Computers & Graphics 136 (2026) 104553

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Shape Modeling

Scene layout via conceptual design

Wenzheng Wu [a], Chucheng Xiang [a], Zhi Lin [a], Yirui Guan [b], Ruchao Bao [a], Zhongyuan Liu [b], Ziqi Wang [c], Ligang Liu [a,*]

[a] University of Science and Technology of China, China
[b] Tencent, China
[c] Hong Kong University of Science and Technology, Hong Kong, China

ARTICLE INFO

Keywords:
Conceptual design
Layout generation
CAD
Large language model
AI agents

ABSTRACT

We introduce a novel approach for designing 3D scene layouts, focusing on a reason-driven generation from text to conceptual design. Our Iterated Design System (IDS) offers a hierarchical structure that incorporates both geometric and semantic information in the scene, reflecting the intricate process of human design reasoning. We leverage Large Language Models (LLMs) to break down the complex task of conceptual design into more manageable components, mirroring human cognitive methods. Additionally, we propose an LLM-in-the-loop optimization strategy to resolve conflicting constraint challenges which often emerge in traditional geometric layout optimization. Once the traditional algorithm detects and suggests solutions for conflicts, the LLM interprets the semantics of these solutions within the design context to make the best decision. Experimental findings highlight our framework's capability to create satisfactory and complex scene layouts across various scene categories.

1. Introduction

The design of scene layout has drawn significant research in indoor design [1], urban planning [2,3], and game development [4]. Several techniques for generating scene layouts, including data-driven methods [5,6] and procedural modeling [7], have been explored to expedite the design process.

Existing methods predominantly focus on generating scene layouts based on user-defined plans, such as bubble graphs [8], which entail determining the exact placement and dimension of each element within the scene. This stage is referred to as *detail design* (Fig. 2, c). However, in practice, designing these bubble graphs is as critical as the detailed design itself. Creating these bubble graphs requires designers to analyze customer requirements and integrate them with their own insights. Designers must determine which structures need to be included in the scene and define their spatial arrangement. This process of translating textual ideas into design concepts is referred to as *conceptual design* [9,10] (Fig. 2, b). Traditionally, conceptual design is conducted manually, requiring a significant amount of time. To address this, this work focuses on developing a computational method to automate the conceptual design process.

To realize this creative process, the challenges are two-fold: Firstly, in the conceptual design phase, it is essential to rely on professional design expertise, comprehensive research, and practical experience for analyzing requirements and making informed decisions. For example, when arranging a specific bedroom scene, the designer needs to draw on common knowledge and experience of bedrooms to know the essential objects and their placement scales, while also conducting sufficient research (or making informed assumptions) about the user of the bedroom. The user's interests and living habits can significantly influence the selection and arrangement of objects in the scene. Secondly, the process of conceptual design involves intricate reasoning. For instance, the designer must synthesize various requirements, reason about which objects should be placed, further infer the relationships between those objects, and ultimately decide their final positions. Such reasoning can be supported by specific methodologies, factoring in considerations like functionality and aesthetics [9].

We utilize Large Language Models (LLMs) to address these challenges. This approach is founded on the observation that LLMs are endowed with vast human knowledge [11] and possess some level of reasoning capabilities [12,13]. Their application in domains like automatic programming and robotics demonstrates their potential for executing complex tasks [14–16]. Current methodologies [14–16] highlight that the effective application of LLMs to complex tasks lies in

* Corresponding author.
E-mail addresses: wuwzh@mail.ustc.edu.cn (W. Wu), xcc2020@mail.ustc.edu.cn (C. Xiang), lz0817@mail.ustc.edu.cn (Z. Lin), ezguan@tencent.com (Y. Guan), iambrc@mail.ustc.edu.cn (R. Bao), lockliu@tencent.com (Z. Liu), ziqiw@ust.hk (Z. Wang), lgliu@ustc.edu.cn (L. Liu).
URL: http://staff.ustc.edu.cn/~lgliu/ (L. Liu).

https://doi.org/10.1016/j.cag.2026.104553
Received 16 September 2025; Received in revised form 4 February 2026; Accepted 26 February 2026
Available online 27 February 2026

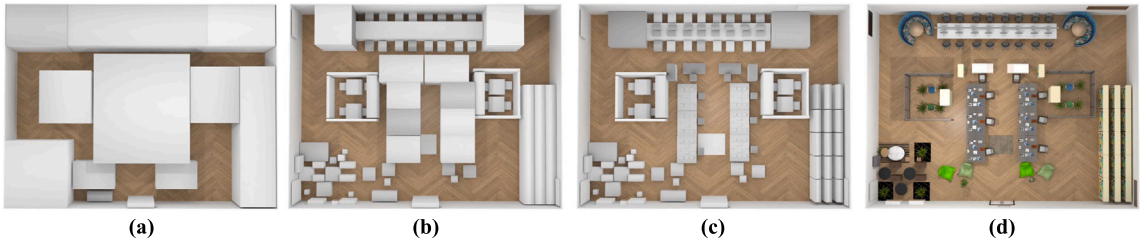

Fig. 1. A coarse-to-fine scene layout designed by our framework with a simple textual input: A library with areas for studying and reading. The scene is generated by gradually expanding the requirements associated with geometric substructures: **(a)** The library is decomposed into a main study area in the middle, book shelves on the right, a children's section at the corner, a computer station at the back... for different functional purposes. **(b)** Each substructure is considered separately and expanded into more requirements: the main study area should be equipped with rich zones for individual study, and the placement of the bookshelf should maximize space utilization while leaving enough walking space, etc. Such requirements lead to more detailed decompositions of substructures. **(c)** This design process is performed iteratively on each substructure until the detail is considered enough. **(d)** Once the scene layout is crafted, various tools are available to elaborate on the details of each item within the scene; for instance, structures can be substituted with assets.

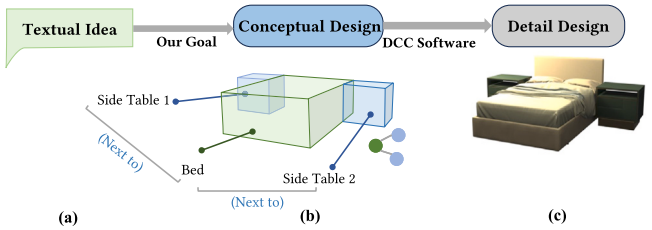

Fig. 2. The design process of designers usually begins with ideas (a) and can be broken down into two distinct phases: conceptual design [10] and detail design. During the conceptual design phase, designers examine the initial user ideas from various perspectives, broaden the requirements, and develop a design proposal that includes a layout of scene structures and their interconnections (b), such as a bubble graph [8]. The next phase focuses on implementing the design proposal in detail. This includes tasks such as arranging items according to the bubble diagram and designing appropriate geometric details for the structure (c).

strategically decomposing the task and designing a structured workflow that allows LLMs to systematically address each stage. Given the existence of established conceptual design theories that provide practical methodologies for scene design workflows [9,10], an effective and expert-level LLM workflow can be developed using these theories as a foundation. Contrasting with current methods employing LLMs for scene layout design [17,18], our conceptual design workflow more logically interprets user concepts by adhering to human design principles.

To achieve this, we design an automated framework for generating scene layouts, following a conceptual design to detail design workflow by integrating LLMs with geometric optimization techniques. We introduce the Iterated Design System (IDS), an adaptive hierarchical framework designed to break down the design process progressively from *coarse to fine* [10]. Each node within this structure can be progressively expanded into child nodes, offering more precise and elaborate semantics. In every unfolding task, the LLM multi-agent system evaluates the node's requirements, expands it into a list [9], and determines its substructures along with constraints on their sizes and positions. Since it is challenging for LLMs to express precise geometric constraints in textual form, we enable them to articulate constraints through the use of our custom-developed domain-specific language (DSL). The DSL narrows down the undecided outputs of LLMs, thereby imposing well-defined constraints on the geometric characteristics of the scene. Subsequently, we develop a geometric optimization based on the mixed integer optimization [19], which identifies a feasible, non-overlapping layout solution satisfying the constraints proposed by LLMs.

