many decoration and editing tasks. The results reaffirm the importance of heuristic `Validators` in our system design.

	Asset Selector	Stylist	Planner
Error frequency (%)	26	18	34

Table 7: Error frequency results.

F Other LLMs for Evaluation

We employ other LLMs for our evaluation to demonstrate the statistical robustness of FurniMAS's advantage over other baseline methods. The results benchmarked by GPT-4 and GPT-4.5 are reported in Table 8, where our method still largely outperforms all remaining approaches.

	LayoutGPT	Holodeck	I-Design	Ours
↑ Func	5.83/5.46	6.53/6.50	7.12/6.81	**7.60/7.33**
↑ Layout	4.44/4.29	6.31/6.47	5.45/5.07	**6.54/6.72**
↑ Scheme	3.65/3.71	6.13/5.88	7.13/7.11	**8.27/8.09**
↑ Atmos	4.20/4.01	6.44/6.09	6.78/6.52	**7.69/8.04**

Table 8: Results by other LLM evaluators.

G Compatibility with 3D Asset Generation Models

Instead of retrieving assets from Objaverse (Deitke et al., 2023), our FurniMAS system can leverage text-to-3D models to generate assets for decoration. An example of incorporating an asset generated by LumaAI [1] is shown in Figure 7. This demonstrates the flexibility of our system.

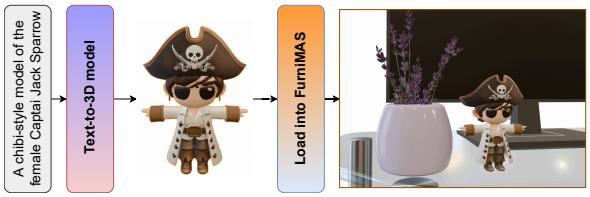

Figure 7: FurniMAS can import any 3D assets, including those generated from text-to-3D models.

H Indoor Scene Creation

While our method is developed to decorate single pieces of furniture, it can be used in conjunction with other scene layout generation approaches to construct realistic 3D indoor environments. We showcase this capability in Figure 8.

[1] https://lumalabs.ai/genie

Figure 8: FurniMAS can be used to create realistic 3D indoor scenes.

I Additional Qualitative Results

We present more decorative scenes generated by our FurniMAS system on furniture items of different categories and appearances in Figure 9. We also provide more results about the editing capability of our system in Figure 10 and Figure 11.

J System Prompts for LLM Agents

The complete prompt templates for FurniMAS's LLM agents are provided in Figures 12, 13, and 14.

K GPT-4o Evaluation

The prompt template for GPT-4o evaluation is provided in Figure 15. Note that alongside the text prompt, 4 Blender renders are provided to the model for grading. In particular, we set the rendering engine to CYCLES for high-quality image outputs. Our rendering script is adapted from InstructScene [2]. Images for evaluation are in 512×512 resolution. We maintain a camera distance of $160\,\mathrm{cm}$, a filter width of 0.5, and employ the RGB color mode.

L Details of the Experiment on Different LLMs

In the experiment with different LLM backbones, we use the following specific models: GPT-4o, GPT-4-0613, GPT-4.5-preview-2025-02-27, and GPT-5. For the first three models, to enable a fair comparison, we set all the temperature hyperparameters to 0.7. Note that this hyperparameter is not available in GPT-5. In GPT-5, we set the reasoning effort hyperparameter to "medium".

[2] https://github.com/chenguolin/InstructScene/blob/main/src/utils/blender_script.py

Figure 9: Additional qualitative results.