However, the optimization may become infeasible due to conflicting constraints introduced by LLMs. To resolve this issue, we employ the Irreducible Infeasible Subsets (IIS) method [20] to eliminate these conflicting constraints. From a mathematical perspective, there are multiple viable solutions to eliminate these contradictions. Each solution may alter the semantic interpretation of the problem, with its significance depending on the specific context. Ranking these solutions based on the degree of semantic change is a labor-intensive task for humans but well-suited for LLMs. Therefore, we propose an LLM-in-the-loop strategy that leverages LLMs to identify and select the semantically optimal solution for resolving constraint contradictions.

Our contributions can be summarized as follows.

- We present a new workflow for scene layout generation based on conceptual design, gradually refining scene details in alignment with the principles of design theory.
- We leverage large language models (LLMs) to implement the conceptual design framework. To this end, we introduce IDS, a flexible hierarchical model, and develop an LLM agent workflow to streamline the generation process.
- Geometry optimization is used to refine the outputs of the LLMs. An LLM-in-the-loop method is introduced to address contradictions while accounting for semantic concerns.

We demonstrate the coarse-to-fine layout conceptual design through complex tasks, see Fig. 1 for an example.

2. Related work

Conceptual design for scene layout. Conceptual design [10] represents a preliminary stage in the design process. For scene layout design, this stage often involves analyzing the requirements and identifying the components within the scene, along with their identities and interactions—this is essential for crafting a logical layout design that adheres to the provided specifications. Many layout generation methodologies, like those using bubble graphs [21,22], depend on a comprehensive conceptual design by the designer. Numerous theoretical design frameworks consider conceptual design as the initial step [9,23]. In particular, [9] introduces a quantized model for conceptual design, where requirements are described in terms of contexts, and the resulting structure designs are delineated as forms. With the significant advancements in large language models (LLMs), it is now possible to transform this into an automated workflow.

Scene layout generation. Recently, the generation of scene layouts from textual idea has garnered significant attention. Traditional optimization-based methods derive scene layouts from bubble graphs [21,22], which should be carefully designed by experts. Meanwhile, data-driven methods learn the layout [24–28] from specific datasets [29–31]. Such techniques prioritize semantic structural details in specific settings, such as indoor arrangements [17,32] and commercial environments [33]. Thus,

these approaches lack generality to broader types of scenes. Recently, LLM-based methods that finetune or use existing large models [34,35] have drawn significant attention [17,18,36–39]. In these works, scene layout is represented by code (e.g., Blender code [40]) and written by LLMs based on human instructions. However, LLMs in these tasks merely serve as tools for generating outputs in specific code formats, without engaging in the actual conceptual design process or leveraging their world knowledge and reasoning capabilities. As a result, some of their outputs do not align with intuitive design expectations. FlairGPT [41] also uses LLMs to analyze multi-level scene nodes for adding details. However, its design is rigidly limited to three layers, making it unable to handle further functional subdivision for finer parts (e.g., on a desktop, it reduces to tertiary object placement). Our recursive approach, by contrast, flexibly supports detail generation at multiple scales.

LLM-based multi-agent framework. Recent works [34,35,42] demonstrate significant progress of LLMs, facilitating their integration into a wide array of tasks spanning multiple domains, like geometric processing [13], robotics [14], and generative projects [15,16]. The Agent framework, in particular, has seen extensive adoption [12,43]. This framework skillfully incorporates the capabilities of LLMs into specific tasks by assigning suitable prompts to the language model [44,45], developing a fitting toolset for the tasks (DSL) [46], and orchestrating their processes (SOP) [12] across large task sets. Our framework utilizes LLMs under the assumption that they have acquired extensive design-oriented knowledge, enabling them to encode design semantics into geometric data within the Agent framework. Some findings reveal the spatial comprehension ability of LLMs [46,47], which serves as our foundational basis.

Layout optimization and conflict constraints handling. The aim of layout optimization is to determine the sizes, positions, and orientations of objects to meet design objectives. [48,49] use a Markov chain Monte Carlo (MCMC)-based method to optimize furniture layouts. [21] proposed a floor plan synthesis approach using mixed-integer quadratic programming (MIQP). Traditional methods handle conflicting constraints in optimization problems through techniques such as the deletion filter [50], the additive method [51], or a hybrid strategy [20] to identify an Irreducible Infeasible Subset (IIS). However, finding an IIS does not immediately explain the reasons for infeasibility. Recently, [52] developed OptiChat, an autonomous system leveraging a LLM, to diagnose infeasible optimization problems. In our study, we utilize LLM agents to reconcile conflicts in layout design, with LLMs aimed at understanding the semantics of optimization expressions.

3. Overview

Problem statement. User can generate a scene by simply writing down design intentions in natural language. Upon user submission of a textual concept, our framework systematically examines it and subsequently generates a hierarchical scene graph, referred to as the Iterated Design System (IDS, Fig. 3). IDS breaks down the scene in a hierarchical manner, from coarse to fine, where each node denotes a subdivided substructure derived from its parent node and is depicted by a bounding box. At coarser levels, nodes represent functional areas (e.g., "rest area", "workspace"); as the recursion proceeds, nodes become finer-grained until they represent concrete objects (e.g., "bed", "desk"), which form the leaf nodes of the hierarchy and can be replaced with 3D assets. Fig. 4 depicts a practical instance of configuring a node based on an initial user input.

Design a node. Each node within IDS is characterized by geometric attributes, specifically the dimension, position, and orientation of its bounding box. Moreover, for each node, we document semantic attributes that provide comprehensive details about the node. We start from a single high-level "description" attribute of the node, and progressively generate additional semantic attributes (e.g., "requirements"), which are eventually distilled into the "description" attribute of each child node. These semantic attributes are formalized as textual information, stored under corresponding keys within the scene node. Thus, each node ultimately contains both geometric attributes (vectors and numerical values) and semantic attributes (text strings). In the following sections, we focus on the task of expanding one node in IDS to multiple substructures, i.e., conceptual design at one node. We propose an LLM multi-agent framework to implement conceptual design (Fig. 4). Taking the basic information in one node, Semantics Analyzer (Section 4) expands and lists the requirements, and then conducts the substructure list, which determines how many children nodes are needed and the descriptions of each child node. Geometry Analyzer (Section 5) takes the abstract requirements and substructure descriptions as input, outputs a scene graph where the geometric constraints of the substructures are listed using DSLs. After the scene graph is generated, we adapt a geometric optimization (Section 6) to ensure the feasibility of the layout plan, which determines the geometric attributes of each sub-nodes. Note that the scene graph provided by LLMs is not always solvable. We propose an LLM-in-the-loop method to deal with cases with contradictions, and help select the best contradiction removal plan.

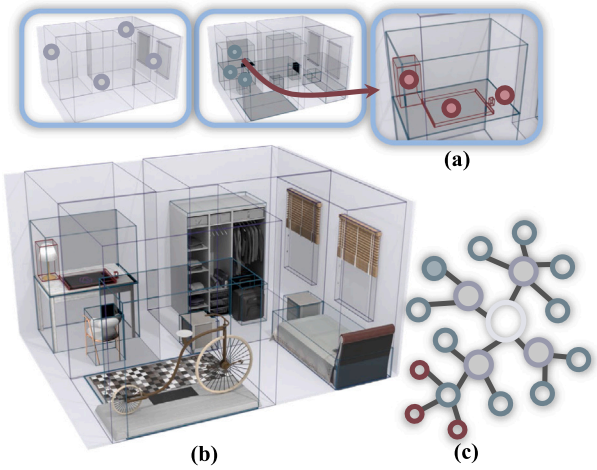

Fig. 3. An example of IDS. We gradually break down the design task into a hierarchical structure (a): Starting with the root node 'Bedroom', which is subdivided into the substructures 'Rest Area', 'Exercise Area', 'Workspace', and 'Storage Area'. Each substructure is further detailed by creating corresponding child nodes. In this scenario, we focus on the desk, yielding a node tree (c) and a detailed design result (b) through asset replacement.

LLM agents. An LLM agent [53] is a framework that integrates the LLM with specific customizations, such as memory storage, DSL tools, and textual prompts. Since multiple agents will be employed to execute the workflow, their configuration represents a critical aspect that demands meticulous customization. In Fig. 4 there are multiple LLM agents that we design for specific tasks. For each agent, the task form (blue text) serves as its memory, the instructions (green text) are prompts that define the task and the output format. Usually, we use several examples to improve the quality of LLM outputs. The tools, defined by several DSLs in the layout in the middle of Fig. 4, enhance the expressive capacity of the text model. More detailed descriptions of the prompts configured for each agent can be found in the supplementary material.

Preprocessor. This is a small step in converting user inputs into the format of an IDS node, as illustrated at the top of Fig. 4. In particular, the node's size must be specified for subsequent procedures. If the user does not specify the size of the root node, LLM will use a 'default' size based on user input and general knowledge. For instance, a 'default' size generated for a teenager's bedroom is 4 m × 5 m × 2.8 m.

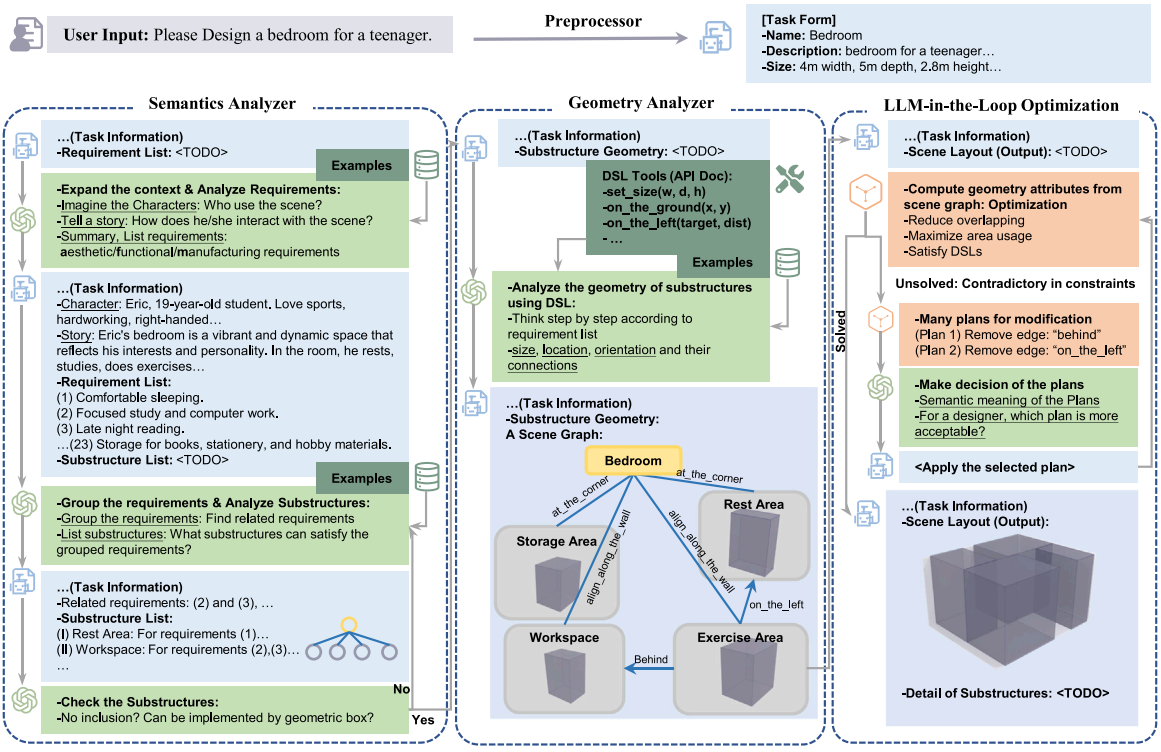

Fig. 4. We harness LLMs to develop the substructural layout for a node within IDS. For the primary node "bedroom", the layout plan is produced through three steps: Semantic Analysis, Geometric Analysis, and LLM-aided Optimization. (a) The Semantic Analyzer extends a list of requirements by examining and hypothesizing about the character's profile and narrative context, subsequently providing related substructure details by categorizing the requirements. (b) The Geometry Analyzer constructs a scene graph using the substructure identities and the requirements list. (c) A layout optimization is employed to convert the graph into a layout plan. If contradictions arise during optimization, we propose potential conflict resolution strategies and utilize LLMs to interpret their significance and make decisions.

4. Semantics analyzer

Semantics analyzer consists of three agents: Requirement analyzer, substructure analyzer and a substructure checker.

Requirements analyzer agent. We are broadening our input by introducing the requirement analysis instruction found in [9]. The requirement agent is tasked with creating a requirement list by taking into account at least the following factors, similar to those considered by humans as depicted in Fig. 2:

1. Functional requirements. What structure should be contained in the node to fulfill its functionality?
2. Aesthetic requirements. Is there any requirements for layout style? For example, the scene could be cozy, simple.
3. Manufacturing requirements. What issues should be considered to determine the geometric attributes and realize the scene? For example, the height of building should be suitable for normal height of human, the layout on a table should be adapted to the user's dominant hand.

The result is a list of clear and concise requirements, following the theoretical instructions [9].

From the comprehensive input, an extensive requirement list should be formulated. Subsequently, the agent is tasked with drafting a narrative about the scene initially. As illustrated in Fig. 4, the agent is expected to initially make logical and thorough assumptions concerning the primary characters within the scene, followed by composing a narrative that explains how the characters engage with the scene. The story should also allude to the inherent functionality and interconnections present. Since the input provided by users is often unprofessional or incomplete, the requirement analysis in conceptual design necessarily involves making reasonable and essential assumptions and inferences, such as hypothesizing additional information about the characters in the scene or the activities they might perform, in order to derive more complete and actionable information about the substructures.

Substructure analyzer agent. Substructures can be inferred from the requirement list, following the way of requirements grouping [9]. Another LLM agent is employed to analyze this issue. This agent leverages the remarkable ability of LLMs to summarize and extract key information from a long context. Importantly, the agent decides whether to decompose the current node into finer sub-regions (e.g., "storage area") or into concrete objects (e.g., "wardrobe", "shelf"), based on the current scale and semantic context of the node. As demonstrated in Fig. 5, we categorize the associated requirements into groups: (1) a rest area for sleeping; (2) a space for studying, using a computer, and nighttime reading; (3) a necessity for an exercise area equipped with a treadmill... The configurations on the right in Fig. 5 illustrate the shapes of the respective contexts. To reduce the mistakes made by this agent, we introduce a substructure checker agent (Fig. 4) that evaluates the semantic validity of the substructure list generated by the substructure analyzer. Its primary function is to detect semantic redundancies. For example, if both "workstation" and "workplace" appear, it is evident that "workstation" should be considered part of "workplace", and thus eliminated. The checker operates by taking the substructure list and the original requirement text as input, identifying overlapping meanings, and returning the redundant substructure names as textual feedback to the substructure analyzer, which then re-runs the substructure analyzer. This correction process can be repeated iteratively; once the maximum iteration count is reached, the checker resolves any remaining redundancy by retaining only one of the overlapping substructures and discarding the others (suggested by the agent), yielding the final refined substructure list.

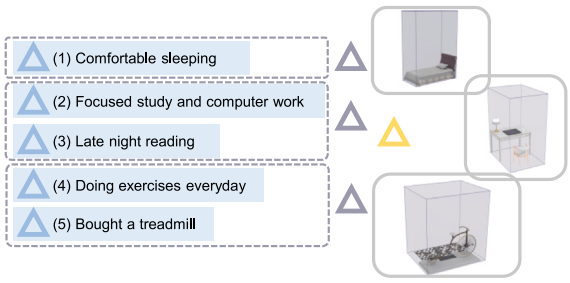

Fig. 5. The substructure agent examines the requirements list (Blue triangles) and categorizes related terms. Each cluster of requirements (Purple triangles) is assigned a substructure. For instance, a resting area for a bed is necessary for comfortable sleep. Similarly, a study area is essential for effective studying and reading.