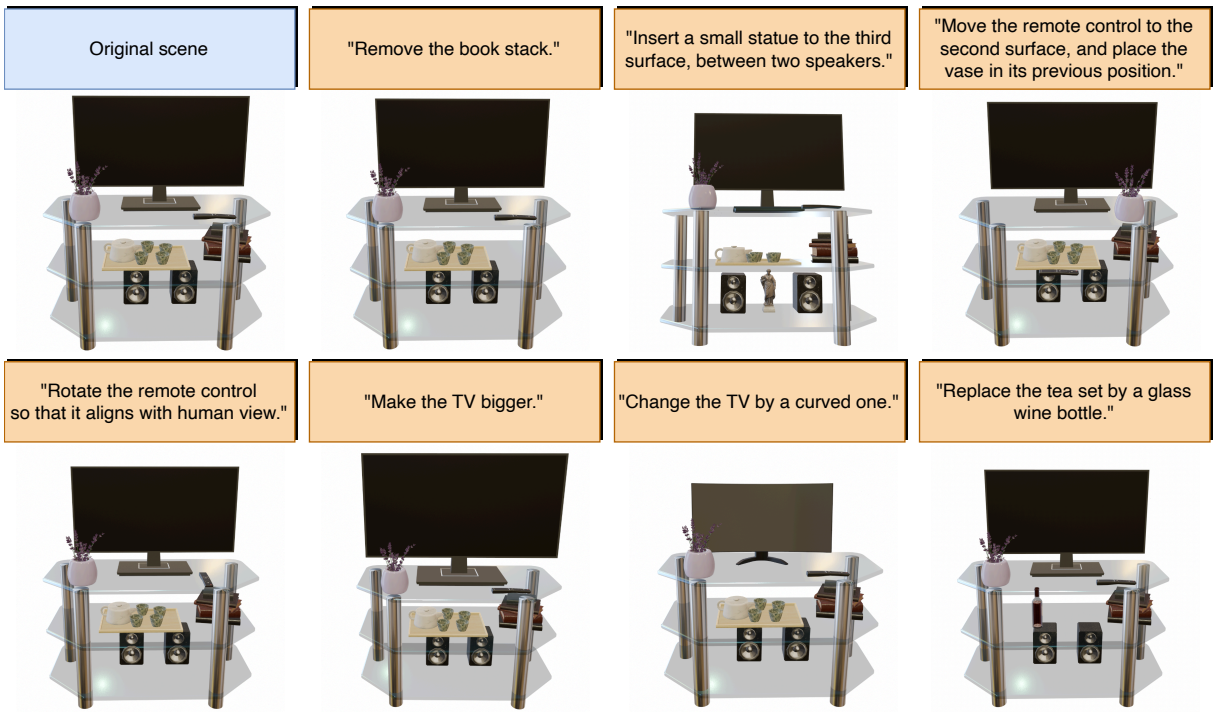

Figure 10: TV stand editing results. Each edited scene is derived from the original scene.

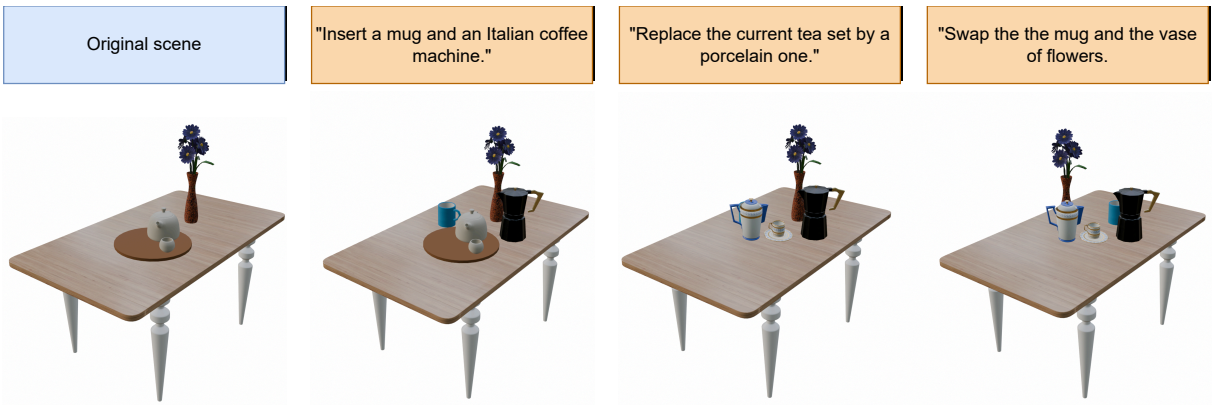

Figure 11: Coffee table continuous editing results. Each scene is derived from its immediate predecessor.

Asset Selector prompt: You are an expert Asset Selector who is taking part in a furniture decoration project. Your job is to propose assets to decorate a furniture item such as a TV stand, a table, a working desk, a bookcase, etc. You are given information about the user preference and horizontal surfaces of the item for placing other assets onto it.

The given furniture item can have several horizontal surfaces. The information about a surface consists of a height value determining its height in centimeters, a height limit value specifying the maximum allowable height in centimeters for any asset placed on it, its max length, its max width, the area of the surface in centimeters, and a list of 2D vertices that form the surface polygon.
Your job is to suggest in total {n_assets} essential assets to decorate the item.