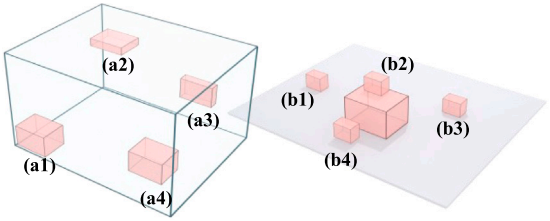

Fig. 6. Illustrations of operators under our DSL for the layout of structures. (a1) `at_the_corner`, (a2) `hang_on_upper_surface`, (a3) `align_along_the_right_side`, (a4) `on_the_ground`. (b1) `in_front_of`, (b2) `on_top_of`, (b3) `around`, (b4) `on_the_left_side`.

5. Geometry analyzer

The second phase focuses on the identification of geometric attributes in each substructure. Directly handling 3D mathematical calculations is challenging for LLMs. Therefore, a general approach is to develop a set of tools for geometric operations, which LLMs can call upon. We detail these tools using a domain specific language we have defined.

Design DSL for LLMs. We design a series of operators under a DSL in scene layout problem for LLMs. Our core objective is to minimize the dimensions of the problem, that is, to reduce the number of parameters that need to be input. For example, placing a box in the scene usually has three freedoms. We introduce operators like "`on_the_ground(x, y)`", "`align_along_the_front_side(x, height)`" which reduce the freedom to 2 and make it easier for LLMs. In addition to the operator that sets the size of a structure, our operator can be categorized into two types: one that describes the relationship between substructures and parent structures and another that describes the relationships between substructures. The two categories of operators can be visualized in Fig. 6. Each DSL is equipped with a tiny document which describes its usage for LLMs.

Workflow. Fig. 4 illustrates that the geometry analyzer is responsible for determining DSLs associated with size, location, orientation, and interconnections of substructures. We utilize three LLM agents—size analyzer, connection analyzer, and layout analyzer, which operate in a sequential manner. Initially, the DSL "`set_size(width, depth, height)`" specifies the dimension of each box, where the depth direction is defined as front to back. Once the size is established, it is promptly recorded in memory, allowing the connection analyzer to select the connection DSL tools that impose constraints among the substructures. The layout analyzer then determines the relationship of the substructure with its parent, including the orientation of boxes, considering only axial directions. The resulting list of DSL tools creates a scene graph, with layout constraints indicated on vertices and edges.

6. LLM-in-the-loop layout optimization

Formulation. The output of the previous group of agents forms a set of DSL tools. In order to translate the textual representation into geometry, we adapt an optimization, maximizing the space usage while ensuring that no overlapping occurs. We use N_i to represent the i−th node. The variable in the optimization is n vectors $(w_i, d_i, h_i, x_i, y_i, z_i) \in \mathbb{R}^6$ where w_i, d_i, h_i are the size of N_i and (x_i, y_i, z_i) denote the coordinates of its center point. We fix the orientation of each structure in the optimization. So it is sufficient to assume that w_i is along the x axis, d_i is along the y axis, and h_i is along the z axis. The first objective is the use of space in the parent structure can be represented by the dominant area of all substructures projected to the ground.

$$E_{\text{area}} = 1 - \sum_{N_i \text{ on the ground}} \frac{w_i d_i}{W D}, \tag{1}$$

where we denote the size of the parent structure as (W, D, H). For each selected DSL $f \in T_{DSL}$, we can define an objective E_f that ensures that the result of f is satisfied as much as possible. For example, the size DSL "`set_size`(N_i, w^*, d^*, h^*)" can be written in the optimization as

$$E_{\text{Size},i} = \left(\frac{w_i - w_i^*}{W}\right)^2 + \left(\frac{d_i - d_i^*}{D}\right)^2 + \left(\frac{h_i - h_i^*}{H}\right)^2, \tag{2}$$

the "`on_the_left_side`(N_i, N_j, d_{ij})" can be written as

$$E_{\text{OnTheLeftSide},i,j} = \left(\frac{x_j - w_j/2 - x_i - w_i/2 - d_{ij}}{W}\right)^2. \tag{3}$$

The objective definition for other DSL tools is in supplementary.

Constraints are derived to ensure nonoverlapping. On the one hand, structure boxes should not exceed the parent boundary. For example, in x direction we should make sure

$$|x_i| \le \text{Width}(N) - w_i/2. \tag{4}$$

On the other hand, structure boxes should not overlap with each other. In x direction, there is

$$x_i - x_j \ge \frac{w_i}{2} + \frac{w_j}{2} - M(1 - \sigma_L), \tag{5}$$

$$x_j - x_i \ge \frac{w_j}{2} + \frac{w_i}{2} - M(1 - \sigma_R), \tag{6}$$

where M and binaries σ_L, σ_R are auxiliary variables identifying the separation of N_i and N_j on the x direction. Here, we adopt the standard big-M formulation, where M is a sufficiently large constant that relaxes one inequality depending on the binary decision variable, and $\sigma_L, \sigma_R \in \{0, 1\}$ encode whether N_i lies to the left or right of N_j. Other constraints are derived from DSL. For example, "`on_the_left_side`(N_i, N_j, d_{ij})" needs that the distance along x direction should be sufficiently large:

$$x_j - x_i \ge \frac{w_j}{2} + \frac{w_i}{2}. \tag{7}$$

To this end, taking λ_k as weights, the overall objective is

$$E = E_{\text{area}} + \sum_{f \in \boldsymbol{dsl}(N)} \lambda_f E_f. \tag{8}$$

Challenge. The introduction of binary variables in the nonoverlapping constraints and quadratic objectives makes the overall optimization an MIQP problem [21]. Nonetheless, it encounters issues when conflicting constraints are present, particularly when constraints are provided unreliably by LLMs. For instance, consider two boxes N_i, N_j: if N_i is positioned to the left of N_j and N_j to the left of N_i, a contradiction arises. Consequently, some constraints must be altered or removed to create a solvable system. Conventional algorithms can identify various modification strategies, but ultimately leave the selection to users or heuristic methods. Unfortunately, in many cases, users or heuristics fail to choose the "optimal" modification strategy.

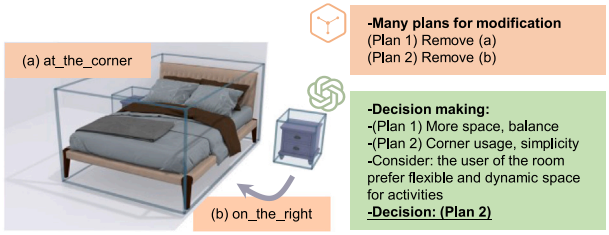

Fig. 7. Contradictory constraints and candidate plans to resolve them. The candidate plans are generated by ComputeIIS, each representing a specific constraint modification (e.g., removing a constraint or adjusting a parameter). The LLM acts as a decision-maker, evaluating the semantic implications of each plan and selecting the most reasonable one based on design context..