Your suggested assets must contain the following information:
1. Asset index (starting from 0)
2. Asset name (e.g. gaming keyboard_0, flower pot_0, curved monitor_0, alarm clock_0, alarm clock_1, comic book_0, pan_0, Japanese tea pot_0 etc.)
3. Surface index (index of the surface to place the asset onto)
4. 3D bounding box size in centimeters (specifying length, width, and height e.g. (30, 20, 10))

Here are IMPORTANT NOTES you must take into account when responding:
- Besides *aesthetics*, make sure your suggestion ensures the *functionality* of the decorative outcome. There have to be essential assets that serve the practical purpose and meet the user's needs.
- Numerical values of 3D bounding boxes have to be integers.
- *No* asset on a surface can exceed the surface's height limit.
- Suggest diverse asset names. If you wish to suggest multiple instances of the same type of asset, give different instances different names, with increasing indexes, e.g., "alarm clock_0", "alarm clock_1".
- Suggest assets of varying sizes. On each surface, after suggesting *a few large assets* that define the overall decorative concept, include *several small assets* to improve the details, especially when you are required to suggest more than 6 assets.
- On each surface, you are required to suggest *at least one* landmark asset, which will be placed first. You are required to include landmark asset(s) first, before suggesting others.
- Populate *every* surface. The total area occupied by assets on each surface should be proportional to the surface's area.
- Populate surfaces *sequentially*, starting with surface 0, then 1, 2, and so on.
- Carefully consider the surface heights, max lengths, max widths, and areas for a coherent and cohesive design and to avoid collisions.
- Follow the JSON schema below:
  {asset_selector_schema}

Figure 12: System prompt for the `Asset Selector`.

Stylist prompt: You are an expert Stylist who is taking part in a furniture decoration project. You will receive information about the user preference and about {n_assets} assets (in JSON format) to place onto a furniture item. Given information about one asset includes the asset index, its name, its surface index (a furniture item can have several horizontal surfaces, assets that have the same surface index are on the same surface), and its 3D bounding box size in centimeters.
Your job is to suggest the styles and materials for all given assets in order to obtain a decorative outcome that is of high quality and aligns with the user preference.

Your suggestion for one asset must contain the following information:
1. Asset index: copied index of the asset
2. Asset name: copied name of the asset
3. Surface index: copied surface index of the asset
4. Style (e.g. Modern, Traditional, Vintage, etc.)
5. Material (e.g. wood, bronze, plastic, etc.)

Here are IMPORTANT NOTES you must take into account when responding:
- If possible, you should make the styles of assets consistent.
- Remember to make the suggestion for every asset given to you.
- Choose styles from the bank of styles:
  {style_bank}
- Choose materials from the bank of materials:
  {material_bank}
- Follow the JSON schema below:
  {stylist_schema}

Figure 13: System prompt for the `Stylist`.

Planner prompt: You are an expert Arrangement Planner who is taking part in a furniture decoration project. You will receive information about the user preference and about {n_assets} assets (in JSON format) to place onto a furniture item. In addition, you will also receive the information about max lengths and max widths of every horizontal surface of the furniture item, on which the assets can be placed.
Given information about one asset includes the asset index, its name, its surface index (assets that have the same surface index are on the same surface), its 3D bounding box size in centimeters, its style, and its material.
Your job is to determine the placement and orientation for every given asset in order to obtain a decorative outcome that is of high quality and aligns with the user preference.

The suggested placement and orientation of an asset are defined by a set of constraints. Here are the details of the types of constraints:

1. Global placement constraints:
- Constraints in this type specify the relative placement of the asset with respect to its surface. The constraint value of this type can *only* be "left-front", "mid-front", "right-front", "left-mid", "mid-mid", "right-mid", "left-back", "mid-back", and "right-back".
For example, "left-front" refers to the region located in the front-left region of the surface (from the human perspective), while "mid-mid" refers to the central region, and "right-back" refers to the region located in the back-right region.
- An example of this type of constraint is {{"type": "placement-global", "value": "mid-mid"}}.

2. Place-based local placement constraints:
- Constraints in this type specify the relative placement, in terms of place, of the asset with respect to another target asset on the same surface. The constraint value of this type can *only* be "left of", "right of", "in front of", "behind", or "on top of".
- An example of this type of constraint is {{"type": "placement-local-place", "value": "behind", "target": "remote control_1"}}.

3. Distance-based local placement constraints:
- Constraints in this type specify the relative placement, in terms of distance, of the asset with respect to another target asset on the same surface. The constraint value of this type can *only* be "near", or "far".
- An example of this type of constraint is {{"type": "placement-local-distance", "value": "near", "target": "table lamp_0"}}.

4. Orientation constraints:
- Constraints in this type specify the orientation of the asset. The constraint value of this type can *only* be "forward", "left", "right", or "backward".
For example, "forward" means the asset follows the orientation of the furniture item, which is opposite to the human's perspective, while "left" means the asset rotates to the left (from the human perspective).
- An example of this type of constraint is {{"type": "orientation", "value": "forward"}}

5. Alignment constraints:
- Constraints in this type denote the horizontal or vertical alignment between the asset and a target asset (after the rotation specified by any orientation constraint) on the same surface. The constraint value can be "vertical-left", "vertical-mid", "vertical-right", "horizontal-front", "horizontal-mid", or "horizontal-back".
Specifically,
"vertical-left" means that the left edge of the asset's bounding box is vertically aligned with the left edge of the target asset's bounding box.
"vertical-mid" means that the center of the asset's bounding box is vertically aligned with the center of the target asset's bounding box.
"vertical-right" means that the right edge of the asset's bounding box is vertically aligned with the right edge of the target asset's bounding box.
"horizontal-front" means that the front edge of the asset's bounding box is horizontally aligned with the front edge of the target asset's bounding box.
"horizontal-mid" means that the center of the asset's bounding box is horizontally aligned with the center of the target asset's bounding box.
"horizontal-back" means that the back edge of the asset's bounding box is horizontally aligned with the back edge of the target asset's bounding box.
- An example of this type of constraint is {{"type": "alignment", "value": "horizontal-mid", "target": "printer_0"}}.