LLM-in-the-loop approach. We introduce an LLM-in-the-loop method to decide when conflicts emerge and to propose several modification plans. Assume that $\{\text{Plan}(i)\}$ is generated by certain detection algorithms. In our approach, we adapt the `ComputeIIS` module from Gurobi [54]. A plan evaluation agent is used to choose the optimal $\text{Plan}(i^*)$. It is important to note that the LLM does not directly reason about geometric or spatial relationships mathematically. Instead, once `ComputeIIS` identifies conflicting constraints, the algorithm generates multiple candidate plans, each corresponding to a different way of relaxing the conflicts (e.g., removing a constraint, adjusting a parameter, or eliminating a substructure). These plans are then translated into natural language descriptions that convey their semantic implications. For example, "Plan 1: remove the structure bookshelf" versus "Plan 2: modify the size of the bookshelf". The LLM evaluates these semantic descriptions based on design context and common-sense reasoning (e.g., "a bedroom without storage is inconvenient"), rather than performing spatial computation. Once the LLM selects a plan, the corresponding constraint modification is applied automatically, and the optimization is re-executed. While it is challenging for us to provide a quantitative evaluation of the plans' quality, the LLM agent can convert plans into semantic representations related to the scene's design and make decisions through reasoning. Refer to Fig. 7 for a small example. Alg. 1 illustrates the LLM-in-the-loop optimization process. Note that Alg. 1 is guaranteed to terminate, since the number of constraints is finite and, within a fixed maximum number of iterations, we force the agent to select a constraint-removal plan.

ALGORITHM 1: LLM-in-the-loop Optimization

```
while hasConflicts() do
    {Plan(i)} = PlanDetection() // We use ComputeIIS() in Gurobi
    Plan(i*) = Agent_conflict({Plan(i)}).
    Adapt Plan(i*)
end while
return Optimize()
```

7. Experimental results

In this section, we assess the outcomes of scene layout conceptual design via our approach. We perform ablation studies tailored to the LLM-in-the-loop optimization within our framework. Furthermore, we undertake a comparative analysis focusing on LLM model selection and parameter adjustment, confirming the effectiveness of the latest LLM innovations. Our approach is applied to diverse input types to evaluate the generative capabilities in conceptual design.

7.1. Implementation details

The core of our multi-agent implementation is LLM. Here we adapt GPT-4o [55] model as the core of our LLM agents. In calling the API, temperature is the parameter that influences LLM creativity [56], which is recommended to be chosen between 0 and 2 [56]. We set the temperature at 0.8 to strike a balance between creativity and coherence. At each task of expanding a structure into substructures, it usually takes about 60k tokens in total, where the token usage in each agent is listed in Table 1. For more detailed settings in agent prompting, please refer to the Supplementary material.

7.2. Qualitative results

Fig. 8 shows several results generated by our implementation. With the implementation of conceptual design, we can generate various types of scenes with flexible complexity by analyzing, expanding, and implementing the design requirements.

Fig. 9 illustrates the diversity of our results within the same scene category. Given a straightforward input: 'create a bedroom with a storage unit' (the second row in Fig. 9), our approach infers the character's profile, and creates relevant substructures. Fig. 9 showcases indoor and outdoor renderings of a garden, highlighting our method's adaptability to various scene types.

7.3. Evaluations

It is difficult to measure the result of conceptual design in a computational way. However, in [9,10] several experiential metrics are introduced. In this section, we aim to analyze our results based on fidelity using various inputs, and contrast our findings with other scene generation techniques.

Fidelity analysis. We test the fidelity of our method in detecting and executing input requirements. We have generated 20 groups of inputs in 5 categories. Each group of inputs contains one simple requirement, one complex requirement with more semantic instructions, and one complex requirement with geometric instructions, which has more instructions in scale, amount of substructures, and layout than the simple one, and contains a negative requirement, for example:

1. Simple input: A classroom with a podium and some desks.
2. Input with semantic instructions: A classroom *for university students' laboratory course teaching*.
3. Input with geometric instructions: A classroom. There is a *2-meter-long* podium *in the front area of* the classroom. There is a trash can next to the podium. There are *8 desks* in the center area of the classroom. This classroom *has no back door*.

We compare the results of three inputs in their rate of correctness under each additional requirement. This "rate of correctness" is quantitatively measured as the ratio of the number of outputs that meet the input requirements to the total number of outputs. We asked three users to assess the 20 groups of results (60 outputs in total) against the additional instructions provided in the inputs. Using the classroom example above, participants judged whether each output satisfied the following aspects:

- Quantity: whether the number of desks matched the requirement of *8 desks*;
- Size: whether the podium conformed to the requirement of being *2 meters long*;
- Position: whether the podium was correctly located *in the front area of* the classroom;
- Negative requirement: whether the classroom correctly *had no back door*;
- Semantic requirement: whether the classroom matched the usage context *for university students' laboratory course teaching*.

Fig. 8. Indoor scene layout design using our approach. Through conceptual design, scene generation is not constrained by specific complexities or types. For each scene type illustrated in the figure, a concise textual description serves as the input for our method.

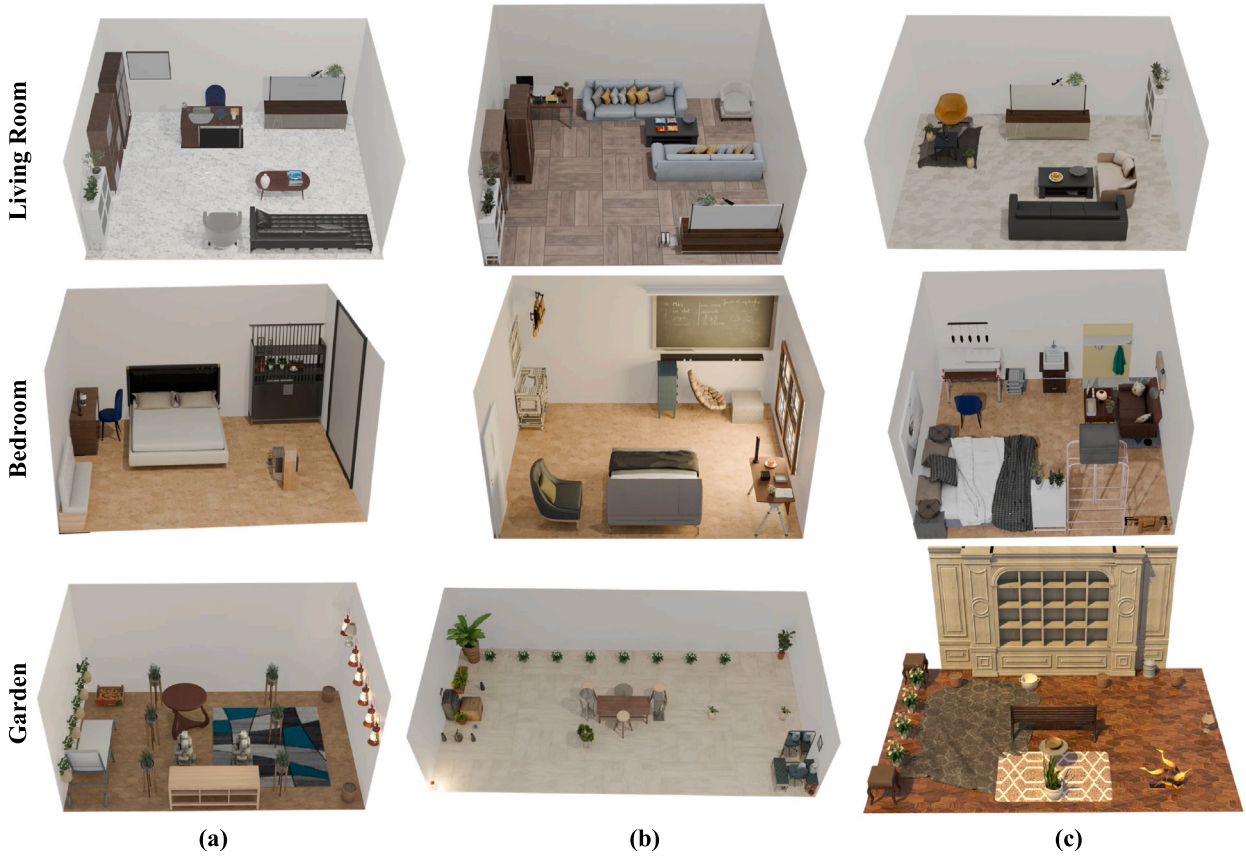

Fig. 9. Diversity of our method within the same scene category. Our conceptual-design-based framework is not limited to generating indoor scene layouts. For instance, given the description "a small, well-organized terrace garden designed to offer a serene space for relaxation and enjoyment", both indoor (a, b) and outdoor (c) gardens can be generated.

Table 1
Token usage in our LLM multi-agent workflow. The substructure analyzer and substructure checker agent performs multiple times in one workflow.

Agents	Requirement analyzer	Substructure analyzer	Substructure check	Size analyzer	Connection analyzer	Layout analyzer	Optimization
Input	3921	8857/9731/9730	1131/1131/1131	3263	5582	6166	5672
Output	342	1034/1033/1025	547/553/383	425	535	822	347

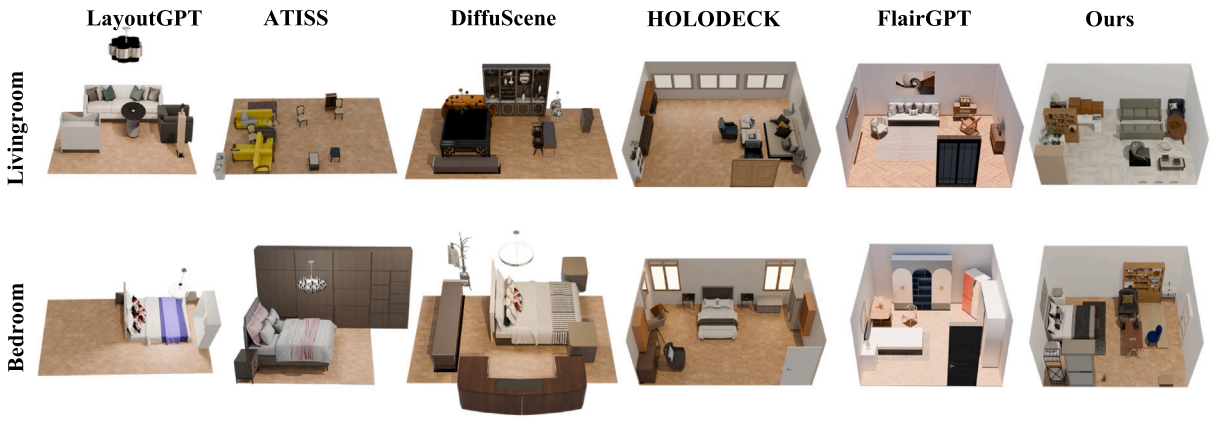

Fig. 10. Comparison with similar indoor layout generation studies. Our results yield valid layouts ensured through optimization and provide more intricate details by progressively designing the scene's substructures. Additionally, our approach effectively prevents objects from exceeding the boundaries.

Table 2
Comparison of rates of correctness among simple input, complex input with semantic instructions, complex input with geometric instructions, under different types of additional requirements. The results highlight the ability of the model to meet the needs of complex text input (Compared with the control group).

Metric	Simple input	Semantic	Geometric
Quantity	5.0%	10.0%	**90.0%**
Size	0.0%	0%	**81.7%**
Pos	48.3%	61.7%	**85.0%**
Negative	35.0%	45.0%	**90.0%**
Semantic	30.0%	**83.3%**	31.7%
Rank (↓)	1.9	**1.85**	2.25

The simple input, which did not provide any of these instructions, served as a control group to verify whether our model would strictly follow user-provided requirements. In addition, participants were asked to provide an overall quality ranking of the three inputs (considering layout and content coherence). To further reduce bias among the options, we also inserted several distractor outputs in which objects were randomly removed or their attributes and layouts were modified. Table 2 shows the comparison.

Compared to the control group which showed significantly weaker adherence to additional semantic and geometric requirements, we observed that adding textual instructions effectively controlled the outputs from semantic features to geometric attributes. This indicates that more deterministic instructions reduce the assumptions made by the agents. On the other hand, the overall quality ranking revealed that the three input types were rated similarly in terms of scene plausibility. With stricter constraints on number, size, and position, the overall quality slightly decreased, which may be due to certain requirements introducing potential inconsistencies, making the results less harmonious.

In practical applications, users may choose either to provide complex inputs for highly customized results, or to use simpler and more ambiguous prompts to generate multiple candidate outputs at once, among which different assumptions are explored by the model and one can be selected as the most satisfactory design.

Comparisons. We compare the performance in the indoor dataset by analyzing the results under the same input with previous layout generation works [17,28,36,37,41]. Fig. 10 shows the result of the comparison. To evaluate the results quantitatively (Table 3), we use (1) GPT-4o [57], (2) CLIP model [58] and (3) User study of 3 human participants to evaluate the results of all candidates according to how much the scene layout agree with the textual input. We test 50 results across 3 types of input: bedroom, living room, dining room, since the available scene type of [36,37] are limited (Textual inputs are in the supplementary, note that the work ATISS [37] and LayoutGPT [36] does not support detailed textual input, we only use types as input in the two methods). Specifically, we let GPT-4o give a score of 0–10 to the top-down view of the scene in terms of aesthetics, layout rationality, and detail richness, and calculate the average. Then we input the top-down view of the scene and the text "a top-down view of [scene type]" into the clip model to calculate the average clip score. For human participants, we let them score the results from the common perspective views. The results in Table 3 indicate that our method achieves a clear advantage in terms of detail richness, owing to the IDS' recursive nature: at each iteration, it adds sufficient semantic details, in contrast to the fixed generation pipelines used in previous methods.

7.4. Ablation study

Substructure checker agent. We designed an experiment to assess the necessity of the substructure checker agent. We examined the outputs of the substructure analyzer agent on 50 input examples, using an LLM agent to identify the presence of inclusion substructures and recorded the inclusion rate. Table 4 presents a comparison of results without the checker, with one round of checking and correction, and with two rounds.

LLM-in-the-loop optimization. We apply a layout optimization method based on MIQP to achieve a valid layout solution. Fig. 11 illustrates the results with and without this optimization. Using MIQP, we attain nonoverlapping layouts (Fig. 11, middle and right). Nonetheless, poor problem settings might lead to a lack of solutions. As shown in Fig. 11(a), when the total size surpasses the box's capacity during the decomposition of a substructure within a moving box, even though individual substructure sizes can be adjusted slightly. In the case of

Table 3
Comparison of the CLIP score, GPT-4o and human evaluation between related scene synthesis works. The results indicate that the scenes generated by our approach achieve the highest CLIP scores, demonstrating the strongest alignment with the input room types. Furthermore, our scenes rank first or second in aesthetics, layout rationality, and detail richness, which highlights their overall quality and alignment with user expectations.

Evaluation metrics (↑)	LayoutGPT	ATISS	HOLODECK	DiffuScene	FlairGPT	Ours
CLIP score	28.8	28.4	29.9	29	29.9	**30.3**
Aesthetics (by GPT-4o)	7.0	6.4	7.0	6.9	7.1	**7.3**
Layout rationality (by GPT-4o)	**7.9**	6.6	7.1	7.2	7.5	7.7
Richness of details (by GPT-4o)	4.6	4.3	5.3	4.8	4.8	**5.4**
Aesthetics (by human)	6.4	6.0	7.2	6.9	7.5	**7.7**
Layout rationality (by human)	5.4	5.4	7.5	5.8	7.8	**7.8**
Richness of details (by human)	4.6	5.0	7.5	6.0	7.5	**8.2**