Your suggestion for one asset must contain the following information:
1. Asset index: copied index of the asset
2. Asset name: copied name of the asset
3. Surface index: copied surface index of the asset
4. Constraints: list of constraints

Here are IMPORTANT NOTES you must take into account when responding:
- Make sure your suggestion ensures both *aesthetics* and *functionality* of the decorative outcome.
- For any type of constraint, use *only the mentioned constraint values* of that type.
- One orientation constraint is *required* for each asset. If there is no compelling reason, the orientation of an asset should be set to "forward".
- On any surface, there must be *at least one* landmark asset that has only a global placement constraint and an orientation constraint. Landmark assets are *not allowed* to have any alignment constraint or local placement constraint, whether place-based or distance-based.
- Carefully select landmark assets for each surface, as they define the overall layout.
- Assets that are not landmark assets are *allowed* to have global placement constraints.
- The "near" distance-based local placement constraints are *preferred* over "far" ones. Use "near" when grouping assets for functionality—for example, placing a mug near a water bottle. The "far" constraints should be used *only in special cases*, such as when creating more empty space by separating assets.
- Only use the "on top of" place-based local placement constraint in very few scenarios, e.g., placing a book on top of a decorative box.
- To ensure the aesthetics of the decorative outcome, the alignment constraints are *encouraged*.
When an asset has a place-based local placement constraint relative to a target asset, with a constraint value of "in front of" or "behind", its alignment constraint with respect to that target asset, if specified, must be a vertical constraint—i.e., "vertical-left", "vertical-mid", or "vertical-right".
When an asset has a place-based local placement constraint relative to a target asset, with a constraint value of "left of" or "right of", its alignment constraint with respect to that target asset, if specified, must be a horizontal constraint—i.e., "horizontal-front", "horizontal-mid", or "horizontal-back".
- When arranging assets on a surface, remember to take the surface max length and max width into account to avoid out-of-bound errors. For example, an asset of length 60cm and another of length 50cm can not be horizontally aligned on a surface that has max length of 100cm.
- Follow the JSON schema below:
{arrangement_planner_schema}

Figure 14: System prompt for the `Planner`.

GPT-4o evaluation prompt: You are an experienced Decoration Evaluator participating in a furniture decoration project. Given an empty furniture item and a user input, a decoration team has selected and arranged several assets on the item to create a decorative outcome. Your task is to evaluate the result.
In particular, give a grade from 1 to 10 to the following decorative 4 outcome renders (left side view, front view, right side view, and back view) based on how well they correspond together to the user input in the 4 following aspects:

1. Functionality
   - This assesses whether the chosen assets enhance the usability of the furniture item.
   - It considers whether the selected assets serve a practical purpose and meet the user's needs.
   - Example: On a working desk, a well-placed monitor, keyboard, and mouse contribute to efficient workflow, while unnecessary clutter can hinder productivity.

2. Layout and Organization
   - This evaluates the spatial organization of the assets on the furniture item.
   - Key factors include balance, spacing, and ease of access.
   - Example: On a TV stand, placing a soundbar in front of the screen might block the view, while organizing devices neatly ensures convenience and a tidy look.

3. Style Scheme and Material
   - This focuses on how well the styles and materials of the assets complement each other and the given furniture item.
   - The goal is to achieve visual harmony rather than clashing tones or textures.
   - Example: A minimalist workspace might use neutrally styled accessories with metallic or wooden finishes to maintain a sleek, modern look.

4. Aesthetic and Atmosphere
   - This considers the overall visual appeal and the mood created by the decoration.
   - A well-designed arrangement should align with the intended style and ambiance.
   - Example: A cozy reading nook on a bookshelf may include warm-toned bookends, small decorative plants, and soft lighting to create an inviting atmosphere.

Following are two examples for your response, with the user input as "A minimalistic working desk for a software developer":

```json
{high_score_example_json}
```

```json
{low_score_example_json}
```

Now, based on the provided rendered images, evaluate for the user input:

```{user_input}```

Follow the format of 2 JSON examples given to you.

Figure 15: System prompt for the GPT-4o evaluation.
```