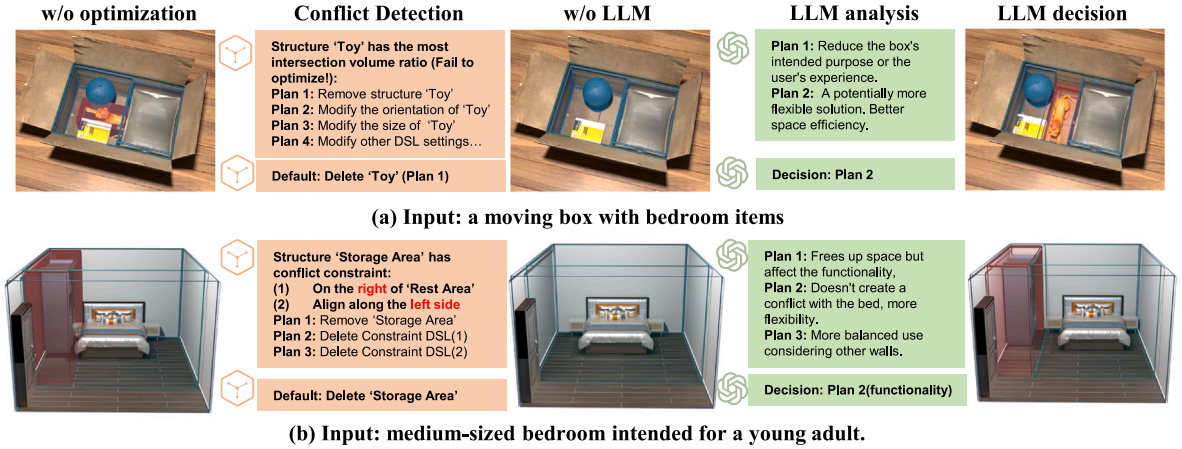

Fig. 11. Two cases where the contradiction conflict analyzer is effective: (a) a moving box filled with bedroom items that exceeds the capacity limit. The LLM choose to adjust the orientation of the 'toy model' (right) to try to maximize the usage, instead of remove it (middle); (b) is a bedroom where previous LLM agents require that the 'Storage Area' with shelf should align along the left side of the room as well as be placed to the right of the 'Rest Area' with a bed. LLM detects the mistake in previous agents and remove the ridiculous one (c), instead of simply remove one structure (middle).

Table 4
Comparison of rates of inclusion between w. and w/o. substructure checker agent. The inclusion is detected by another LLM agent among 50 inputs. If the output has logical inclusions like containing both "reading desk" and "reading area", then it is labeled as a case with inclusion. Then we calculate the rate of such cases.

	w/o.	one-round	two-round
Rate of Inclusion	58.0%	22.0%	8.0%

the heuristic approach, the 'toy' structure is directly removed (Fig. 11(a), middle). The contradiction analyzer agent identifies the need for spatial efficiency with the moving box, prompting the LLM to modify its orientation. The optimization then achieves better solution.

In Fig. 11(b), two constraints of the storage area conflict with each other: "`align_along_the_left_side`" and "`on_the_right_of('Rest Area')`". This occurs when the LLM occasionally confounds the orientation between the parent structure 'Bedroom' and the target structure 'Rest Area'. In the room, the right side of the bed aligns with the left wall, which is contrary to "`on_the_right_of('Rest Area')`" in the parent structure's coordinates. Using the heuristic approach, we eliminate the shelf structure. However, the Contradiction Analyzer appreciates the presence of the shelf and the "`align_along_the_left_side`" for functionality. Consequently, it discards "`on_the_right`", leading to a more logical outcome.

8. Conclusion

In this paper, we develop an LLM multi-agent workflow to implement conceptual design for scene layout generation. Based on the theoretical and experiential guidelines, our implementation leverages the power of LLMs in handling the highly reason-driven process, by proposing a novel hierarchical representation IDS, and a theoretically consistent agent workflow. We demonstrate acceptable results under various kinds of inputs, while testing the potential applications and properties of this approach. Owing to the inherent randomness of LLMs, failure cases may arise where LLMs are unable to decompose the scene design into logical structures or provide contradictory guidance during optimization. To this end, we propose two LLM agents for validation and decision making in the workflow, which can reduce failure cases and make reasonable decisions. We believe that the performance of our multi-agent framework will be significantly enhanced with the growth of LLMs.

In the future, we can explore further applications of this conceptual design implementation in various design domains and refine the pipeline. The improvement can also be investigated in two aspects: Firstly, one can improve the efficiency of the multi-agent workflow. It is interesting to discuss how to minimize the use of LLM tokens to reach similar reasoning quality. Additionally, incorporating more geometric and quantitative guidelines, along with more robust and expressive DSLs, into the conceptual design can alleviate the burden of causal inference for LLMs while enhancing the robustness and reliability of the results. At present, our method mainly treats box-shaped substructure partitioning; for non-rectangular regions, future work can extend the framework by introducing forbidden sub-areas or similar mechanisms to achieve the desired effect.

The empirical findings from our study substantiate that the multi-agent framework represents a viable approach for conceptual design of scene layout, as well as numerous reasoning-intensive challenges in practical applications.

CRediT authorship contribution statement

Wenzheng Wu: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Chucheng Xiang:** Methodology. **Zhi Lin:** Data curation. **Yirui Guan:** Visualization. **Ruchao Bao:** Methodology. **Zhongyuan Liu:** Conceptualization. **Ziqi Wang:** Writing – review & editing, Conceptualization. **Ligang Liu:** Resources, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (62025207).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2026.104553.

Data availability

No data was used for the research described in the article.

References

[1] Zhang S-H, Zhang S-K, Liang Y, Hall P. A survey of 3D indoor scene synthesis. J Comput Sci Tech 2019;34:594–608.
[2] Verdie Y, Lafarge F, Alliez P. LOD generation for urban scenes. ACM Trans Graph 2015;34(3):30.
[3] Li Z, Li Z, Cui Z, Pollefeys M, Oswald MR. Sat2Scene: 3D urban scene generation from satellite images with diffusion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, p. 7141–50.
[4] Kumaran V, Rowe J, Mott B, Lester J. Scenecraft: Automating interactive narrative scene generation in digital games with large language models. In: Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment, vol. 19, (1):2023, p. 86–96.
[5] Ayyildiz DV, Alnaser AJ, Taj S, Zakaria M, Jaimes LG. A survey of learning techniques for virtual scene generation. SAE Int J Connect Autom Veh 2024;8(12-08-02-0017).
[6] Sun J, Wu W, Liu L, Min W, Zhang G, Zheng L. Wallplan: synthesizing floorplans by learning to generate wall graphs. ACM Trans Graph 2022;41(4):1–14.
[7] Freiknecht J, Effelsberg W. A survey on the procedural generation of virtual worlds. Multimodal Technol Interact 2017;1(4):27.
[8] Merrell P, Schkufza E, Koltun V. Computer-generated residential building layouts. In: ACM SIGGRApH Asia 2010 papers. 2010, p. 1–12.
[9] Alexander C. Notes on the Synthesis of Form, vol. 5, Harvard University Press; 1964.
[10] Andreasen MM, Hansen CT, Cash P. Conceptual design. Cham, Switzerland: Springer; 2015.
[11] Li G, Hammoud HAAK, Itani H, Khizbullin D, Ghanem B. Camel: Communicative agents for "mind" exploration of large scale language model society. 2023, arXiv preprint arXiv:2303.17760.
[12] Hong S, Zheng X, Chen J, Cheng Y, Wang J, Zhang C, Wang Z, Yau SKS, Lin Z, Zhou L. Metagpt: Meta programming for multi-agent collaborative framework. 2023, arXiv preprint arXiv:2308.00352.
[13] Abdelreheem A, Eldesokey A, Ovsjanikov M, Wonka P. Zero-shot 3D shape correspondence. In: SIGGRAPH Asia 2023 conference papers. SA '23, Association for Computing Machinery; 2023.
[14] Vemprala S, Bonatti R, Bucker A, Kapoor A. Chatgpt for robotics: Design principles and model abilities. Microsoft Auton Syst Robot Res 2023;2:20.
[15] Park JS, O'Brien J, Cai CJ, Morris MR, Liang P, Bernstein MS. Generative agents: Interactive simulacra of human behavior. In: Proceedings of the 36th annual ACM symposium on user interface software and technology. 2023, p. 1–22.
[16] Wang Y, Xian Z, Chen F, Wang T-H, Wang Y, Fragkiadaki K, Erickson Z, Held D, Gan C. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. 2023, arXiv preprint arXiv:2311.01455.
[17] Yang Y, Sun F-Y, Weihs L, VanderBilt E, Herrasti A, Han W, Wu J, Haber N, Krishna R, Liu L, Callison-Burch C, Yatskar M, Kembhavi A, Clark C. Holodeck: Language guided generation of 3D embodied AI environments. In: The IEEE/CVF conference on computer vision and pattern recognition. Seattle, Washington: IEEE/CVF; 2024.
[18] Aguina-Kang R, Gumin M, Han DH, Morris S, Yoo SJ, Ganeshan A, Jones RK, Wei QA, Fu K, Ritchie D. Open-universe indoor scene generation using llm program synthesis and uncurated object databases. 2024, arXiv preprint arXiv:2403.09675.
[19] Floudas CA. Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford University Press; 1995.
[20] Guieu O, Chinneck JW. Analyzing infeasible mixed-integer and integer linear programs. INFORMS J Comput 1999;11(1):63–77.
[21] Wu W, Fan L, Liu L, Wonka P. MIQP-based layout design for building interiors. Comput Graph Forum 2018;37(2):511–21.
[22] Hu R, Huang Z, Tang Y, Van Kaick O, Zhang H, Huang H. Graph2plan: Learning floorplan generation from layout graphs. ACM Trans Graph 2020;39(4). 118:1–118:14.
[23] Lynch K. The image of the city. MIT Press; 1964.
[24] Li Y, Xu P, Ren J, Shao Z, Huang H. Gltscene: Global-to-local transformers for indoor scene synthesis with general room boundaries. In: Computer graphics forum, vol. 43, (no. 7):Wiley Online Library; 2024, e15236.
[25] Li M, Patil AG, Xu K, Chaudhuri S, Khan O, Shamir A, Tu C, Chen B, Cohen-Or D, Zhang H. Grains: Generative recursive autoencoders for indoor scenes. ACM Trans Graph 2019;38(2):1–16.
[26] Lin C, Mu Y. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. 2024, arXiv preprint arXiv:2402.04717.
[27] Wang K, Savva M, Chang AX, Ritchie D. Deep convolutional priors for indoor scene synthesis. ACM Trans Graph 2018;37(4):1–14.
[28] Tang J, Nie Y, Markhasin L, Dai A, Thies J, Nießner M. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, p. 20507–18.
[29] Deitke M, Schwenk D, Salvador J, Weihs L, Michel O, VanderBilt E, Schmidt L, Ehsani K, Kembhavi A, Farhadi A. Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023, p. 13142–53.
[30] Wu T, Zhang J, Fu X, Wang Y, Ren J, Pan L, Wu W, Yang L, Wang J, Qian C, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023, p. 803–14.
[31] Zheng J, Zhang J, Li J, Tang R, Gao S, Zhou Z. Structured3d: A large photo-realistic dataset for structured 3d modeling. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part IX 16. Springer; 2020, p. 519–35.
[32] Fang C, Hu X, Luo K, Tan P. Ctrl-room: Controllable text-to-3D room meshes generation with layout constraints. 2023, arXiv preprint arXiv:2310.03602.
[33] Zhang S-K, Liu J-H, Li Y, Xiong T, Ren K-X, Fu H, Zhang S-H. Automatic generation of commercial scenes. In: Proceedings of the 31st ACM international conference on multimedia. 2023, p. 1137–47.
[34] OpenAI R. Gpt-4 technical report. 2023, arxiv:2303.08774, View in Article 2:13.
[35] Liu A, Feng B, Xue B, Wang B, Wu B, Lu C, Zhao C, Deng C, Zhang C, Ruan C, et al. Deepseek-v3 technical report. 2024, arXiv preprint arXiv:2412.19437.
[36] Feng W, Zhu W, Fu T-j, Jampani V, Akula A, He X, Basu S, Wang XE, Wang WY. Layoutgpt: Compositional visual planning and generation with large language models. Adv Neural Inf Process Syst 2023;36:18225–50.
[37] Paschalidou D, Kar A, Shugrina M, Kreis K, Geiger A, Fidler S. Atiss: Autoregressive transformers for indoor scene synthesis. Adv Neural Inf Process Syst 2021;34:12013–26.
[38] Fu R, Wen Z, Liu Z, Sridhar S. Anyhome: Open-vocabulary generation of structured and textured 3d homes. In: European conference on computer vision. Springer; 2024, p. 52–70.
[39] Çelen A, Han G, Schindler K, Van Gool L, Armeni I, Obukhov A, Wang X. I-design: Personalized llm interior designer. 2024, arXiv preprint arXiv:2404.02838.
[40] Hu Z, Iscen A, Jain A, Kipf T, Yue Y, Ross DA, Schmid C, Fathi A. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In: Forty-first international conference on machine learning. 2024.
[41] Littlefair G, Dutt NS, Mitra NJ. FlairGPT: Repurposing LLMs for interior designs. In: Computer graphics forum. Wiley Online Library; 2025, e70036.
[42] Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, et al. Llama 2: Open foundation and fine-tuned chat models. 2023, arXiv preprint arXiv:2307.09288.
[43] Zhang S, Gong C, Wu L, Liu X, Zhou M. AutoML-GPT: Automatic machine learning with GPT. 2023, arXiv preprint arXiv:2305.02499.
[44] Zeng A, Liu M, Lu R, Wang B, Liu X, Dong Y, Tang J. Agenttuning: Enabling generalized agent abilities for llms. 2023, arXiv preprint arXiv:2310.12823.
[45] Saha S, Levy O, Celikyilmaz A, Bansal M, Weston J, Li X. Branch-solve-merge improves large language model evaluation and generation. 2023, arXiv preprint arXiv:2310.15123.

[46] Makatura L, Foshey M, Wang B, HähnLein F, Ma P, Deng B, Tjandrasuwita M, Spielberg A, Owens CE, Chen PY. How can large language models help humans in design and manufacturing?. 2023, arXiv preprint arXiv:2307.14377.
[47] Gurnee W, Tegmark M. Language models represent space and time. 2023, arXiv preprint arXiv:2310.02207.
[48] Yu L-F, Yeung S-K, Tang C-K, Terzopoulos D, Chan TF, Osher SJ. Make it home: automatic optimization of furniture arrangement. ACM Trans Graph 2011;30(4).
[49] Merrell P, Schkufza E, Li Z, Agrawala M, Koltun V. Interactive furniture layout using interior design guidelines. In: ACM SIGGRAPH 2011 papers. New York, NY, USA: Association for Computing Machinery; 2011.
[50] Chinneck JW, Dravnieks EW. Locating minimal infeasible constraint sets in linear programs. ORSA J Comput 1991;3(2):157–68. http://dx.doi.org/10.1287/ijoc.3.2.157.
[51] Tamiz M, Mardle SJ, Jones DF. Detecting IIS in infeasible linear programmes using techniques from goal programming. Comput Oper Res 1996;23(2):113–9.
[52] Chen H, Constante-Flores GE, Li C. Diagnosing infeasible optimization problems using large language models. 2023, arXiv:2308.12923.
[53] Huang W, Wang C, Zhang R, Li Y, Wu J, Fei-Fei L. Voxposer: Composable 3d value maps for robotic manipulation with language models. 2023, arXiv preprint arXiv:2307.05973.
[54] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2024, URL https://www.gurobi.com.
[55] OpenAI. Hello GPT-4o. 2024, https://openai.com/index/hello-gpt-4o/.
[56] OpenAI. How should I set the temperature parameter? 2024, https://platform.openai.com/docs/guides/text-generation/how-should-i-set-the-temperature-parameter.
[57] Wu T, Yang G, Li Z, Zhang K, Liu Z, Guibas L, Lin D, Wetzstein G. Gpt-4v (ision) is a human-aligned evaluator for text-to-3d generation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024, p. 22227–38.
[58] Hessel J, Holtzman A, Forbes M, Bras RL, Choi Y. Clipscore: A reference-free evaluation metric for image captioning. 2021, arXiv preprint arXiv:2104.08718.